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

D2.3 - Data Collection and Representation - Final 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, and Spain. This document reflects only the author's view and 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:	July 31, 2023
Authors:	Sergio Morales (UOC), Julio Medina (UCAN), and Representatives of all the involved Solution and Use Case providers via their contribution to the project's common Modelio model, which has all their data models.
Contributing Partners:	ABI, ACO, AND, AST, AVL, BT, CAMEA, CSY, DT, HIB, IMTA, ITI, PRO, ROTECH, SOFT, TEK, UCAN, UOC, VCE, WESTMO.
Date:	Jul 28, 2023
Version	V1
Revision:	06
Abstract:	This deliverable is a consolidated and extended version of D2.2 [AIDOART-D2.2] considering the feedback by the end-users, integrating the latest features, and proposing a mega model as a comprehensive data modelling managing tool for the AIDOaRt use cases. This deliverable provides the final version of the AIDOaRt Data Engineering Tool Set..
Status:	Type: Other, Dissemination Level: PU

DOCUMENT REVISION LOG

VERSION	REVISION	DATE	DESCRIPTION	AUTHOR
V1	01	11/07/2023	First Document automated Generation from the modelling repository of the project.	Bilal Said (SOFT)
	02	11/07/2023	Completed with Section 7	Sergio Morales (UOC)
	03	14/07/2023	Complete updated document ready for internal review	Julio Medina (UCAN)
	03.5	24/07/2023	Feedback from reviewers	Bilal Said (SOFT) Anna Reale (DT) Mattia Modugno (ROTECH)
	04	27/07/2023	Considering of the reviewers' comments and adding conclusions	Julio Medina (UCAN) & Sergio Morales (UOC)
	05	28/07/2023	Solving Cross-references in the word version	Julio Medina (UCAN)
	06	28/07/2023	Completing mappings	Julio Medina (UCAN) & Sergio Morales (UOC)

Executive Summary

This deliverable proposes a mega model as a comprehensive data modelling managing tool for the AIDOaRt use cases. Also, it refines the intermediate version of deliverable D2.2 [AIDOART-D2.2] considering the feedback provided by the end-users and integrating the latest features of the tools involved in the implementation of the use cases, then offering a consolidated and extended version of it.

This deliverable is the final version of the AIDOaRt Data Engineering Tool Set developed within the Tasks T2.1 (Design time and Runtime data collection), T2.2 (AIDOaRT data internal representation), and T2.3 (Data cleaning, analysis, and management). It is a refinement of the Data Engineering Tool Set built within the AIDOaRt architecture. It updates the mapping to the interfaces of the generic components defined in the Data Engineering Tool Set to (1) specific methods and tools provided by solution providers, and (2) use case requirements of Use Case providers that can potentially be satisfied through those components. Furthermore, it defines a generic data representation and maps data requirements to corresponding data representations. It provides updates regarding tool components delivered until M28. This deliverable also contains an update on use cases with a special focus on their data requirements and how they link to solutions.

It also provides the status of the different solutions for the AIDOaRt Core Tool Set components. Finally, it defines a mega-model to allow providing a global data representation that can be accessed and reused in various stages of the AIDOaRt process. This is an abstracted and elaborated summary of the data models used in the use cases with the aim of providing a common, agreed-upon, global data representation to serve as the foundation for MDE-based activities throughout the AIDOaRt framework.

Table of Contents

Key Terminology Abbreviations

Abbreviations	Terminology
AI	Artificial Intelligence
AIOps	AI Operations
CPS	Cyber-Physical Systems
CPSoS	Cyber-Physical Systems of Systems
DevOps	Development Operations
MBE	Model-Based Engineering
MBRE	Model-based Requirements Engineering
MDE	Model-Driven Engineering
ML	Machine Learning
RTOS	Real Time Operating System
SE	Systems and Software Engineering
SLR	Systematic Literature Review
SMS	Systematic Mapping Study
SysML	Systems Modeling Language
UC	Use Case
UML	Unified Modelling Language
WP	Work Package

Partners Names Acronyms

In the following table, we list the partners' acronyms and full names. The short acronyms are used throughout the deliverable text to identify the partner providing a particular case study requirement or data requirement, or providing a given solution.

Partner Name Acronym	Full Partner Name
ABI	Abinsula SRL
ABO	Åbo Akademi
ACO	ACORDE Technologies S.A.
AIT	AIT Austrian Institute of Technology GmbH
AND	Anders Innovations Oy
AST	Automated Software Testing GmbH
AVL	AVL List GmbH
BT	Bombardier Transportation
CAMEA	CAMEA, spol. s r.o.
CSY	CLEARSY SAS
DT	Dynatrace Austria GmbH
HIB	HI Iberia Ingeniería y Proyectos S.L.
IMTA	Institut Mines-Telecom Atlantique Bretagne-Pays de la Loire
INT	Intecs Solutions S.p.A.
ITI	Instituto Tecnológico de Informática
JKU	Johannes Kepler University Linz
MDH	Maelardalens Hoegskola (Coordinator)
PRO	Prodevelop SL
QEN	Qentinel Oy
RISE	RISE Research Institutes of Sweden
ROTECH	Ro Technology srl
SOFT	Softeam
TEK	Tekne SRL
TUG	Technische Universitaet Graz
UCAN	Universidad de Cantabria
UNISS	Università degli Studi di Sassari
UNIVAQ	Università degli Studi dell'Aquila
UOC	Fundació per a la Universitat Oberta de Catalunya
VCE	Volvo Construction Equipment AB
WESTMO or WMO	Westermo Network Technologies AB

1 Introduction

Besides the description of the content of this deliverable already expressed in the executive summary, it may be convenient to indicate that this is the last deliverable planned to report labour on work package 2.

Among the content in this deliverable the most novel kind of data is the formalisation of the individual data models of the use cases, as well as the mega-model in UML, more precisely in the modelling environment of the Modelio tool. This allows providing a global data representation that can be accessed and reused in any stage of the AIDOaRt process with easy automation. Hence, serving as a foundation for MDE-based activities throughout the AIDOaRt framework.

Since, according to the initial description of work, by this month of the project labour in WP2 is to be finished, further exploitation, usage, and eventual improvements of the mega-model will be reported in future deliverables of other work packages, mainly those of the integration work package (WP5).

Observe that the data models of the use cases prepared by the use case providers as well as the mega-model of the project, even though they are fully described here in this document, they become useful in real practice by using their automatable XMI or other equivalent model serialisations. This model will remain available to all partners in the corresponding Modelio SAAS AIDOaRt repository.

The document is structured as follows.

Section 2 shows the development status and prospective roadmap for some solutions related to the AIDOaRt Data Engineering Tool Set and the concrete capabilities they provide.

Section 3 presents the list of solution components realising the functional specification of the Data Engineering Tool Set components.

Section 4 presents an update in the mapping of the use case requirements and data requirements to the Data Engineering Tool Set components.

Section 5 presents some concrete applications of AIDOaRt Data Engineering Tool Set Solutions in specific Use Cases that have requested the facing of stated specific challenges. These challenges were proposed and discussed in the context of hackathons realised in various plenary meetings and can be found in deliverable D5.3 [AIDOART-D5.3]

Section 6 elaborates on the mega model on top of the UML representation of all the data models from all AIDOaRt use cases.

Section 7 sketches some concluding remarks and links to other documentation.

Finally, Section 8 includes some bibliographic references made along the text.

2 Progress status of the AIDOaRt Data Engineering Tool Set

This section gives an overview of the implementation achievements and development status of the AIDOaRt Data Engineering Tool Set solutions with regards to the milestones of the AIDOaRt project, as stated in deliverables D2.1 and D2.2. This section briefly describes the current status of the solutions and tools.

In this section, we include solutions having features that are released since M21 and up to the end of the project. Solution providers present:

* in a first subsection entitled "Capabilities Implementation Status", the features released from M21 to M28 (i.e., the submission date of the current deliverable D2.3), covering features with release dates declared at MS5 (M24) and MS6 (M28) of the project;

* in a second subsection entitled "Future Capabilities Implementation Roadmap", the features to be released from M29 up to M36, i.e., after the submission date of the current deliverable D2.3 and up to the end of the project, covering features with release dates declared at MS7 (M32) to MS8 (M36).

To briefly summarise the reported status, it should be noted that:

- 10 tools are related to the AIDOaRt Data Engineering Tool Set,

23 tool purposes are related to the solutions participating in core issues:

- 1 concerns the release at the Intermediate milestone (M20)
- 8 concern the release at this deliverable (M21 up to M28)
- 14 concern the release at the final milestone (M29 up to M36)

Considering the three main axes of Data Engineering in our AIDOaRt Framework:

- 6 implement or can be considered as Data Collection components
- 7 implement or can be considered as Data Management components
- 2 implement or can be considered as Data Representation components.

2.1 Solution - ESDE (ACO))

This solution implements the following Data Engineering Tool Set components:

- Data Collection
- Data Management

2.1.1 News and Updates

As explained in section 3.1 of D2.2, a main purpose of ACORDE in AIDOaRt is to extend the ESL embedded Software Development Environment (ESDE) to enable a "deep validation" of embedded software (eSW) on top of a virtual platform model (virtual platform-in-the-loop or VIL). "Deep validation" here means that, as well as a functional validation, performance analysis and even anomalies detection is supported, by exploiting a holistic set of logs/traces that reflect key information on the system status. Specifically, in our approach, that means to extract traces from several levels of the system (multi-level tracing), which can encompass different software layers (application, RTOS, drivers) and hardware components (e.g., processor instructions, memory bank accesses, etc). Therefore, a critical pre-requisite is a proper data collection and management which enables such multi-level tracing and its exploitation.

Figure 1 focuses on the collection and management of data from the virtual embedded system model in the ESDE framework. It summarises the progress of such data collection and management. An orange-coloured box has been used to explicitly show relation to the AIDOaRt architecture.

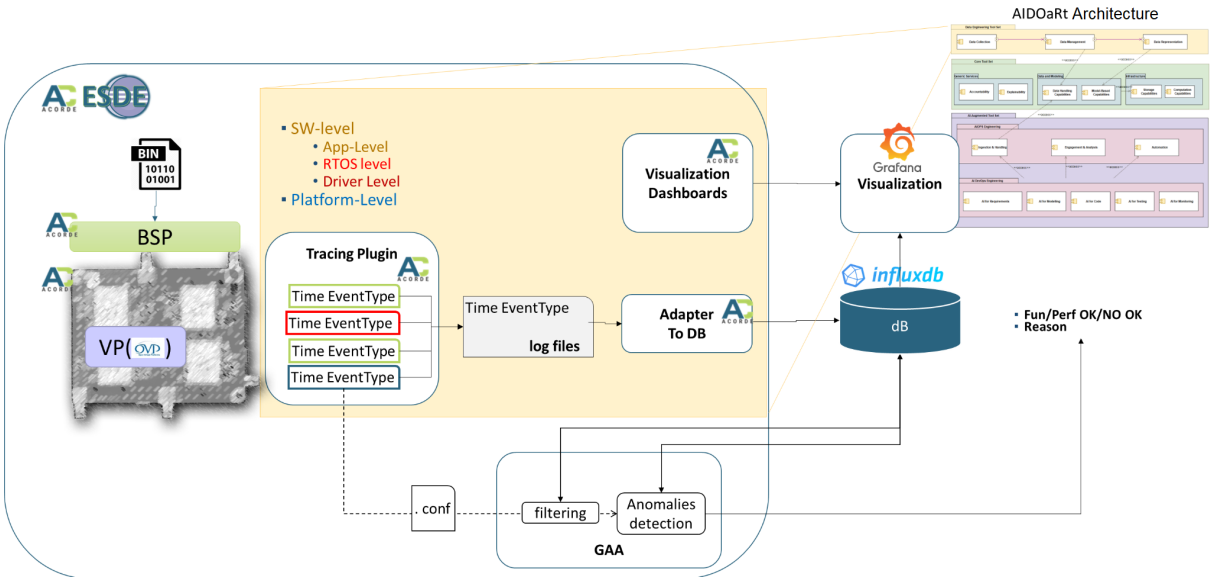

Figure 1 Data collection and management in the virtual embedded system

Specifically, ACORDE has worked in order to enable, a **data collection** of events from different levels of the virtual model of the embedded system, and to facilitate flexible and powerful **data management**, via enabling efficient storage and retrieving of such data by collecting it on time -series suited, cutting edge data services, i.e., by relying on *Influxdb*. Moreover, as shown in the figure, the developed approach provides support for powerful data representation based on *Grafana*.

All this is sustained on several ACORDE developments in relation to AIDOaRt WP2 aspects and activities:

- A **tracing plugin** that provides the capability to set up fundamental SW and HW level event monitors (e.g., function entry/exit, write access to the memory region, etc.). This plugin has a generic part and specific parts (dependent on the compiler and VP framework, i.e., *GCC* and

OVP in this case). This plugin generates text-based traces, either as a single or separated CSV files with the monitored/traced events.

- An **adapter to the database** in charge of adapting and inserting the aforementioned traces into the time-series specific format of the database.
- **Specific dashboarding** for the multi-level analysis of embedded systems.

Figure 2 provides a more specific example. It shows the trace produced after running an application with 4 nesting levels (a *main* function that invokes 16 times a function called *foo*, which in turn invokes 16 times the *bar* function, which in turn calls 16 times the function *thud*).

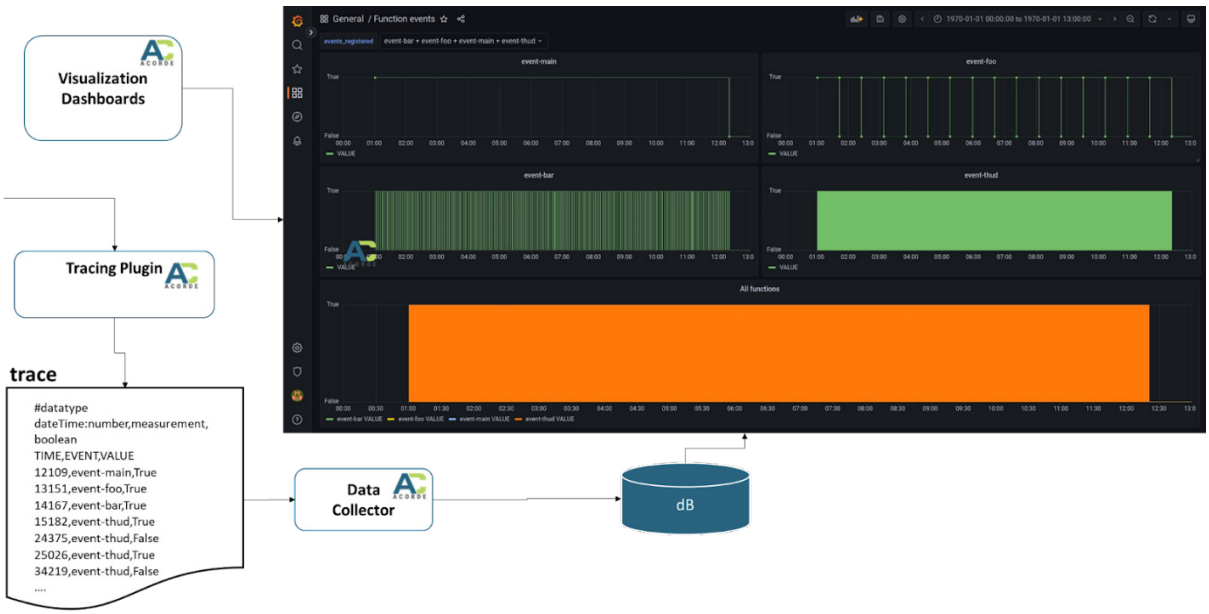

Figure 2 Trace produced after running an application with 4 nesting levels

At the left bottom side, the beginning of the trace file is shown, with a format header and then all the related events at its cycle-accurate time in order of occurrence. This single-file compact format is convenient for most of the cases. The example illustrates a multi-trace yet at a single level, i.e. application software level. However, as mentioned, the trace can contain events related to different levels of the system. For instance, *bar* could be an RTOS function, while *thud* a driver-level function.

Once this capability for multi-level tracing and database management is ready, all the components required for automated or semi-automated bug detection are in place, as an initial version of the generic anomalies analysis (GAA) has already been developed (in the context of WP4) and its evaluation started in the context of port data (WP5 context).

Therefore, the capability of automatic (or semi-automatic) anomaly detection from the embedded traces is partially implemented, but not completed yet. As ACORDE has made explicit in different forums (telcos, meetings) of the project, ACORDE is executing the “Positioning Monitoring Solution for Industrial Environment” solution with higher priority than the “ESDE extension” solution. The main reasons are that the former solution has more immediate exploitation chances and that it has required hardware design and development. Furthermore, at the same time, it has also meant the development of the GAA, which, as mentioned, is a main component of the ESDE extension tackled in AIDOaRt. At

the time of this report writing, GAA is still being evaluated and polished at the distributed domain for port-sensed data. Once this is well consolidated, a very interesting aspect that ACORDE aims to evaluate is the transversality of the GAA technology, i.e., that it can be effectively applied to the embedded domain. Therefore, the aforementioned priorities and the need to cover all the necessary steps involve the revision of the estimated delivery of the automated analysis of embedded traces, now estimated by M36.

2.1.2 Capabilities Implementation Status

Capability Name & Description	Implementation Status	Comments / Release notes
Multi-level functional and performance logs and traces: Framework able to provide time series of performance metrics at different levels of abstraction and at different implementation levels (platform HW, platform SW, application SW)	**Implementation Level:** Implemented **Estimated Delivery Date:** MS4 (M20) **Licence:** Proprietary – ACORDE. It refers to specific extensions performed on ESDE to obtain functional and performance logs and traces developed by ACORDE and considered key for protecting competitive advantages achieved thanks to AIDOaRt. See note (*) below.	Tracing techniques implemented, with compiler (GCC) and VP framework (Imperas) specificities. Work on WP2 enabled fundamental facilities for multi-level SW and HW level tracing in place. WP2 work also enabled automated insertion of collected data on database and facilities for multi-level traces visualisation.
Automated (or semi-automated) bug detection/prediction: Ability to learn and detect or predict possible functional or performance bugs based on performance traces. Two possibilities envisioned. First one, based on offline trace analysis, can handle non-causal analysis for better anomalies/errors detection. This feature will support product fixes and development smoothly integrated in a DevOps environment. A second possibility is to simulate the system equipped with monitoring probes and a ML engine capable to use the collected metrics for an on-the-fly (real-time, or at least causal) computation for detecting/predicting anomalies.	**Implementation Level:** Partially Implemented **Estimated Delivery Date:** MS5 (M36) **Licence:** Proprietary – ACORDE. It refers to specific extensions performed for automated or semi-automated bug detection and/or prediction on the embedded domain. See note (*) below.	WP2 work enabled us to put in place all the fundamental components, like former prototypes of multi-level tracing and database managing, and a former version of the generic anomalies analysis module. It is complemented with further database and visualisation infrastructure. Further evaluation of this framework is ongoing to consider its suitability for the embedded domain. The delivery is replanned by M36 so that use case-related data can be considered too

Table 2 Capabilities Implementation Status of the ESDE Solution

(*) It excludes any open-source and/or third-party source used and any open-source extension that needs to be published. In addition, ACORDE may release fixes and extensions which do not compromise ACORDE's competitive advantages granted by AIDOaRt as open source.

2.2 Solution - Position Monitoring for Industrial Environment (ACO))

This solution implements the following Data Engineering Tool Set components:

- Data Collection
- Data Management

2.2.1 News and Updates

In section 3.2.1 of D2.2, an overview of the industrial monitoring solution developed by ACORDE, with specific support for position monitoring was provided. This section reports the main advances in the data collection and management of the infrastructure developed by ACORDE. Figure 3 sketches some of those main advances so far.

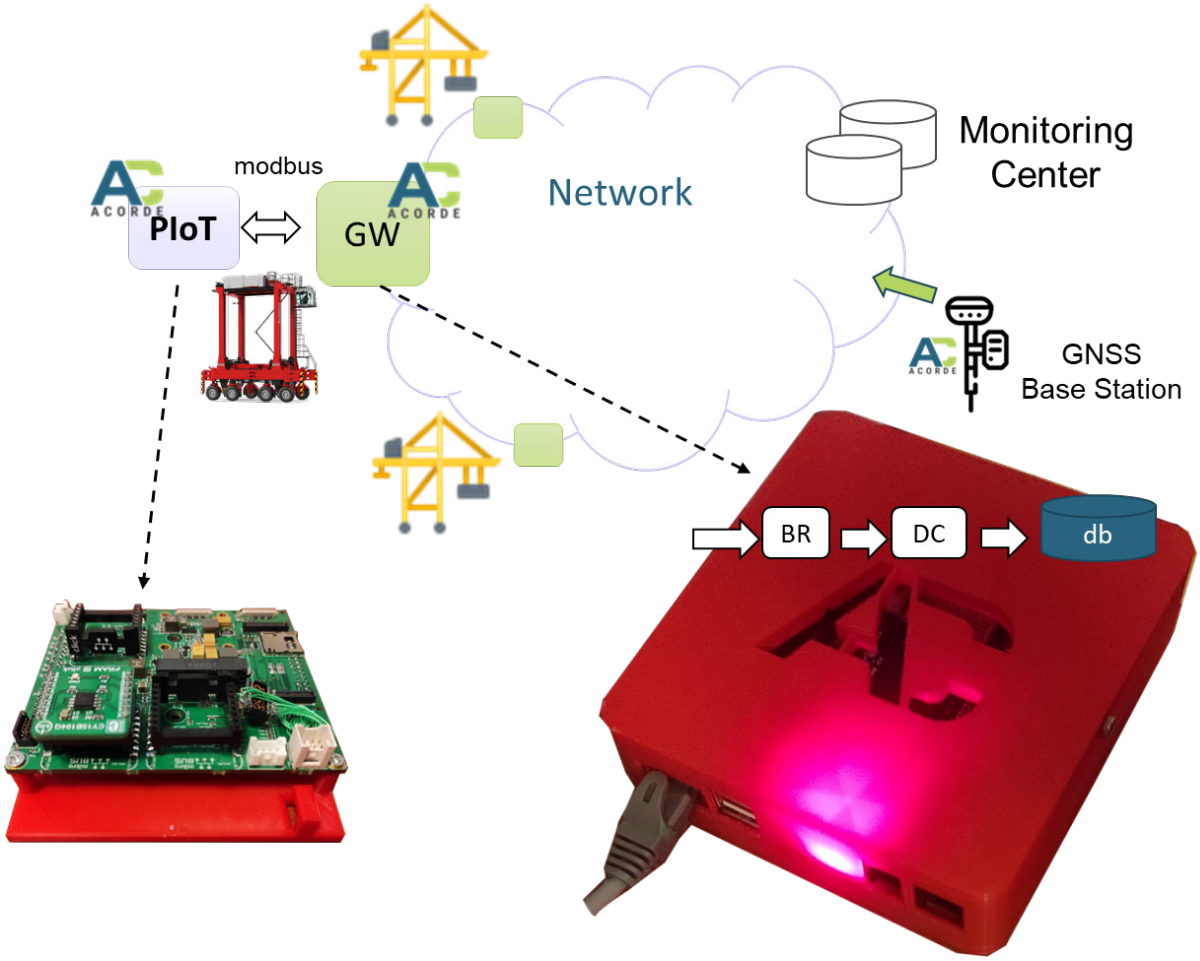

Figure 3 Main advances in the data collection and management of the infrastructure developed by ACORDE

ACORDE has completed the development of a first version of Gateway (GW on the figure) based on a development platform relying on an ARM53 quad-core architecture. The bottom right-hand part of the figure shows that implementation with its envelope as it was used and shown in past May 11th at the demonstrators' session in the plenary meeting of Vasteras. Moreover, in that meeting, ACORDE also showed a second version of the Gateway based on a custom hardware design, which will bring certain advantages, such as supporting wireless interfaces, higher storage capability and industrial temperature ranges. The base of that ACORDE design is a modular platform, shown in the bottom left-hand side of the figure. This modular platform is being employed also on the positioning sensor (PIoT in the figure). The hardware design of the PIoT has been completed. The GNSS receivers have been selected and integrated into the platform. Here, a main update is that the design of the positioning sensing is no longer supported by an additional network of UWB beacons. Further discussion with Prodevelop, the Smart Port Monitoring UC provider, enabled further polishing of the requirements and their priority. It was specified that the PIoT+Gateway devices should be mounted on a *straddle*

carrier crane, shown in the figure), with higher mobility than an STS crane. This meant a more demanding position update rate (10 Hz) while keeping a demanding accuracy (<1m). This performance requirements update involved a re-design of the positioning sensing solution (technology and architecture). The PIoT design relies now on a fusion of local sensors (INS) and RTK receiver, where the support from UWB beacons (as posed in the initial solution) lose focus. On the other hand, the renewed architecture now includes a GNSS base station visible in the port's local network. An interesting aspect is that the renewed solution still benefits from the continuum computing architecture of the solution.

The figure also illustrates the data collection and management services that have been put in place in the Gateway. These are a brokering service (BR) to retrieve data from the ethernet interface; a data collection/adaptation (DC) service in charge of retrieving data from the broker, and other type of interfaces that are planned to be added to the Gateway (e.g. for low-power wireless access to other sensors) and insert them into the database; and finally, a time series database (*Influxdb*). The proper run of these services on the Gateway (together with other analysis services) was also shown in the 4th hackathon demonstration session, hosted in our plenary meeting of May 11th 2023 in Västeras.

As well as the advances in the aforementioned data collection and management IoT+Edge infrastructure, ACORDE has also advanced in stating the data model in key interfaces. One of those interfaces is how the Gateway offers positioning data to the monitoring infrastructure set up by Prodevelop. This data model states both, the positioning data and quality indicators, and has been formalised in the shape of a UML class model that has been integrated into the Modelio data model of the Smart Port Monitoring Platform (SPMP). It is shown in Figure 4, where the new elements added to the SPMP Data model are bounded by a dashed box.

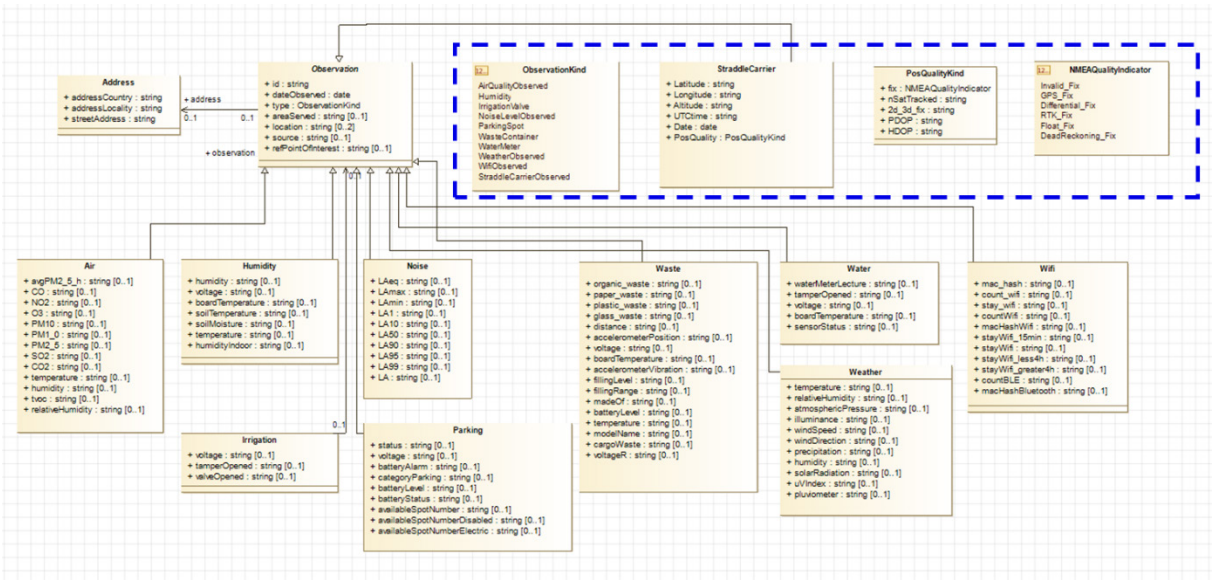

Figure 4 Model of Position data and its quality as integrated on the SPMP data model

Finally, ACORDE has also made an initial effort to formalise the configuration, input and output interface data of its Generic Anomalies Analysis (GAA) component. Such a model in its current state is represented in Figure 5 and has been initially shown to UC partners with modelling expertise (UoC).

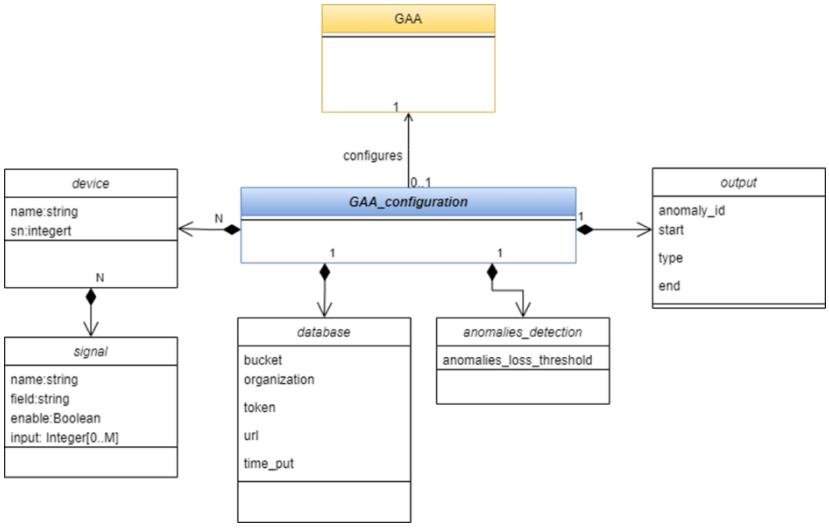

Figure 5 Model of Generic Anomalies Analysis configuration data

2.2.2 Capabilities Implementation Status

Capability Name & Description	Implementation Status	Comments / Release notes
Monitoring System Design and Development: Integration of monitoring infrastructure based on docker containers and open software solutions (Grafana, Prometheus, InfluxDB, Zabbix…)	**Implementation Level:** Partially Implemented **Estimated Delivery Date:** MS4 (M32) **Licence**: Proprietary – ACORDE. It refers to a specific monitoring solution on the Cloud-Edge-IoT architecture developed by ACORDE and considered key for protecting competitive advantages achieved thanks to AIDOaRt. Note (*) of section 1.1.2 applies	ACORDE has i) consolidated and formalised a data model on the positioning data to transfer to the monitoring system ii) completed and demonstrated the HW/SW gateway design, and complemented the design of the PIoT (whose SW design evaluation is on-going) iii) run and showed the possibility to run data collection and management, and even analysis services within the Gateway. The PIoT infrastructure will be completed and used to generate real data and complete the evaluation of the generated infrastructure in the frame of WP5. This means that the complete delivery of this infrastructure is expected a bit later than initially expected (M24), i.e. by M32.
AI/ML based analysis of monitored data: ACORDE aims at enabling an automated AI/ML based processing of monitored data from the Monitoring System which enables	**Implementation Level:** Implemented **Estimated Delivery Date:** MS5 (M24) **Licence:** Proprietary – ACORDE. It refers to specific AI/ML analysis methods applied on the data retrieved from the	As it was shown in the demonstrator session of Vasteras Plenary meeting May 11th, 2023, ACORDE has an initial implementation of Generic Anomalies Analysis (trained for a specific data), running on the gateway. The possibility to perform these types of analyses and run at the edge of the continuum computing architecture of the monitoring

detection (and eventually prediction) of anomalies.	monitoring solution on the Cloud-Edge-IoT architecture developed by ACORDE and considered key for protecting competitive advantages achieved thanks to AIDOaRt. Note (*) of section 1.1.2 applies.	infrastructure is ready. Yet, ACORDE will go on evaluating and polishing this result in the remaining efforts related to WP4 and WP5.

Table 3 Capabilities Implementation Status of the Position Monitoring for Industrial Environment Solution

2.3 Solution - Kolga (AND)

This solution implements the following Data Engineering Tool Set components:

- Data Collection

2.3.1 News and Updates

Initial demo application deployed and solution framework with all the components exists.

2.3.2 Future Capabilities Implementation Roadmap

Capability Name & Description	Implementation Status	Roadmap and Planning
Kólga: Tool for creating CI/CD pipelines using GitLab/GitHub/Azure that can build, test and deliver applications to Kubernetes clusters. Highly flexible solutions can be created on top of the tool, such as AI testing all using the tools such as Github Actions that could already be in use. An example would be to bring a web service from just source code to running in a scalable production environment in a very short (minutes) time. Our only requirements are that the application is based on a container technology such as Docker and that the final running environment is Kubernetes. In other words, Kólga can also enable running your application in the edge for instance.	**Implementation Level:** Partially Implemented **Estimated Delivery Date:** MS7 (M32) **Licence:** MIT	Robusting the codebase from POC level to MVP level. Production-ready deployment and continuous training/deployment of the app

Table 4 Future Capabilities Implementation Roadmap of the Kolga Solution

2.4 Solution - devmate (AST)

This solution implements the following Data Engineering Tool Set components:

- Data Collection
- Data Management

2.4.1 News and Updates

We are currently working on an improvement of the devmate Equivalence Class Prediction System. Currently it is focused on storing data for each user individually. We defined a concept for a centralised data store which makes it possible to share defined test data among several users of devmate.

2.4.2 Capabilities Implementation Status

Capability Name & Description	Implementation Status	Comments / Release notes
Code Parser: Parse input-/output-parameters from existing code or unit-/system-models (eg. XML / DSL specified models)	**Implementation Level:** Partially Implemented **Estimated Delivery Date:** MS5 (M24) **Licence:** Proprietary licence (Automated Software Testing GmbH)	We have implemented code parsers for C# and Java. A C parser is in development and available for testing. The code parser produces an intermediate test model for use in devmate.
Testcase Generator: Generating a set of testcases based on rules (data-driven, combination of input parameters / data)	**Implementation Level:** Partially Implemented **Estimated Delivery Date:** MS6 (M28) **Licence:** Proprietary licence (Automated Software Testing GmbH)	Test-Case generator is implemented at this point and marked for a future review

Table 5 Capabilities Implementation Status of the devmate Solution

2.4.3 Future Capabilities Implementation Roadmap

Capability Name & Description	Implementation Status	Roadmap and Planning
Testmodel Editor: Automatically combining and generating testcases for system/unit under test (primarily blackbox validation test techniques)	**Implementation Level:** Partially Implemented **Estimated Delivery Date:** MS7 (M32) **Licence:** Proprietary licence (Automated Software Testing GmbH)	Testmodel Editor has had a major update and redesign in the newest version. The editor directly manipulates the internal test model
Testcase Evaluation: AI/ML augmented evaluation and reduction of testcases (based on	**Implementation Level:** Not Implemented	Future development efforts will revisit metrics and code coverage while current efforts are focused

an extendable set of metrics e.g. coverage, mutation score, test runtime)	**Estimated Delivery Date:** MS7 (M32) **Licence:** Proprietary licence (Automated Software Testing GmbH)	on language support and usability
Testdata prediction system: AI/ML supported testdata specification (testdata management, similarity measures, testdata prediction system)	**Implementation Level:** Not Implemented **Estimated Delivery Date:** MS8 (M36) **Licence:** Proprietary licence (Automated Software Testing GmbH)	Research into similarity and prediction systems are currently on halt.
Model Parser: Parse input-/output-parameters from existing unit-/system-models (eg. XML / DSL specified models)	**Implementation Level:** Not Implemented **Estimated Delivery Date:** MS8 (M36) **Licence:** Proprietary licence (Automated Software Testing GmbH)	Models can be parsed through Code Parser and support will be added on demand.

Table 6 Future Capabilities Implementation Roadmap of the devmate Solution

2.5 Solution - Keptn (DT)

This solution implements the following Data Engineering Tool Set components:

- Data Management

2.5.1 News and Updates

The Keptn tool v 1.4.0 has been released, including consistent changes to improve the tool usability in the Data Management for CSP use cases. Mainly: Event specs for rollback/remediation scenarios have been finalised, the user interface for quality gates now shows a breakdown of SLIs for easy data visualisation, and the storage solution using git has been made faster and more reliable. Furthermore, the usage of a git repo to keep track of the sequences and run data is now mandatory.

2.5.2 Capabilities Implementation Status

Capability Name & Description	Implementation Status	Comments / Release notes
Keptn CloudEvents: The event-based approach is built upon a well-defined defined set of Keptn events; currently in v0.2.2. All Keptn events conform to the CloudEvents specification in version v1.0. The CloudEvents	**Implementation Level:** Partially Implemented **Estimated Delivery Date:** MS6 (M28) **Licence:** Keptn is made available under the	Version 0.2.4 of Keptn Spec has been released. This showcases progress in the event data definition since now the `Rollback Triggered` remediation event is defined. We are currently designing

specification is a vendor-neutral specification for defining the format of event data. In the course of the Keptn project, the event data is defined for the use-cases of application delivery and remediation as well as life-cycle orchestration tasks such as deployment, test, evaluation, release, problem, etc. The specification of Keptn CloudEvents is not limited to the mentioned tasks and can be easily extended by following the proposed format. The Keptn project is currently in the progress of aligning the Keptn events with the event specification from the Continuous Delivery Foundation (CDF) with the goal of establishing an industry-wide eventing standard for application life-cycle orchestration.	terms of the Apache 2.0 licence. Any use of this Background by any Participant is subject to and must conform to Apache 2.0 (https://github.com/keptn/keptn/blob/master/LICENSE).	example sequences, to ensure that our current event specification fits all scenarios of CSP projects simulations/development
Keptn Control-Plane: **Keptn is built for Kubernetes and consists of a couple of Keptn core services that altogether form the Keptn control-plane. The control-plane is responsible for orchestrating the life-cycle of an application managed by Keptn. Execution-plane service can connect to the control-plane to interact with Keptn via CloudEvents sent through NATS. The CloudEvents are currently stored in a MongoDB that serves as the datastore for all events that are sent via Keptn and allows for full traceability of life-cycle events. The architecture of the Keptn project can be found in the Keptn documentation. Keptn's architecture allows any tool to be integrated into the application life-cycle orchestration managed by Keptn. These execution plane services can run within the same cluster as the Keptn control plane**	**Implementation Level:** Partially Implemented **Estimated Delivery Date:** MS6 (M28) **Licence:** Keptn is made available under the terms of the Apache 2.0 licence. Any use of this Background by any Participant is subject to and must conform to Apache 2.0 (https://github.com/keptn/keptn/blob/master/LICENSE).	A major work has been done to improve Keptn control plane: All base services are now at v1.4.0. We released a new service in charge of handling the internal status of the project using git: the resource-service. This component now substitutes the previous configuration service. The new service always requires a Git upstream to be configured for a Keptn project. Even though the git approach is working well for software projects we need to consider that CSP and HW simulations require larger amounts of shared files and resources. Typically in the industry git is not considered sufficient for these applications so we plan to evaluate whether ulterior changes to the service will be required. Nevertheless, the new service brings many advantages, such as faster response times and the possibility to upgrade Keptn

or on different clusters, allowing to orchestrate multi-cluster deployments, tests, evaluations, and operational tasks such as remediation orchestration or ChatOps.		without any downtime. Furthermore, we introduced zero downtime in the control plane to reduce losses of events and sequences' information
Keptn Quality Gates: **A central component of Keptn are quality gate evaluations based on service-level objectives (SLOs). Therefore, Keptn builds upon SRE best practices such as service-level indicators (SLIs) and allows to declaratively define SLOs for them. These SLOs define quality criteria for the applications and act as a gatekeeper during software delivery before promoting any application or microservice from one environment (e.g,. hardening) to the next environment (e.g., production).**	**Implementation Level:** Partially Implemented **Estimated Delivery Date:** MS6 (M28) **Licence:** Keptn is made available under the terms of the Apache 2.0 licence. Any use of this Background by any Participant is subject to and must conform to Apache 2.0 (https://github.com/keptn/keptn/blob/master/LICENCE).	Lighthouse, the component in charge of quality gates, is now at version 1.4.0. Keptn Bridge (our UI) is now leveraging a new rendering library that offers more flexibility for displaying the SLI breakdown. The component has been made more robust in case of malformed SLI or SLO. Specific example configurations for the CSP use case are work in progress.

Table 7 Capabilities Implementation Status of the Keptn Solution

2.6 Solution - EMF Views (IMTA)

This solution implements the following Data Engineering Tool Set components:

- Data Representation

2.6.1 News and Updates

As mentioned in previous deliverable D2.2, the Viewpoint Builder and View Builder (i.e., the two EMF Views core components) already provide a basic support for Verification and Validation (V&V) based on metamodel conformance, and the associated VPDL language also comes with base syntactic validation. At this stage, we did not yet have the need for more advanced V&V capabilities, but this could be envisioned during the third and last year of the project if required in the context of a use case, for instance, to facilitate the verification and/or validation of some CPS-related properties. Concerning the scalability and efficiency of model view computation, navigation and querying, several identified bugs have been fixed during this second year of the project. For example, we made the EMF Views solution more reliable when considering UML/SysML models within the specified views, as required notably in the CPS context of the VCE partner (cf. the corresponding subsection/table hereafter). Moreover, as explained in the previous deliverable D2.2, the generated views are partially editable by default (for basic attribute modification) but the currently provided view update capabilities remain

limited. During this second year of the project, we started to work on integrating the use of Machine Learning techniques with EMF Views in order to improve the semi-automated computation and/or update of some elements of the views. We plan to achieve more complete and publishable results during the third and last year of the project (cf. the corresponding subsection/table hereafter). This could be useful in practice in a CPS model-based engineering context, for example, to recommend eventually missing inter-data model relations that the system engineers have not or cannot foresee.

2.6.2 Capabilities Implementation Status

Capability Name & Description	Implementation Status	Comments / Release notes
Model Viewpoint and View computation: Be able to compute a given view in a scalable way, based on a previously specified viewpoint and a corresponding set of metamodels and models.	**Implementation Level:** Partially Implemented **Estimated Delivery Date:** MS6 (M28) **Licence:** EPL 2.0 / GPL 3.0	As introduced earlier, this tool capability is made available in the current version of EMF Views. Updates have been performed (as some bugs have been detected on given view operations) in order to better support the building of viewpoints and views over particular types of models (e.g., UML/SysML ones). For the last phase of the project, we envision extending/refining EMF Views and the associated VPDL language with some AI-related support (cf. the next section/table)
Model Viewpoint and View navigation and query: Be able to efficiently navigate and query an already computed view.	**Implementation Level:** Partially Implemented **Estimated Delivery Date:** MS5 (M24) **Licence:** EPL 2.0 / GPL 3.0	As introduced earlier, this tool capability is made available in the current version of EMF Views. Updates have been performed (as some bugs have been detected on given view operations) in order to better support the use of viewpoints and views over particular types of models (e.g., UML/SysML ones). For the last phase of the project, we envision extending/refining EMF Views and the associated VPDL language with some AI-related support (cf. the next section/table)

Table 8 Capabilities Implementation Status of the EMF Views Solution

2.6.3 Future Capabilities Implementation Roadmap

Capability Name & Description	Implementation Status	Roadmap and Planning

Model Viewpoint and View update: Be able to dynamically and efficiently update an already computed view.	**Implementation Level:** Not Implemented **Estimated Delivery Date:** MS7 (M32) **Licence:** EPL 2.0 / GPL 3.0	As introduced earlier, this tool capability is not yet available in the current version of EMF Views. The work started on integrating the use of Machine Learning techniques with EMF Views in order to improve the semi-automated computation and/or update of some elements of the views. This work will be completed and hopefully published during the third and last year of the project.

Table 9 Future Capabilities Implementation Roadmap of the EMF Views Solution

2.7 Solution - a2k-runman (ITI)

This solution implements the following Data Engineering Tool Set components:

- Data Collection
- Data Management

2.7.1 News and Updates

During the 4'th Hackathon we mainly concentrated on the interchange and management of data between the partners in the smart port use case. We defined and implemented the following data handling services:

Importation of Modelling Data: To analyse and simulate the smart port platform, we need to define a model for the architecture of the smart port platform. This architecture is a very large, distributed, heterogeneous, cyber-physical system consisting of hundreds of IoT devices, and dozens of processing devices consisting of various gateways, intermediate fog processors, and centralised cloud servers. Initially, this proved problematic as A2K only has a graphical interface for model editing and so the definition of the platform became tedious and error-prone. We solved this by developing a new import service which can import model components for basic text-based spreadsheets. This considerably simplified data exchange between the partners. An example of such a spreadsheet is presented in Figure 6.

# id	Type?	Name			
#	Subsystem				
#	Bus		Internal Bus Type		Exported [Name/none]
#	Processor		Processor Type	Num Procs (num)	
#	Memory		Memory Type	Num Mems (num)	
#	Device		DeviceType		
#	Interface		Network Type	Num Ports (num)	
#	component		Subsystem Type		Used Port [Name/none]
#	port		Internal Bus Type	Num Ports (num)	
#	link		Bus Type	Source	Destination
0	Subsystem	Server			
1	Bus	Main bus	Intel Bus		none
2	Processor	Main Processor	Intel i7-9700K @ 3.60GHz (x86_64)	8	
3	Memory	Main Memory	SDRAM DDR4 8G	2	
4	Bus	PCI Express	PCI Express		none
5	Interface	Ethernet	Gigabit Ethernet	1	
0	Subsystem	Firewall			
1	Bus	Main bus	Intel Bus		none
2	Processor	Main Processor	Intel i7-9700K @ 3.60GHz (x86_64)	8	
3	Memory	Main Memory	SDRAM DDR4 8G	2	
4	Bus	PCI Express	PCI Express		none
5	Interface	Ethernet	Gigabit Ethernet	2	
0	Subsystem	Switch 5p			
1	Bus	Internal bus	AMBA AHB		none
2	Processor	Network processor	ARM Cortex-A53 @ 3.2 GHz (armv8-a)	2	
3	Interface	Ethernet	Gigabit Ethernet	5	
0	Subsystem	Router 5p			
1	Bus	Internal bus	AMBA AHB		none
2	Processor	Network processor	ARM Cortex-A53 @ 3.2 GHz (armv8-a)	2	
3	Interface	ethernet	Gigabit Ethernet	5	
4	Interface	wifi	WiFi 802.11ac	1	

Figure 6 Spreadsheet Showing Definition of Smart Port Hardware Architecture

Importation of Application Architecture and Message Protocols: Similarly, we have defined data interchange spreadsheets to enable the definition of the software applications, parameters, and communications protocols running on the various processors on the smart port architecture. This allows us to represent the flow of messages and software activities within the system. An example is shown in Figure 7.

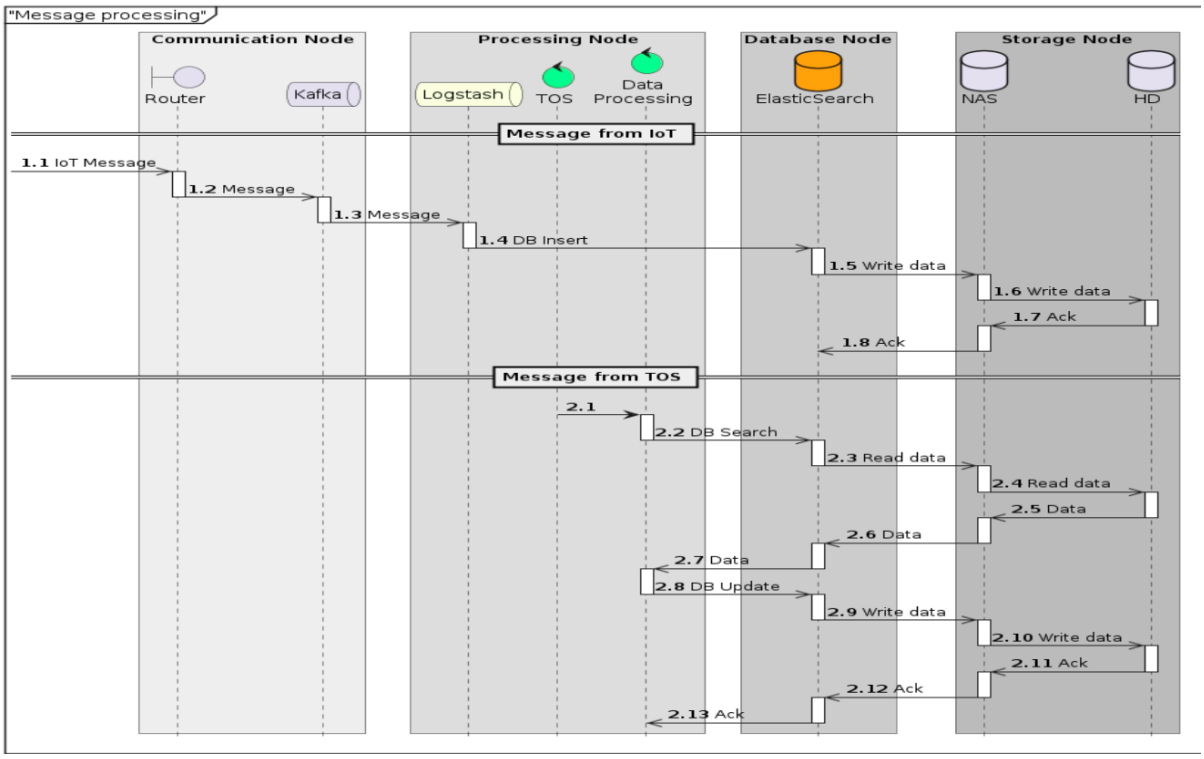

Figure 7 Example Sequence Diagram of Smart Port Processing

Definition of Smart Port Activities: We are also interested in simulating the day-to-day activities of the smart port. To this end, we need to know the expected arrival times of ships, the use of loading cranes and vehicles, and so forth. These are used in the A2K simulator to generate simulated scenarios and to analyse the expected system performance under different conditions. Again, we have defined some spreadsheet templates to implement this data exchange. An example of such a sheet is shown in Figure 8.

Vessel arrival times and port resources used

Las filas con la primera celda en blanco se ignoran.
Las que comienzan con # son leyendas: NO MODIFICAR
Para añadir más elementos, simplemente insertas más filas

Fields	Type	Description
Name	string	Vessel name
Arrival	time	Arrival time
Containers	integer	Number of containers
STS	integer	Assigned Ship-to-Shore cranes
RTG	integer	Number of Rubber Tyred Gantry cranes
RS	integer	Number of Reach Stackers
ECH	integer	NUmber of Empty Container Handler
TT	integer	Terminal Tractor

# id	Name	Arrival	Containers	STS	RTG	RS	ECH	TT
1	BT 1	9:30	2000	4	6	10	8	42
2	BT 2	11:00	2600	6	8	15	4	50
3	BT 3	12:30	1800	3	4	8	3	42
4	BT 4	14:00	2200	5	6	12	5	60
# end								

Figure 8 Specification of Port Activities

2.7.2 Capabilities Implementation Status

Capability Name & Description	Implementation Status	Comments / Release notes
a2k/tunning: Automatic criticality level / operational mode change management based on real-time data collection and analysis using adaptive learning algorithms.	**Implementation Level:** Partially Implemented **Estimated Delivery Date:** MS7 (M36) **Licence:** Proprietary ITI	Currently under development. The a2k/tuning component will connect with the a2k/detection component to adjust the CPU clock frequency according to the predicted CPU load. The goal is to minimise power consumption in a dynamic fashion.
a2k/detection: To monitor the operation of a cyber-physical system in real-time to provide warnings and advice when critical situations are observed or predicted.	**Implementation Level:** Partially Implemented **Estimated Delivery Date:** MS7 (M36) **Licence:** Proprietary ITI	The a2k/detection component is currently under development and evaluation. This component will be used to model and predict expected CPU loads. The prediction model is trained and learned using the simulation facilities provided by the a2k/scheduling service of the a2k-modev component.)

Table 10 Capabilities Implementation Status of the a2k-runman Solution

2.7.3 Future Capabilities Implementation Roadmap

Capability Name & Description	Implementation Status	Roadmap and Planning
a2k/tunning: **Automatic criticality level / operational mode change management based on real-time data collection and analysis using adaptive learning algorithms**	**Implementation Level: Partially Implemented** **Estimated Delivery Date: MS7 (M36)** **Licence: Proprietary ITI**	This component adjusts the processor frequency considering the predicted workload demands. We use a real-time simulator and a prediction method based on AI to do this. Currently, we are evaluating different CPU frequency control algorithms and how to interface these with the workload prediction methods described below.
a2k/detection: To monitor the operation of a cyber-physical system in real-time t45o provide warnings and advice when critical situations are observed or predicted.	**Implementation Level:** Partially Implemented **Estimated Delivery Date:** MS7 (M36) **Licence:** Proprietary ITI	We have already developed a simulator platform to test and evaluate a range of different algorithms for predicting the future CPU workload in the smart port use case. We are now investigating a range of different prediction methods. In the very near future, we will link these with the a2k/tuning component mentioned above.

Table 11 Future Capabilities Implementation Roadmap of the a2k-runman Solution

2.8 Solution - ConvHandler (ROTECH)

This solution implements the following Data Engineering Tool Set components:

- Data Management

2.8.1 News and Updates

The ConvHandler is a Python module whose aim is to filter the data which will go as input to the **Onboard Bridger** (described in section 3.9). The data filtering is performed using the Python library Pandas. The first step is to read the input file, provided in .xlsx format, and look for empty measurements and/or outliers, which will then be replaced with a placeholder value.

2.8.2 Capabilities Implementation Status

Capability Name & Description	Implementation Status	Comments / Release notes
Data Converter: This component will elaborate raw data received by various services in order to transform it into exploitable data for end-user purposes.	**Implementation Level:** Not Implemented **Estimated Delivery Date:** MS5 (M24) **Licence:** Licence-free	All the capabilities are fully implemented and we are now integrating the module with the Bridger platform in order to handle large amounts of data

Table 12 Capabilities Implementation Status of the ConvHandler Solution

2.9 Solution - Bridger (ROTECH)

This solution implements the following Data Engineering Tool Set components:

- Data Management

2.9.1 News and Updates

The Bridger is a software platform divided into two parts: Onboard Bridger and Remote Bridger. Its aim is to manage filtered data taken as input from the ConvHandler and reorganise it to be stored inside a relational database. Both have been developed in C++ and can be built on Linux systems via Makefile.

In order to communicate correctly with each other, both parts of the platform need an MQTT connection, where the Onboard is the publisher and the Remote is the subscriber.

The Bridger Onboard published on a range of different topics, each one serving a specific purpose:

"**command**": is used to request the exit of a session, when needed for debugging or redeployment purposes;

"**open**": is used to indicate the start of the transmission of a new file;

"**payload**": this topic is used to publish the content of the current file;

"**close**": this topic is used to signal the end of the current file.

The **Bridger Onboard** takes as input a stream of data and then encrypts it using an AES-based encryption scheme. It then performs an open-close transfer to publish the data for the Remote Bridger with the aforementioned topics.

The **Bridger Remote** connects to the MQTT by subscribing to the topics "command", "open", "payload", and "close", with QoS (Quality of Service) 2, which means it will receive each message exactly once. This guarantees that the Remote Bridger will be able to reconstruct the original message without missing or repeated pieces. When a message is published on these topics by the Onboard Bridger, it is received by the Remote platform and decrypted via the decryption function of the encryption scheme. It is then serialised in different objects which map the tables inside the database.

The database is a PostgreSQL, and the connection to the Remote platform has been performed via libpqxx, which is the official C++ client API for PostgreSQL.

The Bridger solution is almost completely developed, the next and final step will be the connection with the database PostgreSQL and related functions to store the data.

Moreover, preliminary tests have highlighted the necessity to optimise the transfer process for large amounts of data.

2.9.2 Capabilities Implementation Status

Capability Name & Description	Implementation Status	Comments / Release notes
Services Interface: **This solution will be designed to interface different services of the framework via secured communication APIs.**	**Implementation Level:** Not Implemented **Estimated Delivery Date:** MS5 (M24) **License:** License-free	The communication protocol and the encryption capabilities are fully developed. The next step is the optimization of the transfer process for large amounts of data

Table 13 Capabilities Implementation Status of the Bridger Solution

2.10 Solution - AsyncAPI Toolkit (UOC)

This solution implements the following Data Engineering Tool Set components:

- Data Collection

- Data Representation

2.10.1 News and Updates

The definition and modelling of QoS conditions for Asynchronous services, which is a capability that extends the OpenAPI specification with conditions over QoS metrics to generate the appropriate monitoring rules to check them, has been successfully implemented.

2.10.2 Future Capabilities Implementation Roadmap

Capability Name & Description	Implementation Status	Roadmap and Planning
Monitoring of QoS for Asynchronous services: **Automatically generates the monitoring infrastructure in order to monitor and assess the QoS for asynchronous services.**	**Implementation Level:** Not Implemented **Estimated Delivery Date:** MS7 (M32) **License:** Eclipse Public License 2.0	This feature is planned to be implemented by MS7 (M32)

Table 14 Future Capabilities Implementation Roadmap of the AsyncAPI Toolkit Solution

3 Mapping Solutions to AIDOaRt Data Engineering Tool Set Components

In this section, we present the list of solution components realising the functional specification of the Data Engineering Tool Set components. This mapping has been defined by the Solutions Providers based on the potential of their proposed solutions in fulfilling the description, specification, and functional interfaces of each component of the AIDOaRt Framework Architecture.

3.1 Mapping to Data Collection

In this section, we present the list of solution components realizing the "Data Collection" component of the Data Engineering Tool Set. Each solution name is followed by the partner acronym between parentheses. The full partners' names can be found in the Partners Acronyms table in the preamble of this document.

Solution Name	Rationale
ESDE (ACO)	ESDE can be understood as a simulation and (virtual platform) VP-based means to produce and collect data from different layers (application software, OS, driver, HW platform) of a complete embedded system model.
Position Monitoring for Industrial Environment (ACO)	Monitoring solutions intrinsically require data collection from different nodes or subsystems at different abstraction levels. These data need to be collected with the proper tools and techniques, parameters related to the messages size, communication protocols, data throughput, time constraints….
Kolga (AND)	Kólga can be used to get trace information from CI/CD pipelines and store that in an external storage provider for later analysis. This type of feature can be used to analyze how fast CI/CD pipelines run, what type of impact code has on the pipelines and to make adjustments based on sub-tasks inside CI jobs.
devmate (AST)	Devmate could contribute to this component through its "Code Parser" component. Devmate supports parsing code and models in different formats.
HIB_logAnalyzer (HIB)	HIB_logAnalyzer collects Data in order to get valuable information from the logs of the application (e.g., system logs, user generated input and any other relevant data from which application intelligence can be generated using AI.
a2k-runman (ITI)	The component consumes incoming streams of system and device performance and execution status monitoring data.
AsyncAPI Toolkit (UOC)	AsyncAPI toolkit will provide the capability to monitor the QoS of Asynchronous services specified in AsyncAPI.

Table 15 Solutions Mapping to the "Data Collection" Component

Solution Name	Rationale

ESDE (ACO)	ACORDE is working to enable the generation of traces from the executable models produced in ESDE. It includes both, system-level models, and platform dependent models relying on virtual platform (VP) models. The traces produced are time series data, synchronised under a global time base. These simulation data will be used to assess at design time the suitability of a specific algorithm and/or platform configuration.

Table 16 Solutions Mapping to the "IF-DESIGN-TIME-DATA-COLLECTION" Interface

Solution Name	Rationale
Position Monitoring for Industrial Environment (ACO)	The industrial positioning monitoring solution provided by ACORDE provides application and health data in real time. It uses suitable aggregators and agents to feed in run-time the data to different types of data bases.
Kolga (AND)	Kólga can be used to get trace information from CI/CD pipelines and store that in an external storage provider for later analysis. This type of feature can be used to analyse how fast CI/CD pipelines run, what type of impact code has on the pipelines and to make adjustments based on sub-tasks inside CI jobs.
devmate (AST)	devmate has capabilities to parse source code (structured text), load data from its user interface, load saved artefacts of its internal model and load data in form of its internal test model (with or without test data values) through its interfaces.
HIB_logAnalyzer (HIB)	logAnalyzer gets the data from logs in the system and the devices to process it trough the text analytics and NLP pipelines.
a2k-runman (ITI)	The a2k/detection service consumes system performance metrics and sensor data streams at run-time. It uses the a2k/monitoring service to collect performance data in real-time and applies anomaly detection algorithms to this data.
AsyncAPI Toolkit (UOC)	AsyncAPI facilitates the communication of collected runtime data via the generation of the corresponding asynchronous connected platforms' APIs.

Table 17 Solutions Mapping to the "IF-RUNTIME-DATA-COLLECTION" Interface

3.2 Mapping to Data Management

In this section, we present the list of solution components realizing the "Data Management" component of the Data Engineering Tool Set. Each solution name is followed by the partner acronym between parentheses. The full partners' names can be found in the Partners Acronyms table in the preamble of this document.

Solution Name	Rationale
ESDE (ACO)	ESDE enables gathering data from the different levels of the system model. In addition, in AIDOaRt, ACORDE aims data management so that time series from different layers of the system model and tracing is enabled. Data filtering is another capability that is expected to be required and be developed, depending on the number of sources and analysis

	performances, features like ability to downsampling on time series, or dynamic selection of data are foreseen.
Position Monitoring for Industrial Environment (ACO)	Data management is a key functionality for monitoring and control. Data monitored needs to be properly filtered and categorized to develop real-time or offline filtering and analysis capabilities.
devmate (AST)	Devmate contributes to this component through its core functionality of parsing code (structured text) and transforming it to an internal model.
Keptn (DT)	Keptn Events in case accepted as a Standard can be considered an interface between different tool, as supports IF-FILTERING-HARMONIZATION.
a2k-runman (ITI)	The component maintains and manages past histories of performance data for modelling and training. It also creates simulated data for training.
Bridger (ROTECH)	Using this solution, different services will be connected via a secured communication API which encrypts incoming data and converts it to a compatible format.
ConvHandler (ROTECH)	The solution is responsible of converting raw data to refined data relevant to the end users' necessities.

Table 18 Solutions Mapping to the "Data Management" Component

Solution Name	Rationale
ESDE (ACO)	ESDE is extended to support separating traces associated to relevant "probe points" on the different levels of the embedded system model. Thus, each trace reflects a signal on time for each prove. Then, later analysis phases can combine a selected set of traces. To enable a consistent analysis, it is fundamental data harmonization, i.e., the time labeling of the reported events under a consistent global model clock. Moreover, the enabled traceability strongly relies on considering event-based (mostly Boolean) signals. It is convenient in terms of size, and to facilitating the applicability of more generic analysis methods.
Position Monitoring for Industrial Environment (ACO)	The industrial positioning solution of ACORDE, being able to compute on its own a consistent time reference (based on GNSS technology), will be able to label application data (position) and infrastructure status data consistently on a common time basis for the deployed devices.
Keptn (DT)	Keptn Events are an excellent means to support the interconnectivity of different tools.

Table 19 Solutions Mapping to the "IF-FILTERING-HARMONIZATION" Interface

Solution Name	Rationale
devmate (AST)	devmate has capabilities for de-/serialization and transformation of data obtained or used by its various services (parser, user interface modules, generator)
Bridger (ROTECH)	The Bridger solution is divied in on-board platform, which organizes the collected and filtered data and securely transmits it, and the remote platform which receives

	the data from the on-board and organizes it to be compatible with the structure of the Database.

Table 20 Solutions Mapping to the "IF-DATA-TRANSFORMATION" Interface

Solution Name	Rationale
a2k-runman (ITI)	The a2k/detection service uses the a2k/monitoring service to collect run-time performance data from several different sources and sensors. Various filters, pre-processing, and data management tools are used before passing the aggregated data to the a2k/detection service for anomaly detection.
ConvHandler (ROTECH)	ConvHandler module provides data filtering and cleaning capabilities that will be directly applied to a stream of data coming from sensors.

Table 21 Solutions Mapping to the "IF-DATA-FILTERING-AGGREGATION" Interface

3.3 Mapping to Data Representation

In this section, we present the list of solution components realizing the "Data Representation" component of the Data Engineering Tool Set. Each solution name is followed by the partner acronym between parentheses. The full partners' names can be found in the partners' acronyms table in the preamble of this document.

Solution Name	Rationale
EMF Views (IMTA)	If the considered data is expressed as model(s), EMF Views can be used to create various data representations as model views integrating data coming from possibly different and heterogeneous data sets.
Modelio (SOFT)	Modelio would contribute to this component with its existing core capabilities, such as data modelling and metamodelling, model exchange (export / import), model transformation, document generation, model update from edited documentation, traceability, and model consistency checking.
AsyncAPI Toolkit (UOC)	AsyncAPI toolkit embeds an AsyncAPI v2.0.0 Metamodel in order to support the modelling of AsyncAPI specifications in any Eclipse UML-compatible modelling tool.
TemporalEMF (UOC)	TemporalEMF embeds a temporal metamodel profile to define models with temporal capabilities in any Eclipse UML-compatible modelling tool.
WAPIml (UOC)	WAPIml defines a Metamodel for OpenAPI in order to support the modelling of RESTful services based on the OpenAPI specification

Table 22 Solutions Mapping to the "Data Representation" Component

Solution Name	Rationale
EMF Views (IMTA)	Model views specified with EMF Views, and interrelating several models and/or metamodels together, can be considered for building a megamodel.
Modelio (SOFT)	Modelio is an extentable Modeling tool supporting several modelling standards as UML for example. Thus, it would support megamodeling by

	using a prexisting standard or customizing one.
TemporalEMF (UOC)	TemporalEMF leverages the EMF framework to provide support for representing data models.
AsyncAPI Toolkit (UOC)	AsyncAPI is able to load and represent an EMF/UML model.
WAPIml (UOC)	WAPIml leverages an UML2-compatible modeler to load and represent a data model.

Table 23 Solutions Mapping to the "IF-DATA-MEGAMODEL" Interface

3.4 Updates from the previous deliverable

Here we include the differences w.r.t. D2.2, i.e., newly added mappings, updated rationales and removed ones.

There are no new solution components realising the "Data Collection" component of the Data Engineering Tool Set at this stage. The solution “TemporalEMF (UOC)” has been removed in this final period, and the rationale for the mapping of the “ESDE (ACO)” solution to the "IF-DESIGN-TIME-DATA-COLLECTION" interface has been slightly modified.

Regarding solutions realising the "Data Management" component of the Data Engineering Tool Set, the mappings to "IF-FILTERING-HARMONIZATION" and "IF-DATA-FILTERING-AGGREGATION" interfaces from Constellation (SOFT) have been removed, and the rational ESDE (ACO) updated.

Among the solution components realising the "Data Representation" component of the Data Engineering Tool Set, a2k-modev (ITI) and a2k-runman (ITI) have been removed and the rationale for Modelio (SOFT) has been updated.

4 Mapping Use Case Requirements to AIDOaRt Data Engineering Tool Set Components

This section presents an update of the mapping of the use case requirements and data requirements to the Data Engineering Tool Set components. This mapping has been defined by the Case Study Providers based on the potential of each component of the AIDOaRt Framework Architecture in satisfying each of their use case requirements and data requirements.

For the sake of clarity, we present these mappings per component. For each component, we list the correlated requirements in a separate section for each component, grouped in two tables. The first table lists the related use case requirements, and the second table lists the related use case data requirements.

Note that each requirement identifier is prefixed by the partner acronym. The full list of partners' names can be found in the Partners Acronyms table in the preamble of this document.

4.1 Mapping to Data Collection

In this section, we present the mapping of the use case requirements and data requirements to the "Data Collection" component of the Data Engineering Tool Set.

Requirement ID	Requirement Description	Rationale
VCE_R02	AI/ML method for auto-adjusting model parameters w.r.t. similarity of execution traces of a Digital Twin with a CPS	Data engineering for data collection component is expected to ensure the gathering of run-time data required for the fulfilment of the requirement. VCE would use data collection to gather the necessary data for analysis and adjustment of a Digital twin.
VCE_R06	Integration of DevOps workflows and continuous integration/ configuration of models and corresponding technical solutions	Data engineering for data collection component is expected to ensure gathering of design time data is collected correctly regarding the required formats. VCE would use data collection to collect necessary data when designing models using DevOps workflows.
BT_R01	NLP contextual analysis of requirements and match against database of responses/solutions	The data collection component is expected to ensure that requirements are adequately collected and that integration with third-party tools that contain data works. Alstom would use the data collection component to collect requirements for training the AI/ML model

		and its uses in production.
BT_R02	ML aided control model parameterization during propulsion system testing	Alstom will use the Data Collection Component to collect current, voltage, and temperature data from sensors as time series during the test.
AVL_SEC_R03	Train ANN on SUT topology discovery using test observation	The data collection is expected to assure an adequate format for delivering data for training an ANN for building a system topology model.
AVL_SEC_R02	Use an ANN to perform plausibility checks on models	The data collection is expected to assure an adequate format for delivering data for training an ANN for model plausiblity checking.
AVL_SEC_R06	Use AI (ML) methods to learn detect abnormal behavior on a CAN	The data collection is expected to assure an adequate format for delivering data for training an ANN to learn a CAN's abnormal behavior.
AVL_SEC_R05	Use AI (ML) methods to learn on the normal behavior on a powertrain CAN	The data collection is expected to assure an adequate format for delivering data for training an ANN to learn a CAN's normal behavior.
W_R_3	Extract data from steps in DevOps process.	AIDOaRt collects data of many different types.
W_R_4	Log file storing, indexing, searching, clustering and comparing	AIDOaRt collects data of many different types.
HIB_R01	The AIDOaRt AI algorithms must be able to analyze log files (text) from the restaurant application.	Data Collection is necessary for the analysis of Logs implied in this requirement.
HIB_R02	The AIDOaRT solution will enable to process requirements expressed in natural language in Trello boards	Data Collection is necessary for the data in Trello used in this functionality
HIB_R03	The AIDOaRT solution must be able to analyze the continuous integration process and detect anomalies.	Data Collection is required on the assets to be packaged in the new bundles.
HIB_R04	The AIDOaRT AI algorithms will enable analyzing the success of deploying a new version of the POS application.	Data collection is essential for the update process to be performed in TAMUS.
W_R_2	Quality monitoring and predictions in devops process	For quality monitoring, and prediction, data collection is needed.
AVL_SEC_R08	Use live connection to remotely transfer CAN messages	Data collection via remote interface
AVL_SEC_R09	Use secure remote transfer connection	Secure remote data collection
PRO_R07	Monitor the platform in real time to reduce the downtime and the data	Uses DATA-COLLECTION to keep data from platform and sensors.

	lost	
PRO_R01	Use an Infrastructure as Code Language able to deploy the solution in different cloud providers and using different architectures / approaches (Containers & virtual machines)	Uses DATA-COLLECTION to obtain data from the infrastructure in the design time.

Table 24 Use Case Requirements Mapping to the "Data Collection" Component

Data Requirement ID	Data Requirement Description	Rationale
AVL_SEC_DR01	CAN data of a realistically behaving vehicular power train to train a plausiblity model ANN	The data collection is expected to assure an adequate format of the data for training an ANN to check a model's plausibiltiy.
AVL_SEC_DR02	CAN data of a realistically behaving vehicular power train to train an anomaly detection ANN	The data collection is expected to assure an adequate format of the data for training an ANN to for anomaly detection.
W_DR_02	To identify non-trivial indicators for quality shortcomings, the test cases could be parsed with NLP.	AIDOaRt collects data of many different types.
W_DR_03	To identify non-trivial indicators for quality shortcomings, the source code and recent changes could be parsed.	AIDOaRt collects data of many different types.
W_DR_05	To identify non-trivial indicators for quality shortcomings, the test scripts could be parsed.	AIDOaRt collects data of many different types.
W_DR_07	To identify non-trivial indicators for quality shortcomings, the logs from static code analysis could be parsed.	AIDOaRt collects data of many different types.
W_DR_08	To identify non-trivial indicators for quality shortcomings, the compilation logs could be parsed.	AIDOaRt collects data of many different types.
W_DR_09	To identify anomalies or gradual degradation in performance, the test execution logs could be parsed	AIDOaRt collects data of many different types.
W_DR_10	To identify anomalies or gradual degradation in performance, as well as functional issues and error messages, the device communication logs should be parsed.	AIDOaRt collects data of many different types.
W_DR_11	To extract information on non-functional characteristics (from e.g. free, top, etc.), the	AIDOaRt collects data of many different types.

	device communication logs should be processed.	
W_DR_12	To identify the current configuration of a device being tested, the device communication logs could be processed.	AIDOaRt collects data of many different types.
W_DR_13	To identify pass/fail/etc-history of test executions, the test results database should be processed.	AIDOaRt collects data of many different types.
W_DR_14	To identify human-identified risks and risk levels, the risk management data in spreadsheets could be processed.	AIDOaRt collects data of many different types.
W_DR_15	To identify the topology of a test system, which may be useful in a bigger analysis, the test system topology descriptions could be used.	AIDOaRt collects data of many different types.
W_DR_16	To identify the topology of test cases, which may be useful in a bigger analysis, the test case topology descriptions could be used.	AIDOaRt collects data of many different types.
TEK_Data_02	Monitoring data of test execution and processing of results.	Data collection from various sources.
PRO_IoT	IoT devices periodically send data collected by the different sensors they contain. This data is sent via JSON messages. Regarding trucks and cranes, they send one message per second with information about the vehicle's operation/status. The important thing about this data is to verify that it is sent and that the messages are not lost. The content of the messages is not relevant to the purpose of the use case.	Real time data collected from different IoT Nodes
PRO_log	All the resources of the platform including the IoT devices installed in the vehicles will provide information about resource usage (Memory, CPU, Disk).	Real time data collected from different IoT sensors.
PRO_Monitoring	The monitoring platform with collaboration with some AI algorithms will detect problems in the platform. Every time that a problem is found an alarm/notification will be generated.	Real time data collected from different IoT devices
PRO_IaC	This file will contain the description of the platform to be deployed in an IaC language "Terraform".	Infrastructure as Code of the Use case

Table 25 Use Case Data Requirements Mapping to the "Data Collection" Component

Requirement ID	Requirement Description	Rationale

VCE_R06	Integration of DevOps workflows and continuous integration/ configuration of models and corresponding technical solutions	The interface can enable the collection of data from system operations that is required for complete DevOps workflow.
HIB_R02	The AIDOaRT solution will enable to process requirements expressed in natural language in Trello boards	The requirements analysis solution proposed collects data from Design-time (the raw text from the requirements) to use for analysis and generation of relevant metadata (requirement topic and owner).
BT_R01	NLP contextual analysis of requirements and match against database of responses/solutions	The interface will enable the selection/decomposition of the requirements for training the ML models.
BT_R02	ML aided control model parameterization during propulsion system testing	The interface will enable the selection/decomposition of the motor operation data for training the ML models.
AVL_SEC_R02	Use an ANN to perform plausibility checks on models	The data collection is expected to assure an adequate format for delivering data for training an ANN for model plausiblity checking.
AVL_SEC_R03	Train ANN on SUT topology discovery using test observation	The data collection is expected to assure an adequate format for delivering data for training an ANN for building a system topology model.
PRO_R01	Use an Infrastructure as Code Language able to deploy the solution in different cloud providers and using different architectures / approaches (Containers & virtual machines)	Uses IF-DESIGN-TIME-DATA-COLLECTION to obtain data from infrastructure needed to deploy the SPMP platform to use for analysis.
HIB_R03	The AIDOaRT solution must be able to analyze the continuous integration process and detect anomalies.	Data is collected at design time to ensure that the continuous integration process can be documented and analyzed. Data captured includes the different version changes and applied updates to each component of the TAMUS system.
HIB_R04	The AIDOaRT AI algorithms will enable analyzing the success of deploying a new version of the POS application.	Data is collected at design time to ensure that updates are applied correctly on a given TAMUS system. This includes the outputs of the update scripts.

Table 26 Use Case Requirements Mapping to the "IF-DESIGN-TIME-DATA-COLLECTION" Interface

Data Requirement ID	Data Requirement Description	Rationale
PRO_IaC	This file will contain the description of the platform to be	Uses IF-DESIGN-TIME-DATA-COLLECTION to collect data from the infrastructure

	deployed in an IaC language "Terraform".	needed to deploy the SPMP platform to use for analysis.
AVL_SEC_DR01	CAN data of a realistically behaving vehicular power train to train a plausiblity model ANN	The data collection is expected to assure an adequate format of the data for training an ANN to check a model's plausibiltiy.

Table 27 Use Case Data Requirements Mapping to the "IF-DESIGN-TIME-DATA-COLLECTION" Interface

Requirement ID	Requirement Description	Rationale
VCE_R02	AI/ML method for auto-adjusting model parameters w.r.t. similarity of execution traces of a Digital Twin with a CPS	The interface can enable the correct implementation of the digital twin.
VCE_R06	Integration of DevOps workflows and continuous integration/ configuration of models and corresponding technical solutions	The interface can enable the collection of data from system operations that is required for a complete DevOps workflow.
HIB_R01	The AIDOaRt AI algorithms must be able to analyze log files (text) from the restaurant application.	The HIB_logAnalyzer collects data from the runtime to drive the NLP analysis of the system logs.
W_R_2	Quality monitoring and predictions in devops process	By collecting data at runtime, e.g. resource usage of systems involved in nightly testing, one could monitor or predict quality shortcomings.
W_R_3	Extract data from steps in DevOps process.	For W_R_3, data collection is central.
W_R_4	Log file storing, indexing, searching, clustering and comparing	For W_R_4, collecting log files is central.
AVL_SEC_R05	Use AI (ML) methods to learn on the normal behavior on a powertrain CAN	The data collection is expected to assure an adequate format for delivering data for training an ANN to learn a CAN's normal behavior.
AVL_SEC_R06	Use AI (ML) methods to learn detect abnormal behavior on a CAN	The data collection is expected to assure an adequate format for delivering data for training an ANN to learn a CAN's abnormal behavior.
AVL_SEC_R08	Use live connection to remotely transfer CAN messages	Data collection via remote interface
AVL_SEC_R09	Use secure remote transfer connection	Secure remote data collection
PRO_R07	Monitor the platform in real time to reduce the downtime and the data lost	This RUNTIME-DATA-COLLECTION interface would offer our use case the capability to obtain data from the platform.

Table 28 Use Case Requirements Mapping to the "IF-RUNTIME-DATA-COLLECTION" Interface

Data Requirement ID	Data Requirement Description	Rationale
PRO_log	All the resources of the platform including the IoT devices installed in the vehicles will provide information about resource usage (Memory, CPU, Disk).	Uses IF-RUNTIME-DATA-COLLECTION to collect data coming from IoT devices linked to the SPMP platform.
PRO_IoT	IoT devices periodically send data collected by the different sensors they contain. This data is sent via JSON messages. Regarding trucks and cranes, they send one message per second with information about the vehicle's operation/status. The important thing about this data is to verify that it is sent and that the messages are not lost. The content of the messages is not relevant to the purpose of the use case.	Uses IF-RUNTIME-DATA-COLLECTION to collect data coming from IoT devices linked to the SPMP platform.
AVL_SEC_DR02	CAN data of a realistically behaving vehicular power train to train an anomaly detection ANN	The data collection is expected to assure an adequate format of the data for training an ANN to for anomaly detection.
W_DR_02	To identify non-trivial indicators for quality shortcomings, the test cases could be parsed with NLP.	Data needs to be collected.
W_DR_03	To identify non-trivial indicators for quality shortcomings, the source code and recent changes could be parsed.	Data needs to be collected.
W_DR_05	To identify non-trivial indicators for quality shortcomings, the test scripts could be parsed.	Data needs to be collected.
W_DR_07	To identify non-trivial indicators for quality shortcomings, the logs from static code analysis could be parsed.	Data needs to be collected.
W_DR_08	To identify non-trivial indicators for quality shortcomings, the compilation logs could be parsed.	Data needs to be collected.
W_DR_09	To identify anomalies or gradual degradation in performance, the test execution logs could be parsed	Data needs to be collected.
W_DR_10	To identify anomalies or gradual degradation in performance, as well as functional issues and error messages, the device communication logs should be parsed.	Data needs to be collected.
W_DR_11	To extract information on non-functional characteristics (from e.g. free, top, etc.), the	Data needs to be collected.

	device communication logs should be processed.	
W_DR_12	To identify the current configuration of a device being tested, the device communication logs could be processed.	Data needs to be collected.
W_DR_13	To identify pass/fail/etc-history of test executions, the test results database should be processed.	Data needs to be collected.
W_DR_15	To identify the topology of a test system, which may be useful in a bigger analysis, the test system topology descriptions could be used.	Data needs to be collected.
W_DR_16	To identify the topology of test cases, which may be useful in a bigger analysis, the test case topology descriptions could be used.	Data needs to be collected.
W_DR_14	To identify human-identified risks and risk levels, the risk management data in spreadsheets could be processed.	Data needs to be collected.
PRO_Monitoring	The monitoring platform with collaboration with some AI algorithms will detect problems in the platform. Every time that a problem is found an alarm/notification will be generated.	This RUNTIME-DATA-COLLECTION interface would offer our use case the capability to obtain data from the platform.
TEK_Data_02	Monitoring data of test execution and processing of results.	Run-time data collection through is needed during the verification phase of the development.

Table 29 Use Case Data Requirements Mapping to the "IF-RUNTIME-DATA-COLLECTION" Interface

4.2 Mapping to Data Management

In this section, we present the mapping of the use case requirements and data requirements to the "Data Management" component of the Data Engineering Tool Set.

Requirement ID	Requirement Description	Rationale
VCE_R02	AI/ML method for auto-adjusting model parameters w.r.t. similarity of execution traces of a Digital Twin with a CPS	Data engineering for data management component is expected to ensure data gathered is filtered, transformed, and represented accordingly to the demands of the AI/ML component(s) used for auto adjustment. VCE would use data management to ensure the data captured and required for the AI/ML component(s) is correctly represented and harmonized.

BT_R02	ML aided control model parameterization during propulsion system testing	Alstom will use the Data Management Component to clean, resample, and normalize the collected data from several sensors.
AVL_SEC_R02	Use an ANN to perform plausibility checks on models	The Data Management Component should be able to normalize data for ANN training.
AVL_SEC_R04	Use formal model checking methods to derive test cases out of a system model	The Data Management Component should be able to normalize data to use a learned model for model checking.
W_R_3	Extract data from steps in DevOps process.	Data collected is stored and managed.
W_R_4	Log file storing, indexing, searching, clustering and comparing	Data collected is stored and managed.
CSY_R01	Find a representation of PO (hypothesis + goal) that can permit ML	

Table 30 Use Case Requirements Mapping to the "Data Management" Component

Data Requirement ID	Data Requirement Description	Rationale
W_DR_02	To identify non-trivial indicators for quality shortcomings, the test cases could be parsed with NLP.	Data collected is stored and managed.
W_DR_03	To identify non-trivial indicators for quality shortcomings, the source code and recent changes could be parsed.	Data collected is stored and managed.
W_DR_05	To identify non-trivial indicators for quality shortcomings, the test scripts could be parsed.	Data collected is stored and managed.
W_DR_07	To identify non-trivial indicators for quality shortcomings, the logs from static code analysis could be parsed.	Data collected is stored and managed.
W_DR_08	To identify non-trivial indicators for quality shortcomings, the compilation logs could be parsed.	Data collected is stored and managed.
W_DR_09	To identify anomalies or gradual degradation in performance, the test execution logs could be parsed	Data collected is stored and managed.
W_DR_10	To identify anomalies or gradual degradation in performance, as well as functional issues and error messages, the device communication logs should be parsed.	Data collected is stored and managed.
W_DR_11	To extract information on non-functional characteristics (from e.g. free, top, etc.), the device communication logs should be processed.	Data collected is stored and managed.

W_DR_12	To identify the current configuration of a device being tested, the device communication logs could be processed.	Data collected is stored and managed.
W_DR_13	To identify pass/fail/etc.-history of test executions, the test results database should be processed.	Data collected is stored and managed.
W_DR_14	To identify human-identified risks and risk levels, the risk management data in spreadsheets could be processed.	Data collected is stored and managed.
W_DR_15	To identify the topology of a test system, which may be useful in a bigger analysis, the test system topology descriptions could be used.	Data collected is stored and managed.
W_DR_16	To identify the topology of test cases, which may be useful in a bigger analysis, the test case topology descriptions could be used.	Data collected is stored and managed.
TEK_Data_01	Monitoring data of test execution and processing of results.	The cleaning, analysis, and management functions are needed by the use case in different stages of the development.
TEK_Data_02	Monitoring data of test execution and processing of results.	The cleaning, analysis, and management functions are needed by the use case in different stages of the development.
TEK_Data_03	Monitoring data for AI models for diagnostics and prognostics.	The cleaning, analysis, and management functions are needed by the use case in different stages of the development.
PRO_Monitoring	The monitoring platform with collaboration with some AI algorithms will detect problems in the platform. Every time that a problem is found an alarm/notification will be generated.	Uses all data management interfaces to handle the data from devices. Uses DATA-COLLECTION to keep data from platform.
PRO_IoT	IoT devices periodically send data collected by the different sensors they contain. This data is sent via JSON messages. Regarding trucks and cranes, they send one message per second with information about the vehicle's operation/status. The important thing about this data is to verify that it is sent and that the messages are not lost. The content of the messages is not relevant to the purpose of the use case.	Uses DATA-COLLECTION to keep data from sensors.

Table 31 Use Case Data Requirements Mapping to the "Data Management" Component

Requirement ID	Requirement Description	Rationale
W_R_4	Log file storing, indexing, searching, clustering and comparing	Log files may come in different formats, a harmonizing approach to fill gaps in formats (e.g., adding a uniform timestamp), could be needed.
CSY_R01	Find a representation of PO (hypothesis + goal) that can permit ML	This functional interface would help us to manage the anonymisation aspects of our proof database. Anonymisation modifies the proof models such as locally each proof remains coherent, with variable names and labels equally modified along the different propositions. Globally, the same variable are not transformed equally, meaning that a set of hypothesis concerning a special aspect of a project, i.e. the time variable, can be present in several proofs but transformed differently in each proofs due to anonymisation. Harmonization could help by representing relations in proofs in a way that anonymisation would have a reduced impact.
BT_R02	ML aided control model parameterization during propulsion system testing	The interface will enable the alignment of the motor operation data for training ML models.
AVL_SEC_R02	Use an ANN to perform plausibility checks on models	The Data Management Component should be able to normalize data for ANN training.
AVL_SEC_R04	Use formal model checking methods to derive test cases out of a system model	The Data Management Component should be able to normalize data to use a learned model for model checking.

Table 32 Use Case Requirements Mapping to the "IF-FILTERING-HARMONIZATION" Interface

Data Requirement ID	Data Requirement Description	Rationale
PRO_Monitoring	The monitoring platform with collaboration with some AI algorithms will detect problems in the platform. Every time that a problem is found an alarm/notification will be generated.	This functional interface would help us to normalize data coming in different formats from different sensors. So that later they can be used in a simple way when we work with them.
PRO_IoT	IoT devices periodically send data collected by the different sensors they contain. This data is sent via JSON messages. Regarding trucks and cranes, they send one message per second with information about the vehicle's operation/status. The important thing about this data is to verify that it is sent and that the messages are not lost. The content of the messages is not relevant to the purpose of the use case.	This functional interface would help us to normalize data coming in different sensors that will have several different formats.

Table 33 Use Case Data Requirements Mapping to the "IF-FILTERING-HARMONIZATION" Interface

Requirement ID	Requirement Description	Rationale
VCE_R02	AI/ML method for auto-adjusting model parameters w.r.t. similarity of execution traces of a Digital Twin with a CPS	The interface can enable the process of reading and understanding real sensor data to the adjustment of a corresponding model.
W_R_4	Log file storing, indexing, searching, clustering and comparing	Data extraction could involve transforming data, say storing a log file in a different format, or adding various kinds of meta data to it.
W_R_3	Extract data from steps in DevOps process.	Data extraction could involve transforming data, say storing a log file in a different format, or adding various kinds of meta data to it.
CSY_R01	Find a representation of PO (hypothesis + goal) that can permit ML	This functional interface would help us to discover meaningful relation inside our hypothesis models. The Hypothesis, as structured forms naturally carries a strong semantic. This semantic has to be present for machine learning to be efficient.

Table 34 Use Case Requirements Mapping to the "IF-DATA-TRANSFORMATION" Interface

Data Requirement ID	Data Requirement Description	Rationale
PRO_Monitoring	The monitoring platform with collaboration with some AI algorithms will detect problems in the platform. Every time that a problem is found an alarm/notification will be generated.	This functional interface would help us to adapt the different types of data to the desired structure in order to work with them correctly.
PRO_IoT	IoT devices periodically send data collected by the different sensors they contain. This data is sent via JSON messages. Regarding trucks and cranes, they send one message per second with information about the vehicle's operation/status. The important thing about this data is to verify that it is sent and that the messages are not lost. The content of the messages is not relevant to the purpose of the use case.	This functional interface would help us to adapt the different types of data coming from different IoT sensors.
TEK_Data_01	Monitoring data of test execution and processing of results.	The interface supports data transformation capabilities to transform data collected in the DATA COLLECTION component to the internal representation defined in the DATA REPRESENTATION

		component and satisfies the TEK_DATA_01 requirement related to the Monitoring data of test execution and processing of results. Bridger (ROTECH)
TEK_Data_02	Monitoring data of test execution and processing of results.	The interface supports data transformation capabilities to transform data collected in the DATA COLLECTION component to the internal representation defined in the DATA REPRESENTATION component and satisfies the TEK_DATA_02 requirement related to the Monitoring data of test execution and processing of results. Bridger (ROTECH)
W_DR_02	To identify non-trivial indicators for quality shortcomings, the test cases could be parsed with NLP.	Some data might be transformed, e.g., by changing into a common JSON format.
W_DR_03	To identify non-trivial indicators for quality shortcomings, the source code and recent changes could be parsed.	Some data might be transformed, e.g., by changing into a common JSON format.
W_DR_05	To identify non-trivial indicators for quality shortcomings, the test scripts could be parsed.	Some data might be transformed, e.g., by changing into a common JSON format.
W_DR_07	To identify non-trivial indicators for quality shortcomings, the logs from static code analysis could be parsed.	Some data might be transformed, e.g., by changing into a common JSON format.
W_DR_08	To identify non-trivial indicators for quality shortcomings, the compilation logs could be parsed.	Some data might be transformed, e.g., by changing into a common JSON format.
W_DR_09	To identify anomalies or gradual degradation in performance, the test execution logs could be parsed	Some data might be transformed, e.g., by changing into a common JSON format.
W_DR_10	To identify anomalies or gradual degradation in performance, as well as functional issues and error messages, the device communication logs should be parsed.	Some data might be transformed, e.g., by changing into a common JSON format.
W_DR_11	To extract information on non-functional characteristics (from e.g. free, top, etc.), the device communication logs should be processed.	Some data might be transformed, e.g., by changing into a common JSON format.
W_DR_12	To identify the current configuration of a device being tested, the device communication logs could be processed.	Some data might be transformed, e.g., by changing into a common JSON format.

W_DR_13	To identify pass/fail/etc.-history of test executions, the test results database should be processed.	Some data might be transformed, e.g., by changing into a common JSON format.
W_DR_14	To identify human-identified risks and risk levels, the risk management data in spreadsheets could be processed.	Some data might be transformed, e.g., by changing into a common JSON format.
W_DR_15	To identify the topology of a test system, which may be useful in a bigger analysis, the test system topology descriptions could be used.	Some data might be transformed, e.g., by changing into a common JSON format.
W_DR_16	To identify the topology of test cases, which may be useful in a bigger analysis, the test case topology descriptions could be used.	Some data might be transformed, e.g., by changing into a common JSON format.

Table 35 Use Case Data Requirements Mapping to the "IF-DATA-TRANSFORMATION" Interface

Data Requirement ID	Data Requirement Description	Rationale
PRO_Monitoring	The monitoring platform with collaboration with some AI algorithms will detect problems in the platform. Every time that a problem is found an alarm/notification will be generated.	This functional interface would help us to adapt to select the different types of data to be able to perform actions on a specific set of data.
PRO_IoT	IoT devices periodically send data collected by the different sensors they contain. This data is sent via JSON messages. Regarding trucks and cranes, they send one message per second with information about the vehicle's operation/status. The important thing about this data is to verify that it is sent and that the messages are not lost. The content of the messages is not relevant to the purpose of the use case.	This functional interface would help us to adapt to select the different types of data to be able to perform actions on a specific set of sensors.
TEK_Data_03	Monitoring data for AI models for diagnostics and prognostics.	The interface supports the capabilities to filter and aggregate data and satisfies the TEK_DATA_03 requirement related to the Monitoring data for AI models for diagnostics and prognostics. ConvHandler (ROTECH)
W_DR_02	To identify non-trivial indicators for quality shortcomings, the test cases could be parsed with NLP.	Some data should be filtered, e.g., one may not want all data, but only the subset with failures.

W_DR_03	To identify non-trivial indicators for quality shortcomings, the source code and recent changes could be parsed.	Some data should be filtered, e.g., one may not want all data, but only the subset with failures.
W_DR_05	To identify non-trivial indicators for quality shortcomings, the test scripts could be parsed.	Some data should be filtered, e.g., one may not want all data, but only the subset with failures.
W_DR_07	To identify non-trivial indicators for quality shortcomings, the logs from static code analysis could be parsed.	Some data should be filtered, e.g., one may not want all data, but only the subset with failures.
W_DR_08	To identify non-trivial indicators for quality shortcomings, the compilation logs could be parsed.	Some data should be filtered, e.g., one may not want all data, but only the subset with failures.
W_DR_09	To identify anomalies or gradual degradation in performance, the test execution logs could be parsed	Some data should be filtered, e.g., one may not want all data, but only the subset with failures.
W_DR_10	To identify anomalies or gradual degradation in performance, as well as functional issues and error messages, the device communication logs should be parsed.	Some data should be filtered, e.g., one may not want all data, but only the subset with failures.
W_DR_11	To extract information on non-functional characteristics (from e.g. free, top, etc.), the device communication logs should be processed.	Some data should be filtered, e.g., one may not want all data, but only the subset with failures.
W_DR_12	To identify the current configuration of a device being tested, the device communication logs could be processed.	Some data should be filtered, e.g., one may not want all data, but only the subset with failures.
W_DR_14	To identify human-identified risks and risk levels, the risk management data in spreadsheets could be processed.	Some data should be filtered, e.g., one may not want all data, but only the subset with failures.
W_DR_15	To identify the topology of a test system, which may be useful in a bigger analysis, the test system topology descriptions could be used.	Some data should be filtered, e.g., one may not want all data, but only the subset with failures.
W_DR_16	To identify the topology of test cases, which may be useful in a bigger analysis, the test case topology descriptions could be used.	Some data should be filtered, e.g., one may not want all data, but only the subset with failures.

Table 36 Use Case Data Requirements Mapping to the "IF-DATA-FILTERING-AGGREGATION" Interface

4.3 Mapping to Data Representation

In this section, we present the mapping of the use case requirements and data requirements to the "Data Representation" component of the Data Engineering Tool Set.

Requirement ID	Requirement Description	Rationale
BT_R02	ML aided control model parameterization during propulsion system testing	Alstom will use the Data Representation Component to clean, resample, and normalize the collected data from several sensors.
AVL_SEC_R01	Use automata learning and ML techniques to derive SUT models	AVL will use Data representation facilitate (manual) verification the correctness of the learned model.
AVL_SEC_R03	Train ANN on SUT topology discovery using test observation	AVL will use Data representation verify the topology modelling.
AVL_SEC_R04	Use formal model checking methods to derive test cases out of a system model	AVL will use Data representation to model proof obligations into models.
W_R_3	Extract data from steps in DevOps process.	Some of the collected data is annotated or indexed to simplify further work, such as indexing for search, etc.
W_R_4	Log file storing, indexing, searching, clustering and comparing	Some of the collected data is annotated or indexed to simplify further work, such as indexing for search, etc.

Table 37 Use Case Requirements Mapping to the "Data Representation" Component

Requirement ID	Requirement Description	Rationale
AVL_SEC_R01	Use automata learning and ML techniques to derive SUT models	AVL will use Data representation facilitate (manual) verification the correctness of the learned model.
AVL_SEC_R03	Train ANN on SUT topology discovery using test observation	AVL will use Data representation verify the topology modelling.
AVL_SEC_R04	Use formal model checking methods to derive test cases out of a system model	AVL will use Data representation to model proof obligations into models.
W_R_3	Extract data from steps in DevOps process.	Extracted data from the DevOps process should be modelled, managed and stored.
W_R_4	Log file storing, indexing, searching, clustering and comparing	Extracted logs from the DevOps process should be modelled, managed and stored.
BT_R02	ML aided control model parameterization during propulsion system testing	This interface will be used to verify the correctness of the trained models.

Table 38 Use Case Requirements Mapping to the "IF-DATA-MEGAMODEL" Interface

4.4 Concluding remark

A mapping of the use case requirements and data requirements to the Data Engineering Tool Set components has been presented in this section. This effort has been made by Case Study Providers with the expectation that it may help future users of the components of the AIDOaRt Framework Architecture to identify their usefulness and eventually satisfy requirements of potentially related new applications, use cases or data requirements.

The use case functional and data requirements exhibit the need to collect and handle both static design-time and dynamic run-time types of data.

5 Applications of AIDOaRt Data Engineering Tool Set Solutions in Use Cases

This section presents some examples of the way solutions that encompass the various use cases data models are applied in some of the challenges appointed by Use Case leaders.

The cases shown here are limited to solutions related to the Data Engineering Tool Set. They correspond to challenges that were identified, discussed and faced mainly (but not only) during the hackathons and plenary meetings realised along the project.

5.1 Operating Life Monitoring - TEK, ROTECH

Ro Technology is collaborating with TEKNE in the third scenario, TEK_UCS_03, called Operating Life Monitoring. The collaboration is focused on the data management performed after the on-board sensing, such as cleaning, transmission, storage and retrieval, considering the requirements related to the 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.

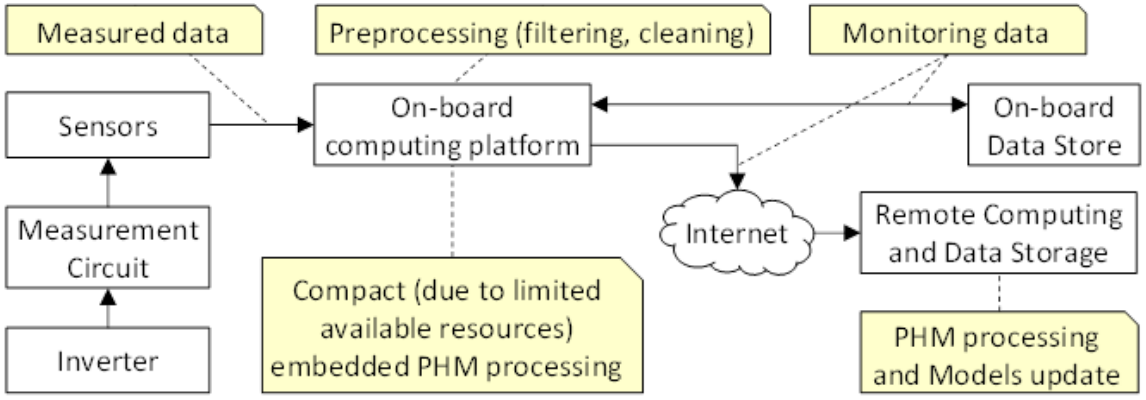

Figure 9 TEK_UCS_3 architecture

In this context, ROTECH is providing 2 solutions to cover the aforementioned needs: the ConvHandler and the Bridger, both implementing the Data Management component of the AIDOaRt Data Engineering Tool Set.

Figure 9 represents the high-level architecture of the TEK scenario. Both solutions are used after the Measured data is produced. In particular, the ConvHandler is installed in the Onboard computing platform and performs the Preprocessing operations on the raw data received.

The Bridger is composed of two instances:

- one installed on the Onboard computing platform, called **Bridger Onboard**. It encrypts the data and publishes it on a MQTT Broker;

- one installed in the Remote Computing and Data Storage platform, called **Bridger Remote**. It decrypts the data and stores it in a Database.

Both the Bridger Onboard and the ConvHandler are strictly related since the data produced by the latter is then encrypted and sent to the MQTT by the former.

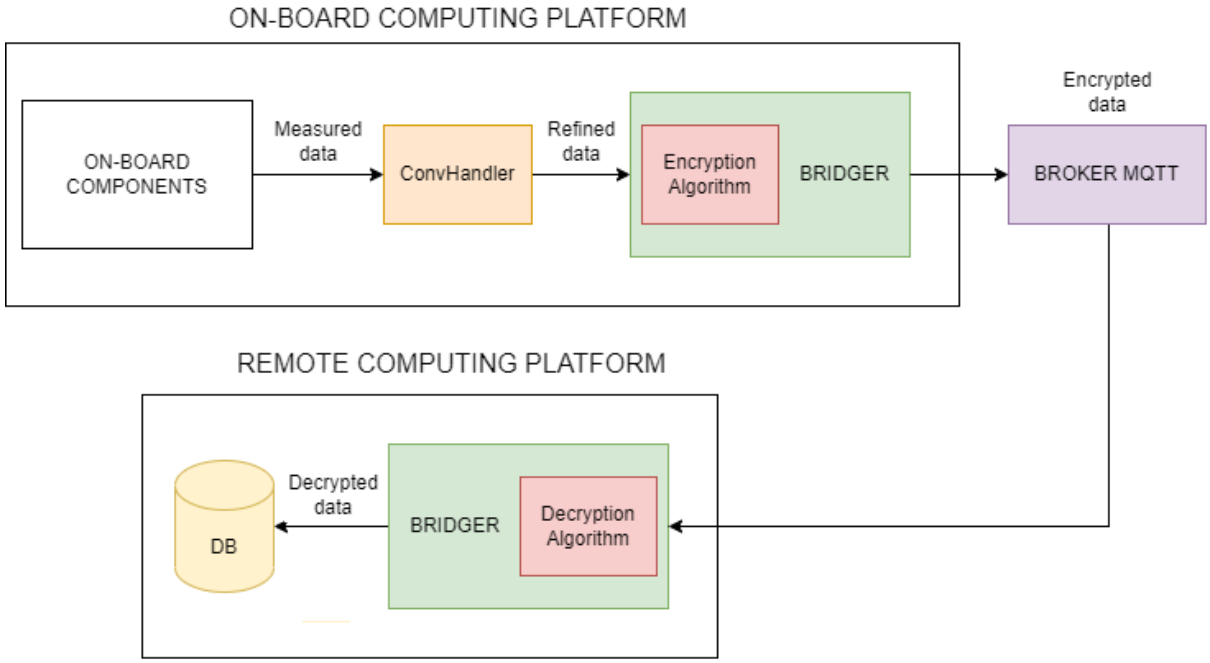

Figure 10 Data flow

Figure 10 shows the interaction between the ConvHandler and the Bridger and the data management in the process.

So far ROTECH has conducted lab tests using an .xlsx file provided by TEK as a data set. All the components are working as expected and the next step is the optimisation of the communication between the two Bridger instances. In fact, considering the large amount of data produced by the system, an improvement of the data exchange process is needed. We are now taking into consideration the usage of a compression algorithm in order to drastically increase the transfer speed and improve the overall performance of the system.

5.2 Application in Anomaly Detection in Cyber-Physical Systems – Location Optimization Challenge - PRO, ACORDE, ITI, UOC

The Smart Port Monitoring Platform (SPMP) is an IoT platform responsible for collecting and storing data from many different sensors. Currently, the quality of the data received is long breached, not only because of problems in collecting the data (frequency of sending and quality of the data received), but also because of problems in the hardware infrastructure over which the system runs.

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. We have made significant progress in the following areas:

- anomaly detection techniques;
- Simulation techniques to determine the needs of the platform in different scenarios; and
- service quality of IoT sensors and platforms.

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.

The challenge continues to be divided into 3 sub-challenges:

- **Data Quality IoT** focuses on ensuring that the information received is correct. For this purpose, the Universitat Oberta Catalunya (UOC) previously provided a mechanism to ensure that the data received is syntactically correct, and now has developed a meta-model towards checking that it is also timely delivered.
- **Infrastructure Performance & Availability** are addressed in collaboration with the Instituto Tecnológico de Informática (ITI), guaranteeing that the infrastructure and elements are available and sufficient to process the data.
- **Location Optimization**, in which ACORDE works on improving the positioning of the different elements, improving accuracy and resilience, exploiting the distributed architecture, and better integrating the data in an advanced monitoring platform.

This challenge is related to the following AIDOaRt components:

- Data Collection
- Data Management
- Data Representation

5.2.1 Data Quality IoT

Cyber-physical systems (CPSs) are sometimes built on top of message-driven architectures, where asynchronous communication within the system is key for its scalability and performance. In distributed architectures like CPSs ones, there must be a shared understanding of the structure, semantics and latency of the messages that are published and consumed by their different components.

Previously, we adopted a model-driven paradigm to efficiently design and develop a message-driven architecture. The proposal relies on the AsyncAPI Specification to formalise and (semi)automate the design and implementation of such architecture by using the AsyncAPI Toolkit developed at UOC. We created a data model for the messages and the architecture using the UML profile for Industry 4.0 that

the AsyncAPI toolkit provides. As a result of the model-to-text transformations that the tool provides, more than 9,000 lines of API code were automatically generated. With this approach, the Java code ensures that messages are syntactically correct (data integrity).

We have progressed in this challenge by specifying a meta-model to define the quality of service requirements for asynchronous messaging, towards automatically deriving and instantiating a dashboard to measure the compliance of the conditions established in an SLA at run time. This meta-model allowed us then to define a set of service-level objectives and their related QoS metrics. It has been designed as an extension of the AsyncAPI specification and it tailors the ISO/IEC 25010 quality model and the key concepts of WS-Agreement adapted for asynchronous messaging. The code listing in Figure 11 illustrates an excerpt of the meta-model grammar:

```
Slo:
        {Slo} name=AnyString ':' condition=BooleanCondition
;

AbstractQoSMetric:
        Reference | QoSMetric;

QoSMetric:
        ('{'
                (AtomicQoSMetric | DerivedQoSMetric) ','
                (
                  ( '"description"' ':' description=AnyString ','? )?
                & ( '"unit"' ':' unit=QoSMetricUnit ','? )
                & ( '"dataType"' ':' dataType=QoSMetricType ','? )
                )
        '}');

 DerivedQoSMetric:
        {DerivedQoSMetric}(
              ('"metricType"' ':' '"derived"' ',' )
                & ('"window"' ':'  window = AnyString ','? )
                & ('"windowUnit"' ':'  windowUnit = WindowUnit ','? )
                & ('"aggregationFunction"' ':'  aggregationFunction = AnyString ','? )
                & ('"atomicMetric"' ':'  atomicMetric = AbstractQoSMetric  ','? )
        )
;
```

Figure 11 Excerpt of the meta-model grammar

Figure 12 depicts an excerpt of the JSON code for the definition of SLOs and metrics:

```
"x-qosMetrics" : {
    "airLatency": {
        "metricType" : "atomic",
        "dataType" : "integer",
        "description" : "Average latency in hours",
        "unit" : "hours"
    },
    "humidityLatency": {
        "metricType" : "atomic",
        "dataType" : "integer",
        "description" : "Average latency in hours",
        "unit" : "hours"
    },
    "irrigationLatency": {
        "metricType" : "atomic",
        "dataType" : "integer",
        "description" : "Average latency in hours",
        "unit" : "hours"
    },
    "noiseLatency": {
        "metricType" : "atomic",
        "dataType" : "integer",
        "description" : "Average latency in minutes",
        "unit" : "minutes"
    },

"x-sla" : {
    "guaranteeTerm" : {
        "scopes" : { "scope1": {"$ref": "Observation"}},
        "slos" : {
            "sloAirLatency" : {
                "qosMetric" : { "$ref" : "airLatency" },
                "operator" : "=",
                "value" : "1"
            },
            "sloHumidityLatency" : {
                "qosMetric" : { "$ref" : "humidityLatency" },
                "operator" : "=",
                "value" : "1"
            },
            "sloIrrigationLatency" : {
                "qosMetric" : { "$ref" : "irrigationLatency" },
                "operator" : "=",
                "value" : "1"
            },
            "sloNoiseLatency" : {
                "qosMetric" : { "$ref" : "noiseLatency" },
                "operator" : "=",
                "value" : "15"
            },
```

Figure 12 Excerpts of the JSON code for the definition of SLOs and metrics

The next steps include the MDE code for transforming the service level objectives defined in JSON format to a running dashboard. When finalised, we will be able to continuously evaluate the health of the system by measuring the number of anomalies automatically detected (either as a result of a data structure violation or because of latency issues) and contrast that figure against the total occurrences.

5.2.2 Infrastructure Performance Resizing of Resources Based on Current Workload - Power Aware Scheduling

Cyber-physical systems often operate in dynamic and unpredictable environments where power availability may vary. Power awareness enables active power management techniques that allow CPS components to adjust their power consumption based on workload demands or power availability, effectively balancing performance and energy usage.
A power awareness algorithm is part of the a2k/tuning solution. This solution uses machine learning based algorithms to obtain more efficient power consumption, allowing applications to run safely with lower costs.
This solution component provides instances of the Data Management and Data Collection services of the AIDOaRt architecture.
This algorithm adjusts the processor frequency by considering the predicted workload demands. We use a real-time simulator (part of ITI's A2K software suite) and a prediction method based on AI to do that. Using the simulator, we can obtain the execution, release, start and finishing times of a set of tasks that run in a processor. We then use this data to create a prediction model that enables tuning the processor frequency to optimise the power, whilst keeping the time constraints of the real-time system, that is, ensuring that all the software tasks meet their deadlines. Figure 13 shows a schematic of this method, where f_{CPU} is the processor frequency, r_i, s_i and f_i are the release, start and finishing times, C_i is the execution time and d_i is the deadline of task i, which is represented with τ_i.

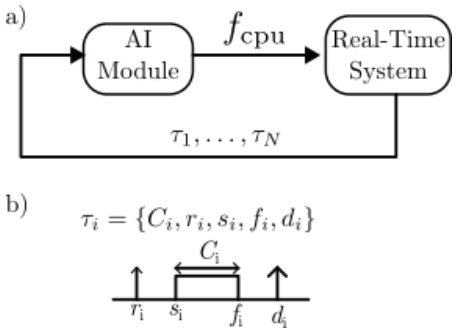

Figure 13 a) Block diagram of the power aware solution; The AI module containing the prediction methods which will invoke, if necessary, CPU frequency changes considering the computational tasks (τi) parameters. b) Layout of the time task parameters

To develop the prediction methods, which will be integrated into the AI module, we have implemented the following steps:

- Seven regression models have been evaluated and compared to determine which model can estimate different parameters with the best performance. The main idea is to assess the optimal workload with minor error. With this purpose, the following regression models have been adjusted and compared: Linear Regression, Ridge Regression, Lasso Regression, Elastic Net, Support Vector Regression, Random Forest Regressor, and XGBoost Regressor. The XGBoost model showed the best performance and was selected for subsequent analysis.
- The selected regression model has been adapted for time series prediction. This way, as shown in Figure 14, it could also be used to predict the system's behaviour (a2k/detection service) and invoke system mode changes before anomalies related to workload arise (a2k/tuning service). The main idea is to apply the regression model using sliding windows

that move forward through time. To do this, each observation will be composed of the information from n time instants (window size: t_{-n}, … t_{-2}, t_{-1}), which will be used to predict the next instant (t_{+1}).

- The previous model has been modified so that it is possible to predict several time instants (t_{+1}, t_{+2}, t_{+3} …), see Figure 15. This change has been studied in two ways: (1) direct multioutput. The multiple-output regression problem is divided into multiple single-output regression problems. This strategy consists of fitting one regressor per target; (2) chained multioutput. The multiple-output regression problem is divided into dependent single-output regression problems. In this strategy, each model makes a prediction in the order specified by the chain using all the available features provided to the model plus the predictions of models that are earlier in the chain.

All the algorithms have been developed using simulated data created by ITI. The necessary pipeline has also been developed to prepare the different models (regression, simple prediction, multiple predictions) for their deployment.

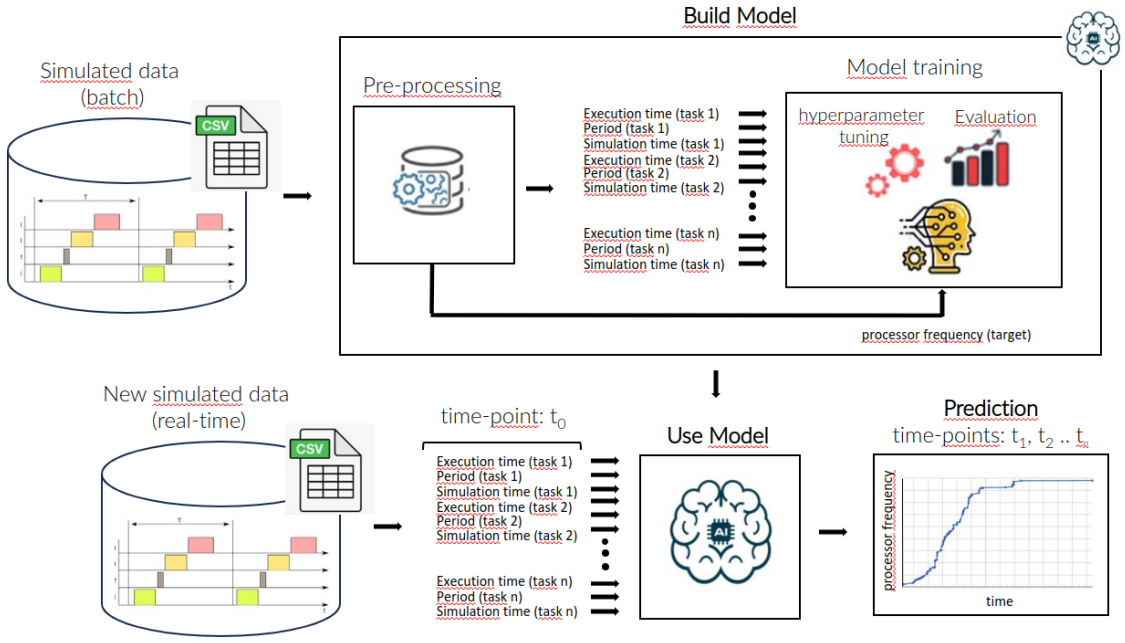

Figure 14 AI module containing the prediction methods

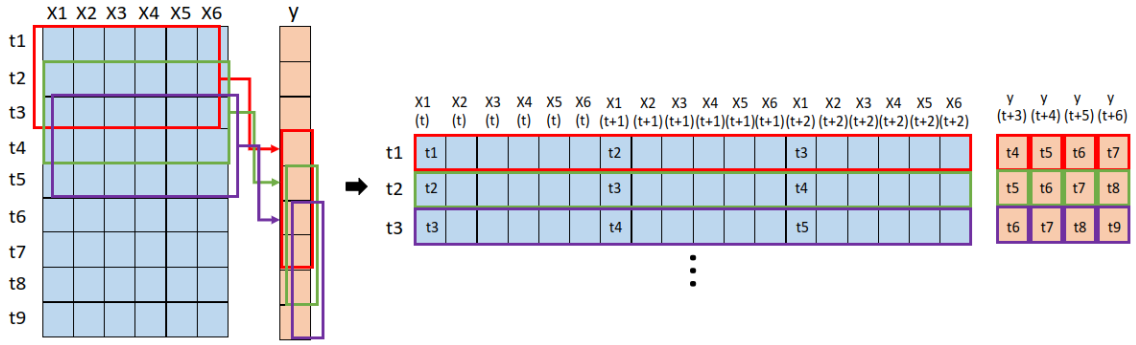

Figure 15 Example of transformation of the input data matrix using a sliding window of size n=3 (3 time-points) to predict the following four-time instants. X is the input data where each row of the matrix corresponds to the data of a time point (t), and y is the

5.2.3 Positioning Monitoring for Industrial Environment

The data monitoring and infrastructure designed and implemented as reported in Section 3.2 enables the collection of different types of monitored data via several interfaces. In the Vasteras May 11th demo, the Ethernet interface support was demonstrated, and current work is to support a low-power interface. In the same demo session, the capability to manage those data via a local database at the edge (i.e., at the gateway) and transfer/synchronisation with the cloud. Moreover, the capability to analyse that data, i.e. to perform anomalies analysis at the gateway was shown. This is of great utility, as it makes it possible, for instance, to decide the synchronisation of edge data with the cloud, i.e. sensing the raw data as is to the cloud, in case of anomalous operation detection. However, if it is assessed that these port elements are operating in normal conditions, the sending/synchronisation of data at lower rates, or mere sending of statistical metadata can be decided, with the derived communication, storage and energy savings

More specifically, the port use case addressed enables several types of data collection on the crane environment. This has an immediate utility in the detection of anomalies in their operation and their optimization. A specific focus is put on positioning data, for which specific novel sensing infrastructure is being designed, which will allow the exploitation of the potential of continuum computing architecture.

The following figure (Figure 16) provides a zoom into the integrated data model regarding positioning and its related quality.

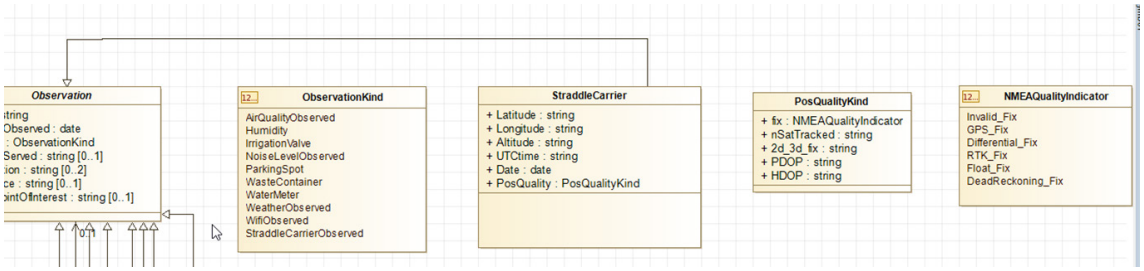

Figure 16 Detail of the data model of cranes position dada and its quality

This model reflects in much an abstraction of a Modbus interface compatible with other positioning IoTs already used by Prodevelop in the past. This abstraction is useful in facilitating the compatibility with previously installed infrastructure, the analysis tools and the presentation tools. As illustrated in the following figure (Figure 17), the abstraction fulfils different integration scenarios.

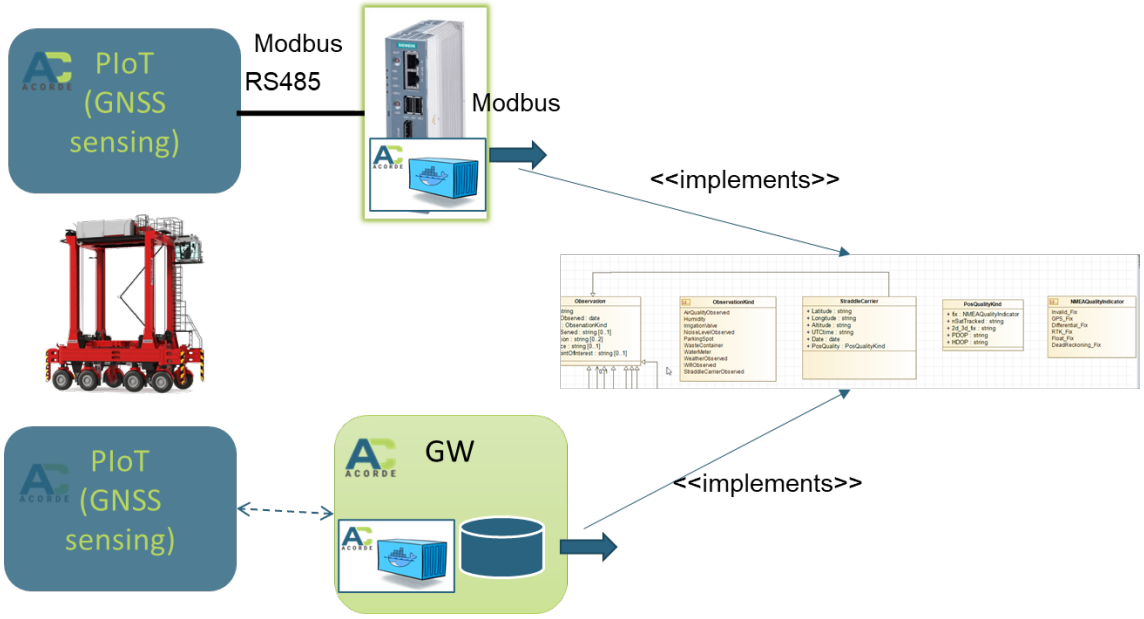

Figure 17 Data model on the positioning & quality data supports several integration scenarios

In the scenario shown in the upper part of Figure 14, the PIoT is connected to a pre-existing industrial Gateway and provides the information through a RS485 wired interface and a Modbus protocol. This is the scheme posed for the first physical piloting agreed upon with Prodevelop. Under this scheme, Prodevelop can access the information on the Modbus format, which facilitates the integration of novel PIoT data on their pre-existing edge infrastructures, relying on commercial SoA industrial gateways.

At the same time, the data model is sufficiently abstract to serve the edge solution developed by ACORDE in AIDOaRt, represented at the bottom part of Figure 14, which includes a novel gateway infrastructure. On that solution, ACORDE provides the PIoT and the Gateway platform, enabling low-power, long-range wireless connection of PIoT. Moreover, in this solution, the monitoring centre, ie. Prodevelop, can integrate the data directly by accessing the edge database. Notice that, in this scenario, there is no need for Modbus protocol intervention, while the defined data model ensures a base homogeneity and format on the information available, always convenient, for the analysis and data representation tooling.

With regard to the data model of the configuration of the Generic Anomalies Analysis (GAA) presented in Figure 2 of Section 3.2, it has several applications. First, that model is expected to be a basis for discussing, generating and adapting to a broad (eventually standard) interface for anomaly analysis tools. Moreover, it shall serve as a schema which enables automated validation of the configuration syntax, consistency and completeness.

5.3 Concluding remark

Data related labour reported here are just an insight of the potential that can be expressed once all solutions become fully integrated in use cases.

Additional collaborations as well as the potential exploitation of the mega-modelling capabilities may encompass horizontal integration of data related experiences, meaning the reutilization of experiences like for example those with the formalisation of the data models, on one use case, or the concrete application of solution to use cases, into another one. These situations, once discovered, will be reported in the next deliverables of the integration work package (WP5)

6 AIDOaRt Data Mega-Model

The main objective of the Data Representation component is to provide a common, agreed-upon, global data representation to serve as the foundation for MDE-based activities throughout the AIDOaRt framework. To achieve this, a mega-model has been defined to allow providing such global data representation so that it can be accessed and reused in all stages of the AIDOaRt process. The mega-model has also been built with the aim to take advantage of current state-of-the-art MDE techniques and scalable model storage technologies.

The AIDOaRt mega-model unifies the different UC data models by normalising their notation and linking those concepts that are (partially) synonymous across the various concrete data models. It yields generic representations for those elements that are shared across these UC data models, thus providing an agnostic overview of the most relevant data elements present in the AIDOaRt framework and its development process.

In this section, we present the whole process and design of the AIDOaRt mega-model. In subsection 6.1, we introduce the goals and expected benefits of having a mega-model for the AIDOaRt framework, and its structure. Afterwards, we outline the approach we followed to design the mega-model in subsection 6.2. As a result, in subsection 6.3, we describe the UC partners' specific data models in their current version, now following the UML notation; and in subsection 6.4, we present the different areas of the mega-model and their corresponding generic diagrams. We finalise with a call to action for partners to use the mega-model to identify common interests or areas of expertise and collaborate.

6.1 Definition, Goals and Expected Benefits of the AIDOaRt Mega-Model

The mega-model is an aggregation of all the UC domains that unifies their notation and connects semantically similar elements. It also provides a generic, UC-agnostic representation of concepts that are managed throughout the AIDOaRt development process. Figure 18 depicts the two-layers structure of the mega-model. The "blue" layer illustrates the different UC data models that now follow a standard notation; whereas the "orange" layer portrays the abstract elements that generalise UC-specific shared concepts and links them.

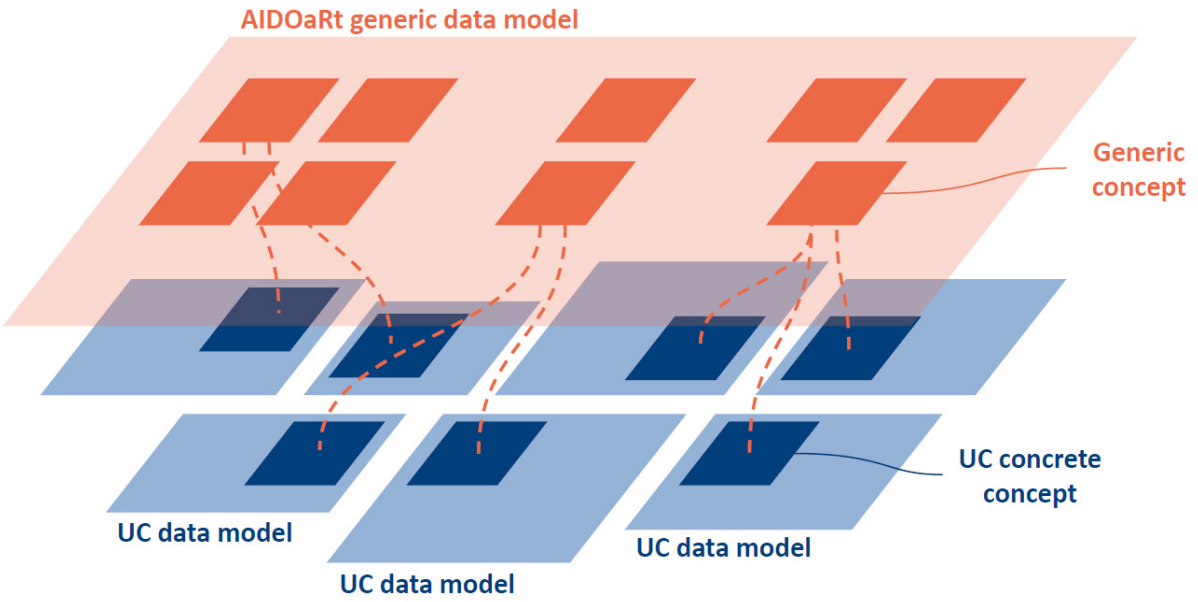

Figure 18 The two layers of the AIDOaRt mega-model

The benefits from this approach are many-fold.

First, by adopting a common formal language for all UC data models, namely the UML notation, we standardise the way data and their structure are designed. Having a lingua franca allows users to clearly appreciate what is represented, speed up the interpretation of the particularities of each domain, and therefore have a much better and common understanding. A standard notation fosters communication and collaboration between the users, facilitating better knowledge sharing. We adopted UML for data representation because of its ability to perform formal verification of data models and the wide support from solution providers (such as Modelio, which is the centrepiece of our MBRE method [AIDOART-D1.4]).

Second, our proposed mega-model can be used to foster synergies between different UCs from various application domains. In fact, the links between generic elements and concrete elements are aimed not only to state the generalisation relationship from specific UC data entities to their resulting AIDOaRt abstract entity but to be used as bridges between different UC domains. Those connections unveil synergies between UC domains, thus identifying potential collaborations. Given a generic element that abstracts two concrete concepts from different UC domains, stakeholders from one UC domain could establish a collaboration with stakeholders from the other connected domain. In this collaboration, they can share their expertise and lessons learned, and apply similar strategies, tools or techniques to common problems. Furthermore, this approach would also allow Solution providers to better understand the potential of their tools, in terms of transversal application with respect to different domains. Similarly, a UC partner could review another UC partner's data model and identify data elements that could be incorporated into their originally provided data model and enrich it.

The generic layer of the AIDOaRt data model bestows users a bird's eye view of the main concepts that are managed across the different use cases. We expect it to be extremely useful for new users of the framework, especially external users outside the AIDOaRt consortium and facilitate their onboarding.

A newcomer could easily comprehend the generic concepts and their relationships, and, if interested, navigate to a specific domain using the links of the generic elements to the concrete ones.

6.2 Approach to Design the AIDOaRt Mega-Model

In this section, we describe the collaborative activities performed in T2.2 to finally create the first version of the AIDOaRt mega-model. This process is illustrated in Figure 19, where the activities are positioned according to the deliverables of WP2. Blue activities correspond to the concrete UC data models and were mainly completed by UC partners. Orange activities were performed to create the generic layer of the mega-model and were executed by the T2.2 Leader with the support of stakeholders from all consortium members.

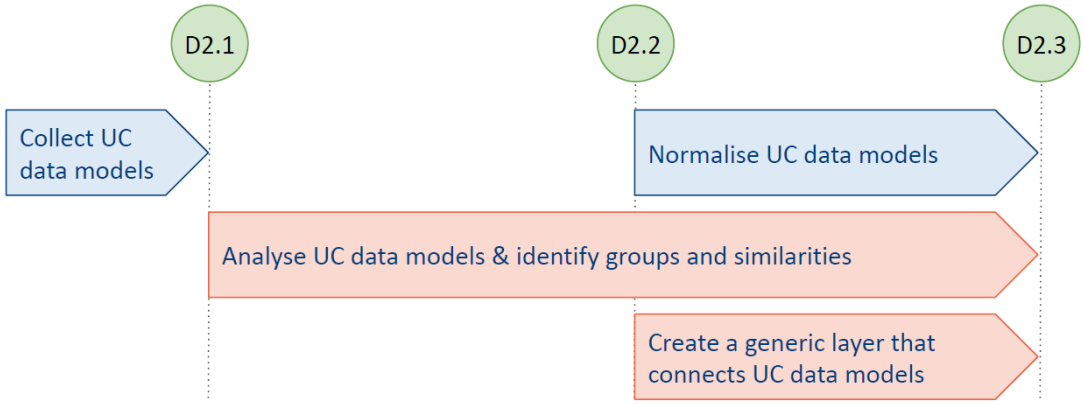

Figure 19 The process to create the AIDOaRt mega-model

6.2.1 Collect UC Data Models

In deliverable D2.1 (Data Collection and Representation - initial version), UC partners provided their data models, in order to represent both: (a) the domains of their use cases; and (b) the information that conforms to data requirements in regard to particular data types or format. Some of these models were described with diagrams, but, in some cases, the information that was required to be represented was clear and structured, whereas, in other examples, the particular fields of information required were still under progress.

Moreover, each UC partner adopted their particular notation, resulting in a variety of visual representations that might hinder a user of the framework from easily understanding the different domains, and grasping the information that is managed across use cases. Figure 20 illustrates the diversity of notations used in the different UC data models. Additionally, we can see that some notations used in the diagrams were not formal, thus impeding the application of an MDE-based method.

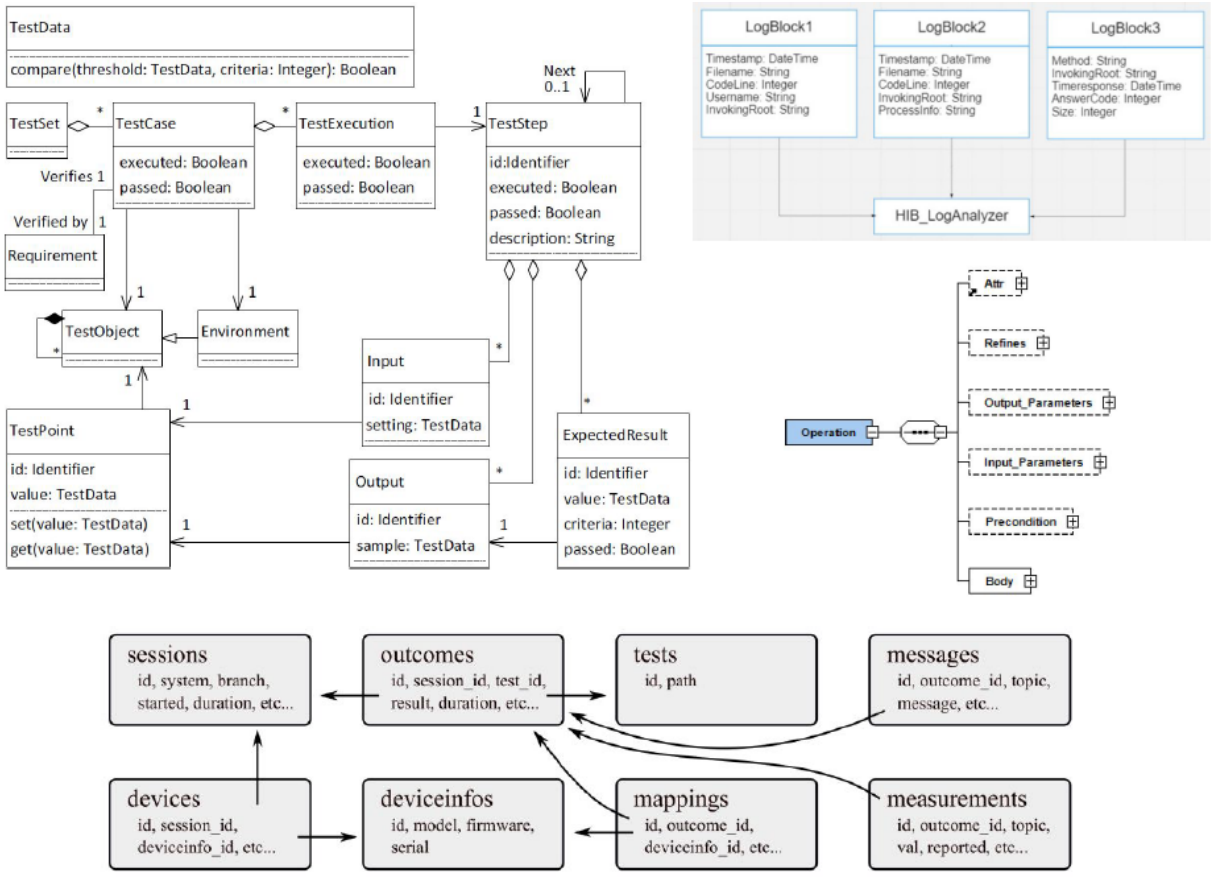

Figure 20 Some of the UC data models initially collected, and their diverse notations

Another issue that comes from this early collection of data models in such a variety of formats is that some partners may have similarities in their domains and actions, but in this arrangement, it was difficult to connect the dots between their respective data elements.

We addressed all these issues in the subsequent activities.

6.2.2 Analyse UC Data Models

Taking into account the UC data models initially provided, we conducted an activity of identification and generalisation of synonyms from UC data models. We performed an analysis of the diverse domains to first group the UC data models into different categories with a correspondence to AIDOaRt phases, and second identify the potential synonyms. The first version of this grouping was reported in D2.2 (Data Collection and Representation - interim version) but has been continuously carried out until this final deliverable. It helped to identify the generic elements and to structure the generic layer of the mega-model into sub-diagrams, one for each respective AIDOaRt phase.

6.2.3 Normalise UC Data Models

In order to have a common notation and its many benefits, partners were requested to adopt UML for their data models and to publish their diagrams in Modelio. To complete this activity, we relied on the proficiency of the stakeholders for their respective domains. Regarding the UML modelling skills, when required, we scheduled workshops to assist those partners that had little or no experience in identifying data elements and representing data models; or, in other cases, the partners themselves

leveraged the modelling task to the expertise of their solution providers. The resulting assets are reported in this deliverable.

6.2.4 Create the Generic Layer

In the last phase of the task T2.2 Data Representation of WP2, we constructed the generic layer and structured it according to the AIDOaRt development process phases. We defined the generic entities and connected them with the UC-concrete ones. The initial draft of this connecting layer and its artefacts was presented to, refined and validated by UC partners, resulting in the first version of the AIDOaRt mega-model that is presented in this document.

6.3 Use Case Data Models

6.3.1 ABI

In this section, we present the data model of ABI.

This is the UML class diagram of the Use Case Requirements. Each instance of Use Case Requirement is identified by an ID and a description, as well as by two enumeration types, the RequirementType, to specify if the requirement is functional or non-functional, and the RequirementPriority, to specify the priority of the requirement (from Highest to Low). A Use Case Requirement can refine a Customer Requirement and an AIDOaRt Generic Requirement. Also, a Use Case Requirement can be derived from another Use Case Requirement. Each instance of Customer Requirement is identified by an ID and a description. The set of custom requirements is further refined to define the Requirement Matrix.

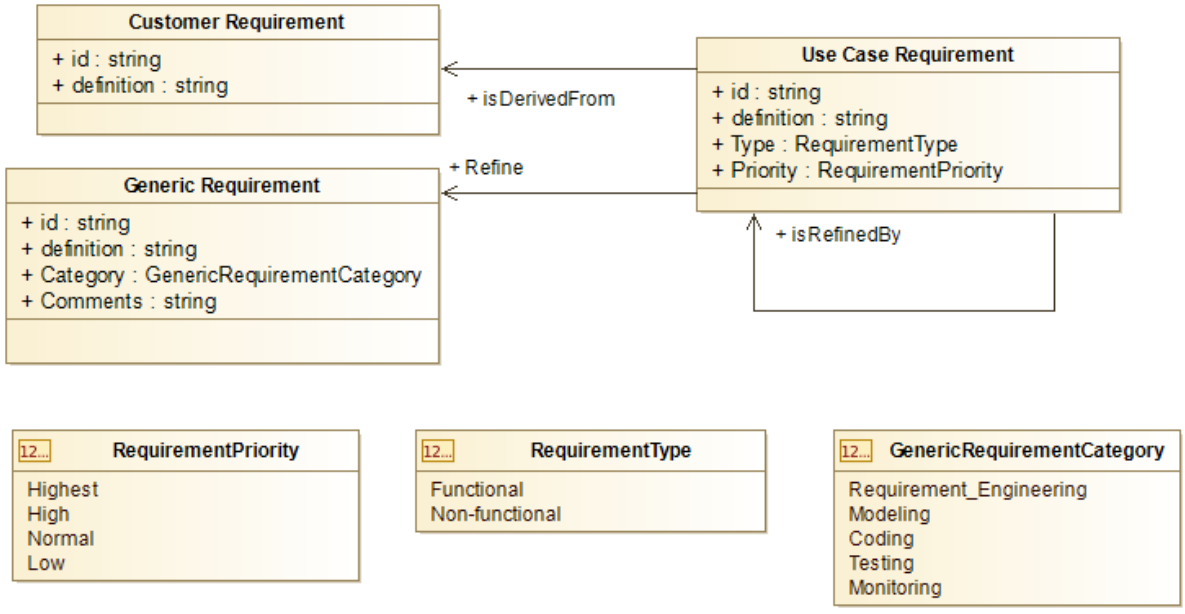

Figure 21 ABI UC Requirements Data Model

Customer Requirement

High-level requirement from the customer.

The Customer Requirement is expressed as specifications in natural language and generally comes through simple Word files or PowerPoint presentations, but they could also come as verbal requests during a meeting.

Generic Requirement

AIDOaRt Generic Requirement

Use Case Requirement

Use Case Requirement, defined starting from the customer requirements keeping in account also any applicable standards and technical assumptions. They are expressed as specifications in natural language.

6.3.2 AVL

In this section, we present the data models of AVL.

6.3.2.1 ODP Data Model

This data model captures the most important entities, associated properties and interrelations for the ODP use case. Specifically, the entities capture the data objects relevant for product verification or validation in automotive development projects.

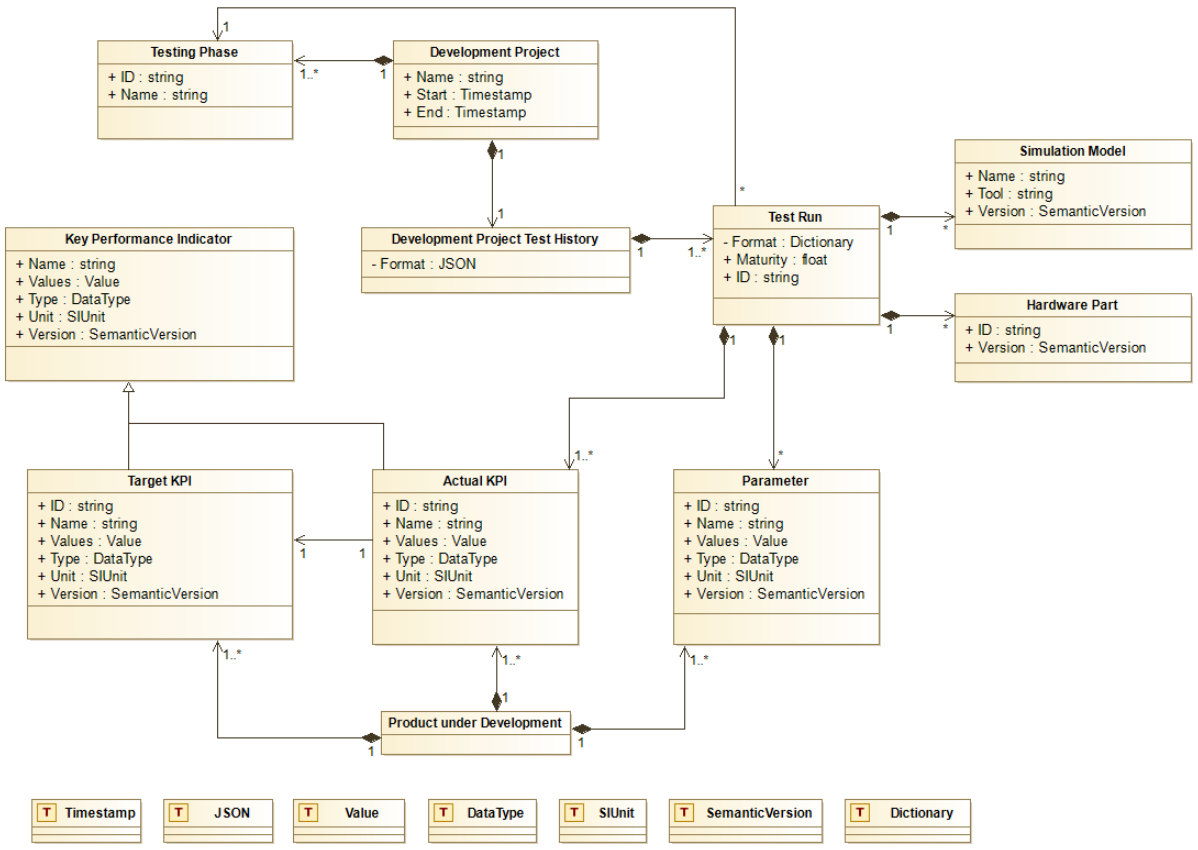

Figure 22 ODP Data Model

Development Project

Project covering the development of a product in the automotive domain. The product can be a car or a component of a car, e.g., its battery.

Testing Phase

In the automotive domain, the testing of the product under development is structured into several phases. E.g., at the beginning of the development project, testing will be in a simulation phase, while towards the end of the development project, testing will be done in a hardware phase when hardware prototypes of the product under development are available.

Testing phases can be both sequential and simultaneous. For example, testing the performance of a car is done in Phase 1a using simulations and Phase 1b using the hardware powertrain on a hardware test bed. Simultaneously to phases 1x, phases 2x might deal with testing another aspect of the car, e.g., its energy consumption.

In each phase, individual test runs are executed.

Key Performance Indicator

A property of the product that is of relevance to some stakeholder (customer, legislation etc.). Usually, a numerical value (scalar or n-dim matrix) with some physical meaning (and hence a unit). Each KPI is associated with a target KPI.

Example: The product "battery electric vehicle" has the KPI "driving range" (scalar, unit kilometres).

Development Project Test History

At any given point in time in the development project (between the start and end date), certain test runs (simulations, physical tests with hardware, or mixed tests using simulation and hardware combined) have already been executed. These already executed test runs are collected in the development project test history.

Test Run

The aim of the test run is to determine the values of a certain KPI or set of KPIs for the current development state of the product under development, i.e., the actual KPIs. These actual KPIs are compared to target KPIs in order to assess whether the current state of development can meet the target KPI (aka satisfies the corresponding requirements).

A test run comprises either executing a simulation model or testing a hardware part (or a combination of simulation and hardware).

A test run has a maturity associated with it. Maturity is a numerical measure of confidence, similar to an uncertainty of a physical measurement value. At the beginning of the development project, when much of the product under development is still unclear, the maturity will be low. At the end of the development project, when understanding of the product under development is detailed and precise, maturity is high.

Simulation Model

Representation of some functional/behavioural aspect of the product under development, created in a specific modelling tool (Simulink, Amesim etc.). Usually, changes in the course of the development project, hence has a version associated with it.

Utilised by a test run to compute some KPI of the product under development. Takes the product design (at a specific point in time, encoded by parameters) as input and produces actual KPI as output.

Example: A simulation model of the electrical performance of the battery uses the number of battery cells (as specified in the development status at a specific point in time) as input and determines charging time as output.

Hardware Part

Hardware prototype of the product under development (or a component of the product under development).

A test run uses the hardware part to determine an actual KPI.

Several (increasingly mature) hardware prototypes are usually created in the course of the development project, hence a version.

Example: Hardware prototype of a battery.

Product under Development

The product under development itself, i.e., the thing that is created in the course of the development project.

The product under development must meet certain target KPIs.

The product under development (its design) is encoded by parameters.

The development status of the product under development at a specific point in time will lead to actual KPIs specific to that development status.

Target KPI

The target value of some KPI, as defined in a product requirement. At the end of the development project, each KPI must satisfy the target KPI.

Actual KPI

A test run uses the development status (of the product under development, available at the time of executing the test run) to determine the KPI value for this specific development status. This KPI value is specific to a development status and is called the actual KPI.

Actual KPI is associated with one target KPI.

Actual KPI usually changes over time due to design changes of the product under development.

Parameter

A characteristic of the product under development that encodes the design of the product. Usually, a numerical value (scalar or n-dim matrix) with some physical meaning (and hence a unit). The value of a parameter at a specific point in time in the development project is determined using the development status of the product under development at this point in time.

The value of the parameter usually changes over time due to design changes of the product under development.

Example: Mass of the vehicle in kilograms, computed from the mechanical design of the vehicle (geometry, material selection) at a specific point in time.

Test run uses parameter values to determine actual KPIs for the state of development as encoded by the parameters.

6.3.2.2 SEC Data Model

The overall idea is to facilitate anomaly detection on a high-security Powertrain CAN network (PT CAN) inside a vehicle and to use a Fuzz testing unit (CAN Fuzzer). Then we generate arbitrary traffic on a low-security Infotainment CAN network (Info CAN) and evaluate if we can observe anomalies on the PT CAN. The goal of the process is to validate and verify the correct function of a security gateway device

(GW) that is connecting those two CAN networks. Having such a GW connecting two (or more) CAN networks in a vehicle is a very common setting in the automotive industry. The GW should let needed information pass and block all other information, especially to prevent unauthorised access from the Info CAN to the PT CAN. The PT CAN anomaly detection should work as a test oracle if fuzzed input on the Info CAN will have a potentially undesired impact on the PT CAN.

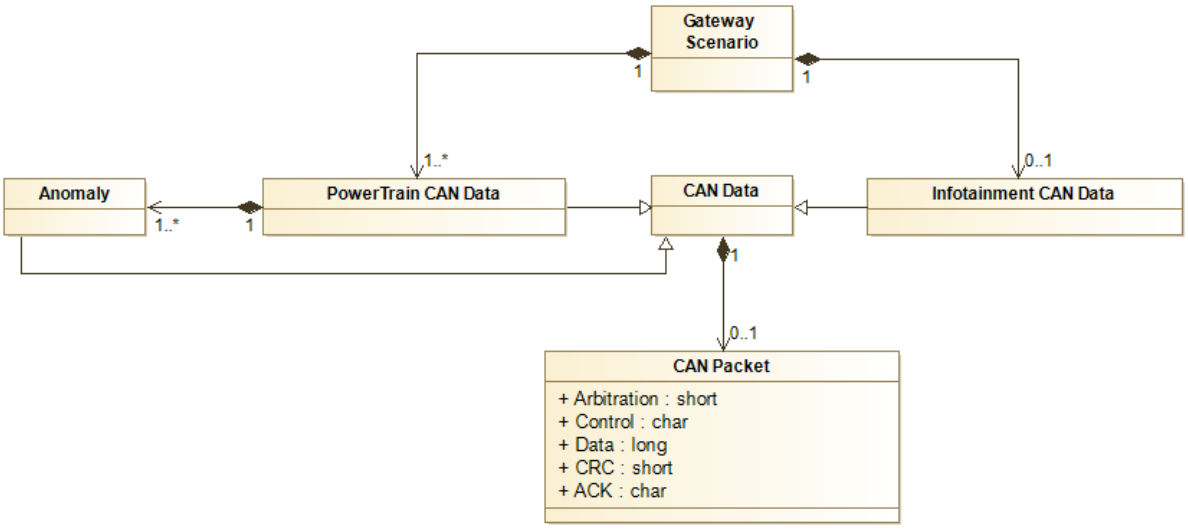

Figure 23 SEC Data Model

Gateway Scenario

Control class that would start the fuzzer and run a test.

PowerTrain CAN Data

This is CAN data collected on the Powertrain CAN network (PT CAN) of a vehicle. This kind of data is recorded previously with a set of normal behaviour during a training phase and later analysed during the testing phase for anomalies using machine learning (while the fuzzer generates data on the Info CAN).

Infotainment CAN Data

This is CAN data collected on the Infotainment CAN network (Info CAN) of a vehicle. The purpose of this data is to get initial seed data for the CAN Fuzzer, which should generate Info CAN data itself for the actual test.

CAN Data

This is a collection of CAN or CAN-FD Frames (see respective class), representing actual traffic running over a CAN bus [ISO-11898-1] inside a vehicle. The data is collected on two different CAN buses inside a vehicle: a low-security Infotainment CAN network (Info CAN) and a high-security Powertrain CAN network (PT CAN). On the Info CAN, a CAN Fuzzer also generates data.

CAN Packet

This class represents a CAN or CAN-FD Frame running over a CAN bus, according to the ISO 11898-1 [ISO-11898-1] specification. The data fields of the class represent the parts of a CAN frame according to the specification.

Anomaly

An anomaly is CAN data that could occur on the PT CAN and that is different from previously recorded. If CAN data represents an anomaly or not is determined via a machine learning algorithm.

6.3.2.3 TCV/MBT Data Model

The provided TCV/MBT data model is used in the AVL toolchain to test and validate ADAS/AD functionality. It consists of three logical units: (1) data generator of the test scenarios, (2) data generated by the scenario execution and (3) actionable data generated by the next test scenario.

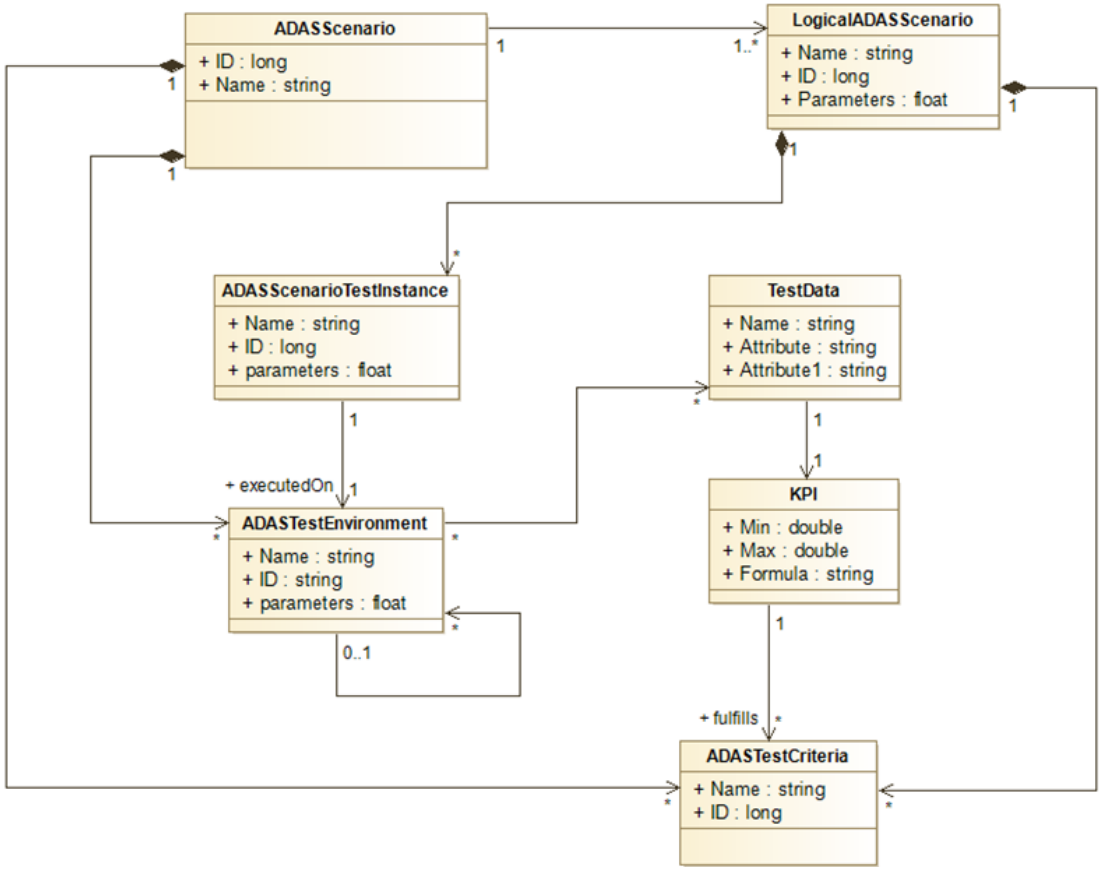

Figure 24 TCV/MBT Data Model

ADASScenario

Defines a database of logical scenarios that are used for testing ADAS/AD functionality. When triggered provides a logical scenario for testing.

LogicalADASScenario

Logical scenario parameters, taxonomy and ontology of the logical scenario under test. When triggered, it provides an instance of the logical scenario for testing.

KPI

A collection of data generated after calculating the test-relevant KPI.

ADASScenarioTestInstance

A scenario under test that is defined by the test parameters, and concretised by the test values and ranges, as well as the corresponding ontology.

ADASTestEnvironment

Collection of the data defining the simulation environment and data generated in an interaction with the test function.

TestData

A selected collection of the Environment data relevant for test evaluation.

ADASTestCriteria

A collection of data that defines a new test action based on the provided KPI values.

6.3.2.4 RDE Data Model

This class model describes the relationship between data in the RDE use case. It features several parallel structures corresponding to the different stages of the UC: First a driver model is learned from a set of real-world measurements. This model is then used to generate possible driver behaviour for new routes which requires a slightly different input format.

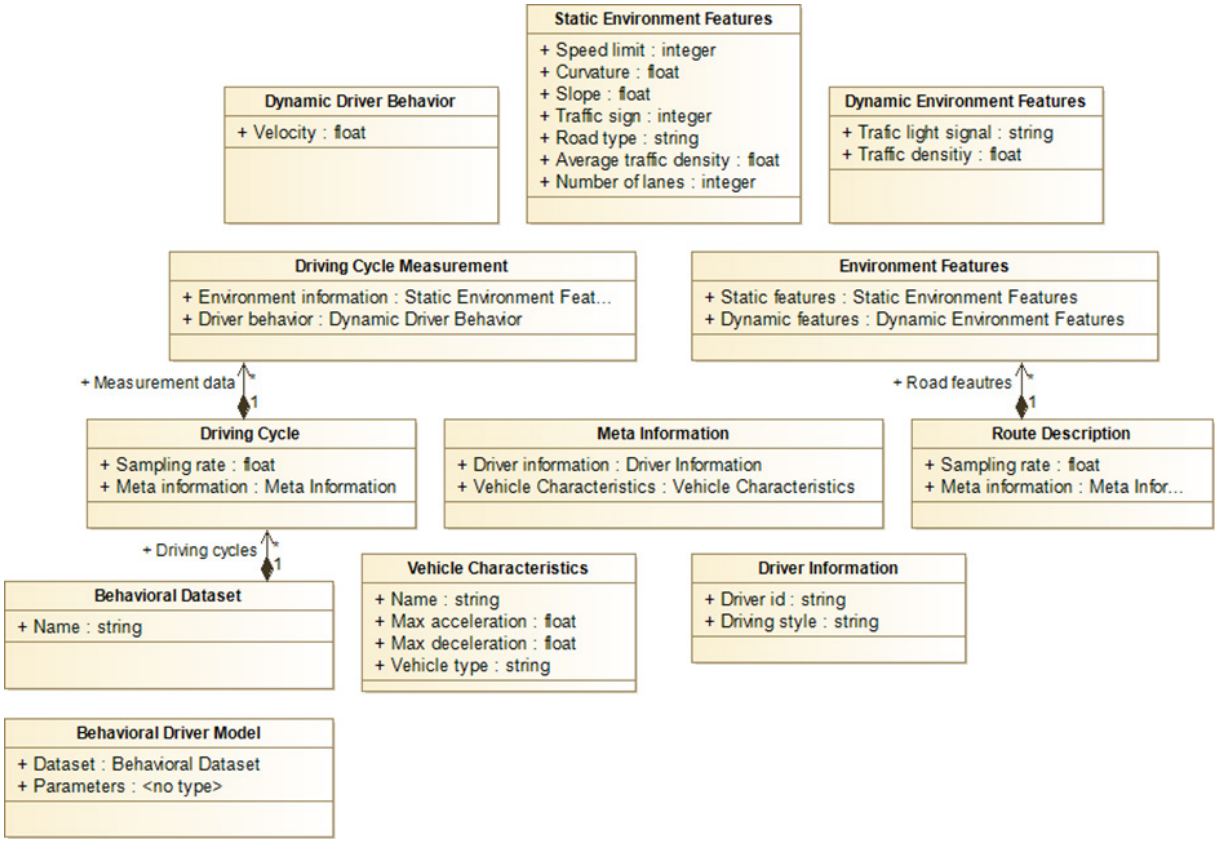

Figure 25 RDE Data Model

Dynamic Environment Features

The Dynamic Environment Features provide the dynamic details on the road, such as information about other traffic participants, and signals, such as traffic lights, which are commonly not measured on test drives.

Static Environment Features

The Static Environment Features describe the static (i.e. unchanging) aspects of a point on a road. This is mainly given by the road infrastructure but can also include traffic statistics. This could also include other values not listed here.

Driver Information

This includes meta-information about the driver and could be extended to include other attributes as well.

Dynamic Driver Behavior

Dynamic Driver Behavior describes an instantaneous snapshot of the driver behaviour. This is commonly just the velocity but could also include more fine-grained information, such as how much the gas and brake pedals are pressed.

Driving Cycle

A Driving Cycle is the result of a test drive along a given route during which the driver's behaviour is continuously recorded using a fixed time step. It also includes meta-information about the car and the driver.

Route Description

This is a sequence of environment features describing a route which also includes dynamic information about the situation on the road. This serves as an input to the behavioural driver model from which a new velocity profile is generated.

Driving Cycle Measurement

A Driving Cycle Measurement is a composition of dynamic driver behaviour and static environment features.

Environment Features

This information combines static and dynamic environment features, thus giving a detailed view of a situation on the road.

Vehicle Characteristics

The Vehicle Characteristics give important details which can be used to ensure that the behavioural model only generates velocity profiles that are actually possible using the specified car.

Meta Information

This is an aggregation of vehicle characteristics and driver information.

Behavioral Dataset

This describes a dataset which is used for generating a behavioural driver model.

Behavioral Driver Model

A Behavioral Driver Model describes a distribution of human driver behaviour depending on the situation on the road and can be used to generate driver behaviour (i.e. velocity profiles) for a given route description. A behavioural driver model is defined by a set of input data and a set of parameters.

6.3.3 BT

In this section, we present the data model of BT.

This diagram shows the concepts of the Alstom use case 2. In particular, it defines the classes involved in the parametrization process of the propulsion system.

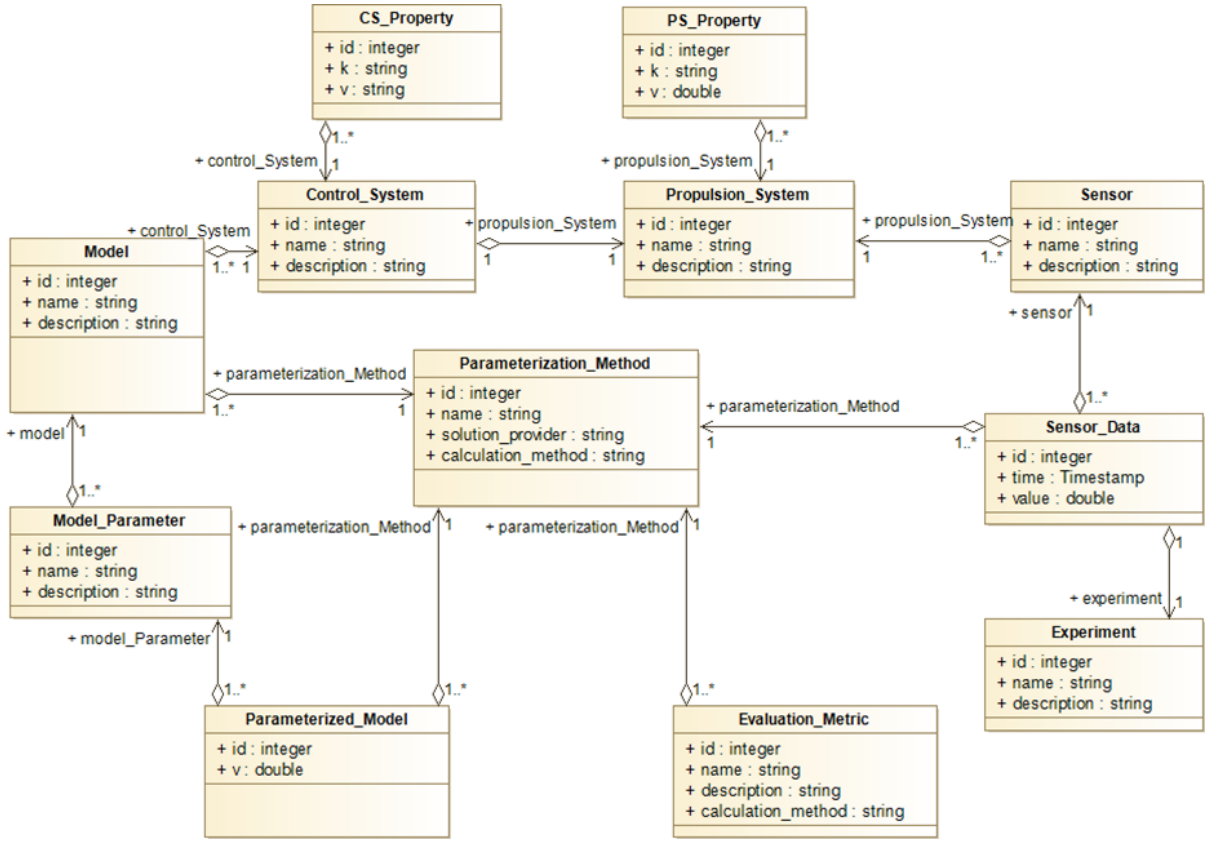

Figure 26 Automated Parametrization of Propulsion System Controller

Propulsion_System

Propulsion_System class defines the name and type of the propulsion system.

Control_System

Control_System class defines the name and type of the control system.

Sensor

Sensor class defines the sensors used to collect data from the propulsion system.

Sensor_Data

Sensor_Data class defines the data collected from the sensors in the propulsion system.

Model

Model class defines the models used in the control system.

Model_Parameter

Model_Parameter class defines the parameters of the models used in the control system.

Parameterization_Method

Parameterization_Method class defines the parameterization method that will be used to find the parameters of the models in the control system.

PS_Property

PS_Property (Propulsion System Property) class defines the properties of the propulsion system.

CS_Property

CS_Property (Control System Property) class defines the properties of the control system.

Evaluation_Metric

Evaluation_Metric class defines the metrics used to evaluate the parameterization method.

Parameterized_Model

Parameterized_Model class defines the results from the parameterization method.

Experiment

Experiment setup when collected data.

6.3.4 CAMEA

In this section, we present the data model of CAMEA.

Data Model for the AI-based reduced-power radar configuration approach generating Report based on original configuration and defined constraints.

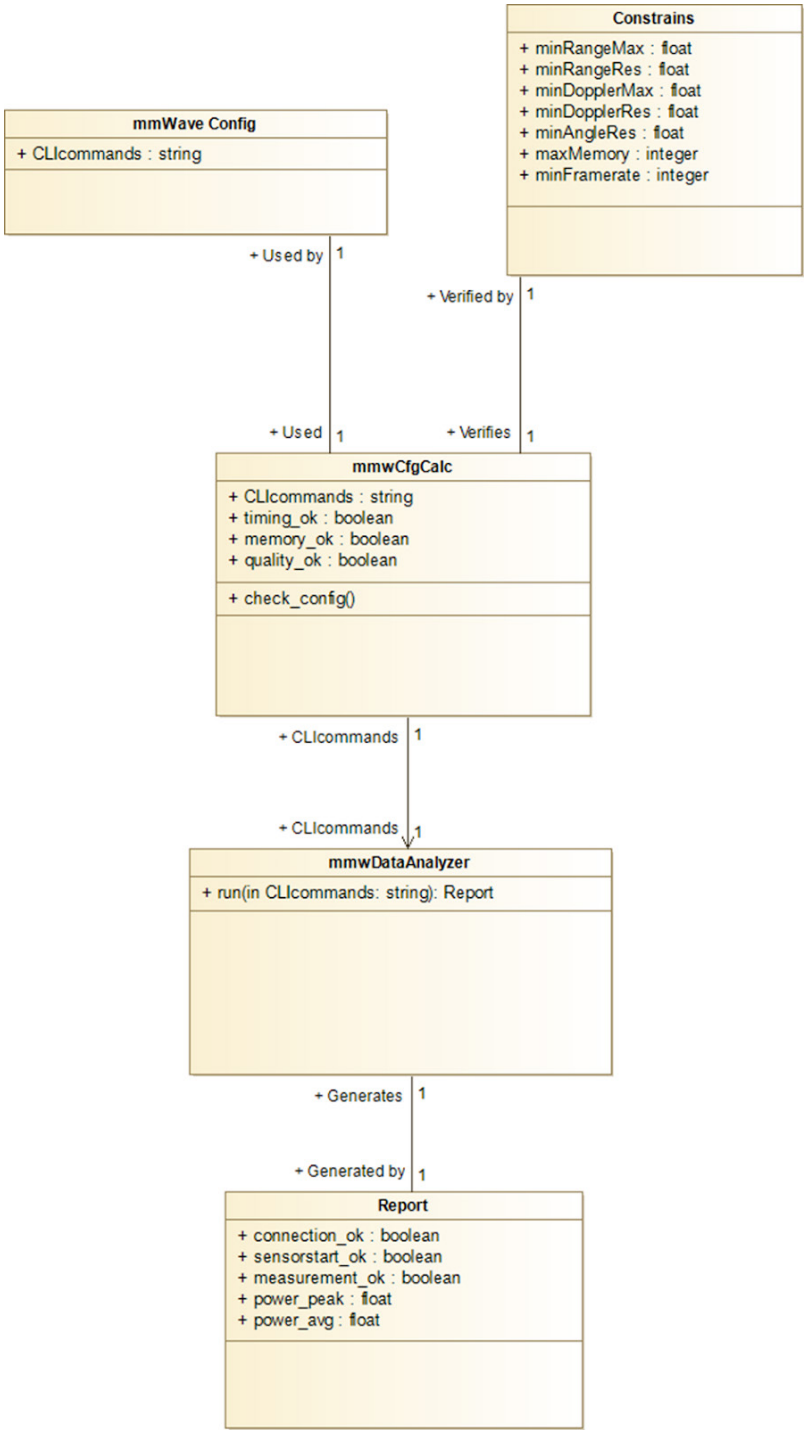

Figure 27 CAMEA Data model

Constrains

Set of constraints for radar configuration tuning.

mmWave Config

Original input configuration of radar sensor.

mmwCfgCalc

mmWave configuration class.

Report

Report class as result of executed test.

mmwDataAnalyzer

Communication application for radar sensor.

6.3.5 CSY

In this section, we present the data model of CSY.

This diagram describes the B theory of component and proof obligations. It starts with a project, made of components. Its compilation generates Proof Obligations that must be proved with proofs to demonstrate the correctness of the components and therefore of the project itself.

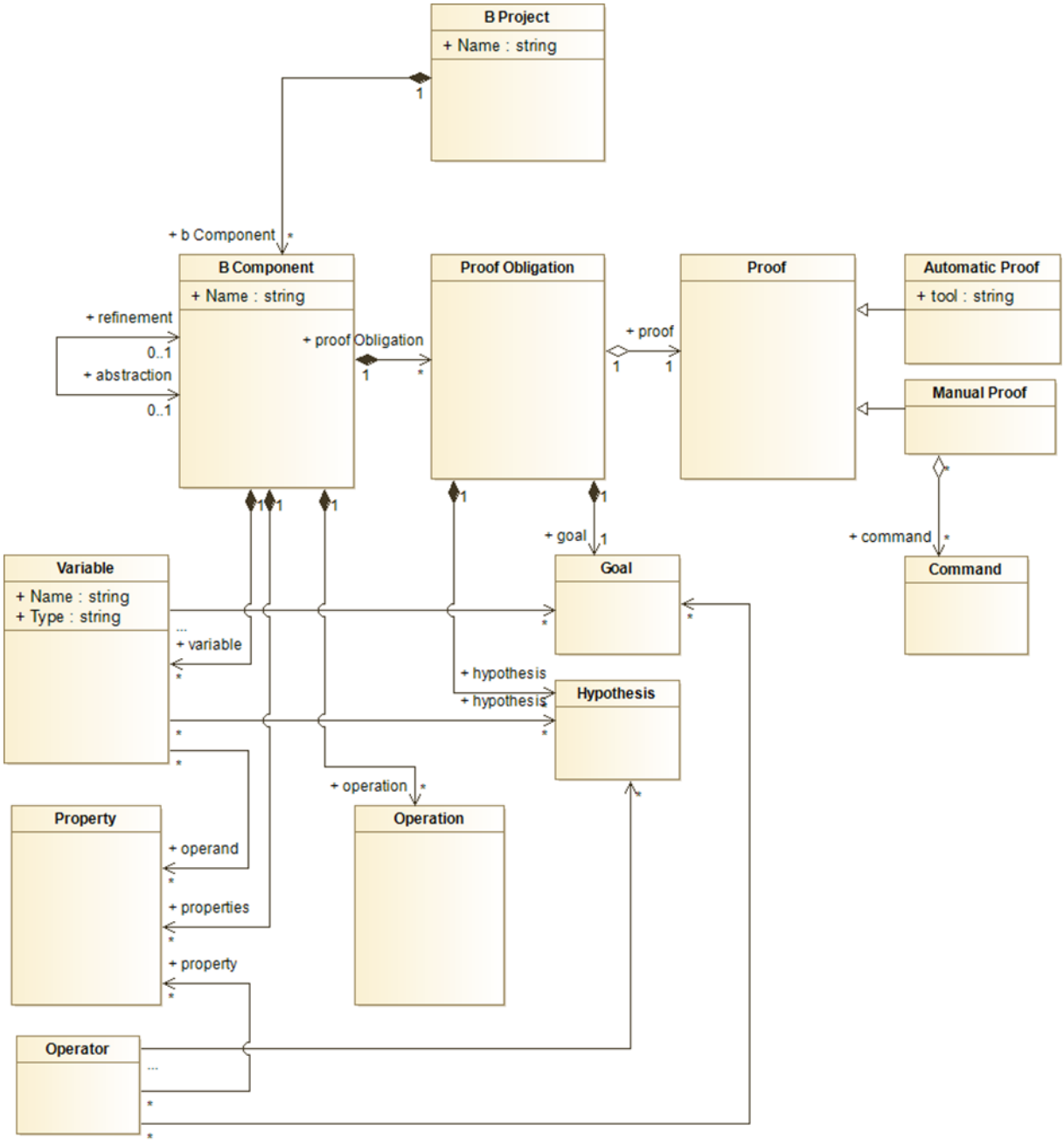

Figure 28 CSY Data model

B Project

A B Project describes the product that is worked on. It is basically a container for all other components. It can be parametrized with a file. It records its own state of completion.

The project has a proof status, meaning that whether it is proved or not, the proof status is part of the safety demonstration.

B Component

A B component is comparable to a class. It can have 3 flavours, a "machine" or "specification" is the highest level of abstraction. It describes the class on an abstract level. This allows the developer to use non-implementable descriptions, non-deterministic behaviours or unbounded values.

For example, a machine can describe an operation that sets the variable x to something in N (the naturals) or say that after 16 ticks, x will always be greater than 12.

On the opposite, a component can be an "implementation", an implementation will be forced to be deterministic and to use types that can be represented by a computer. It is written in an imperative style and can be directly translated to C, ADA or JAVA. In between, "refinement" is a component that refines another component, a machine, or a refinement, and can be further refined by an implementation.

Using several layers of abstraction is a way to give different levels of complexity and to set properties on the right level.

Refinements of components always come with a set of proof obligations, to demonstrate that the refinement is a correct transformation. If not, the proof will be impossible and the project will be unproven.

Proof Obligation

A proof obligation is a mathematical formula to be proven, in order to ensure that a B component is correct.

A proof obligation is a mechanic that demonstrates the correctness of the B model. Proof obligations are generated at every step of the creation of the model. At initialisation, there will be PO to check that properties hold with initial values. On refinement, there will be PO to connect abstract and concrete variables, and to verify that the refinement of an operation respects the guarantees set on a higher level. Also, on operations, there will be PO to verify that component properties are still valid after the transformation.

The B proof obligations B are self-sufficient, i.e., no implicit information must be used in their demonstration. All of the proof obligations use the following structure: H ⇒ P, where P and H are predicates. This formula means that it is necessary to prove the aim P by applying assumption H, H being generally a conjunction of predicates.

In B, the P predicate and some H assumptions are built by applying one (or more) substitution(s) to a predicate. As B is a mathematical language, the substitutions and the predicates considered are directly taken from the B source.

In this model, we call the predicates in H Hypothesis and P the Goal.

Proof

Proofs are demonstrations of proof obligations. They use mathematical transformation and reasoning like substitution, induction or recursivity to transform and reduce predicates to the truth or untruth statements. Transformation can be applied by tools called prover that use heuristics to choose the right substitutions and transformation rules or by a person that will apply commands associated with transformations based on experience and intuition.

Property

Properties of components are what link the variables together. They can be type-binding or logical relations. They are the main source of proof obligation through refinement or operations.

Variable

The variables can be abstract or concrete, depending on the component they are used. abstract variables can be elements of set or unbounded numerals, concrete variables can be literals or enumerate but always refers to an integer.

Operation

Operations are the transformations of the model described in the component. They map easily to imperative functions. Usually, a component will have an initial state that fixes the value of its variables at the startup and operation that change those values. Operations have preconditions that reduce their applicability and postconditions that give guarantees on the variables they modify. Proof obligations must ensure that the body of an operation applied to variables within its precondition will respect the postcondition.

Command

Commands are transformation steps to be applied on predicates in order to solve proof obligations.

Automatic Proof

A proof is automatic if it is completely solved by the action of a prover.

Manual Proof

A proof is manual if it is described by a sequence of commands usually written by a person.

Operator

Operators are mathematical relations that connect variables logically or arithmetically. They are the syntax of the B language.

6.3.6 HIB

In this section, we present the data model of HIB.

Data Model for the log storage in the HIB-logAnalyzer (HIBLA) application based on the logs from TAMUS. It comprises two classes that every log register has to instantiate and one optional class.

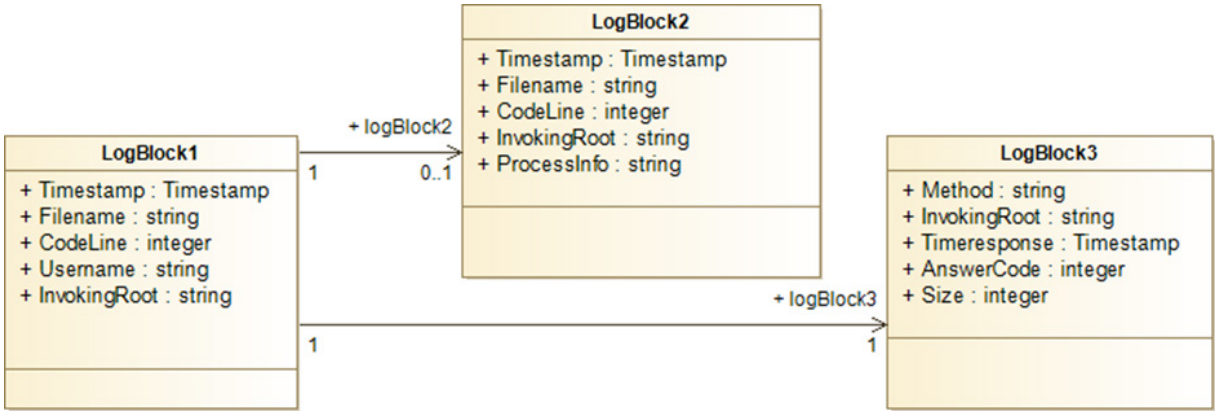

Figure 29 HIBLA Log data model

LogBlock1

Requested root and session information for each of the log registers. This is mandatory and is the 'parent' node of a given logged operation.

LogBlock2

Specific information of the processes invoked. This class is optional in all cases and is related to one LogBlock1 class that is the parent descriptor for each log block.

LogBlock3

Information related to the response given and the processing time for a given request. This is a mandatory class for each LogBlock1 instance.

6.3.7 PRO

In this section, we present the data models of PRO.

6.3.7.1 SPMP environmental info

This diagram describes the model of data that is exchanged via messages between the different cyber-physical systems (CPS). Most of the classes describe sensors that collect information on different parameters of the environmental conditions (weather information, air quality, noise pollution, etc.), others collect information on the status of some of the services available in the area of interest (parking, waste containers, wifi, etc.), and finally there are also classes that describe the information associated with different vehicles that are deployed in the work area (cranes).

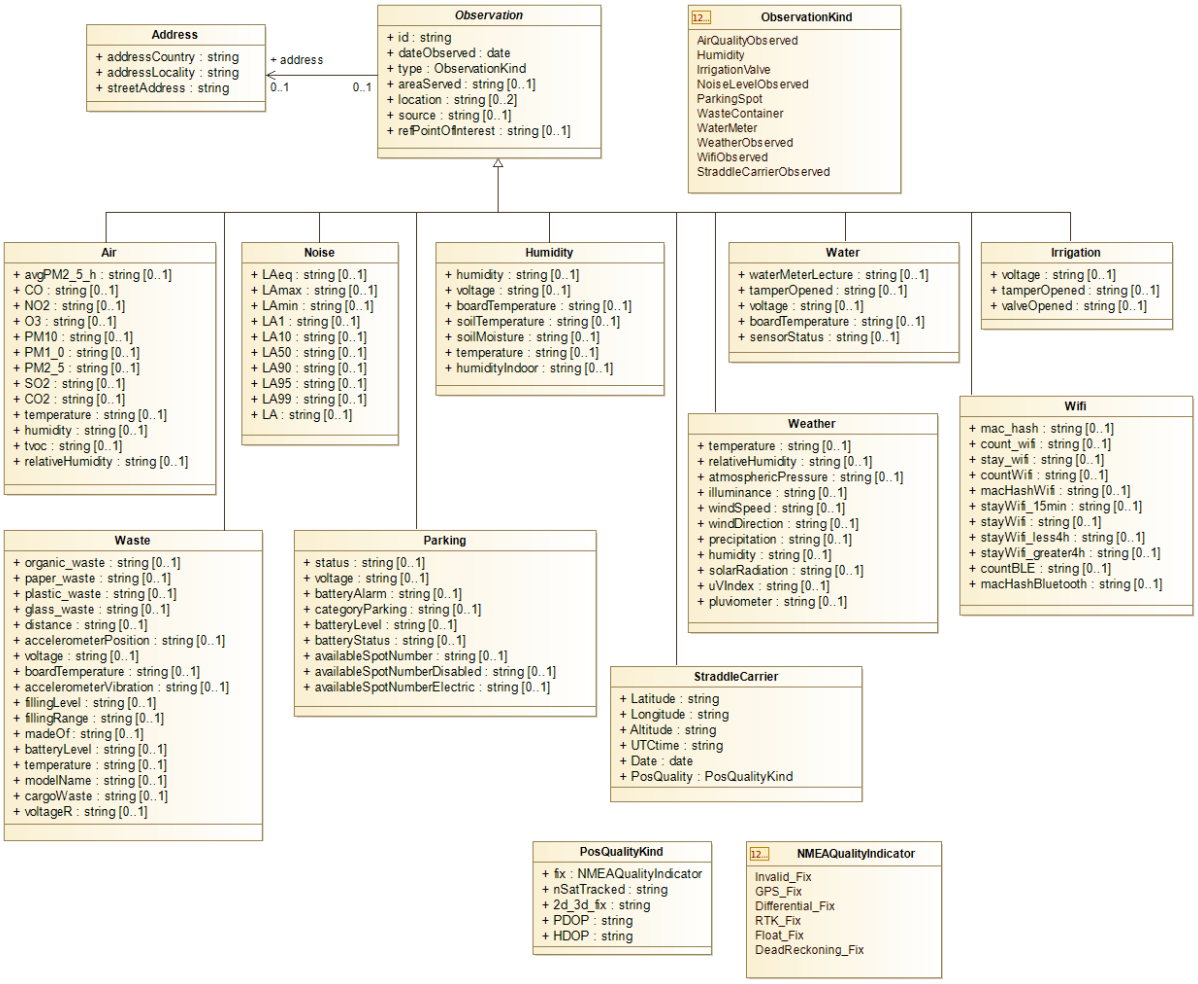

Figure 30 SPMP environmental info

Address

Information about the concrete location in string format. It is extracted from the "Observation class" and is used to filter the different elements according to their physical location.

Observation

This class is the central axis of the diagram because it is in charge of listing the different sensors that are deployed in the area of interest. Therefore, most of the schematic elements are interconnected with this class.

ObservationKind

Type of observed data: It is a list of the main parameters of the environment that are measured at each instant of time.

Air

Information collected by air sensors is then used to calculate air quality parameters in order to ensure minimum health conditions for the working environment.

Humidity

Information collected by a humidity sensor to collect humidity conditions both at atmospheric and ground level.

Noise

Information collected by noise sensors to ensure the quality of the environment in terms of noise level.

Parking

Information collected by a parking sensor to be able to control the parking areas individually by having information from each of the sensors in the parking spaces to be able to obtain the parking status.

Waste

Information collected by a waste sensor which provides information about the different types of waste containers: fill level, temperature, door status, etc.

Water

Information collected by a water sensor

Weather

Information collected by weather sensors to measure the main climatological parameters: temperature, pressure, wind direction and speed, solar radiation, etc.

Wifi

Information collected by a Wifi sensor that provides information on how many devices are connected and how long they are connected.

Irrigation

Information collected by an irrigation sensor used to control the irrigation valve services of the green areas in the area of interest.

StraddleCarrier

Information collected on a straddle carrier in the port environment. It includes position information (Latitude, Longitude and Altitude) at a specific time (UTCTime) and corresponding date (Date). It includes quality information related to that position information (PosQuality).

PosQualityKind

Class with attributes providing information on the quality of the positioning solution. The more important attribute is the "fix", which provides the fix-level information according to the NMEA

standard. Other attributes provide valuable information related to the reasons for degraded quality, the level of positioning information that can be considered (2D vs 3D), and to devise the level of accuracy that might be expected for a degraded quality.

NMEAQualityIndicator

NMEA standard based indication of the quality of the positioning solution. RTK fix should provide cm-level accuracy (as required for straddle carrier for the smart port use case). The remaining values refer to degraded levels of position quality.

6.3.7.2 Platform and Application Model

This UML diagram describes two high-level models for a cyber-physical system (CPS), and a set of associated definitions. The first model hierarchy (PlatformTypeDefinitions) contains elements that describe the hardware components of the CPS and how these components are connected to each other. The second-class hierarchy (Module) describes the application software components that are running on the CPS hardware.

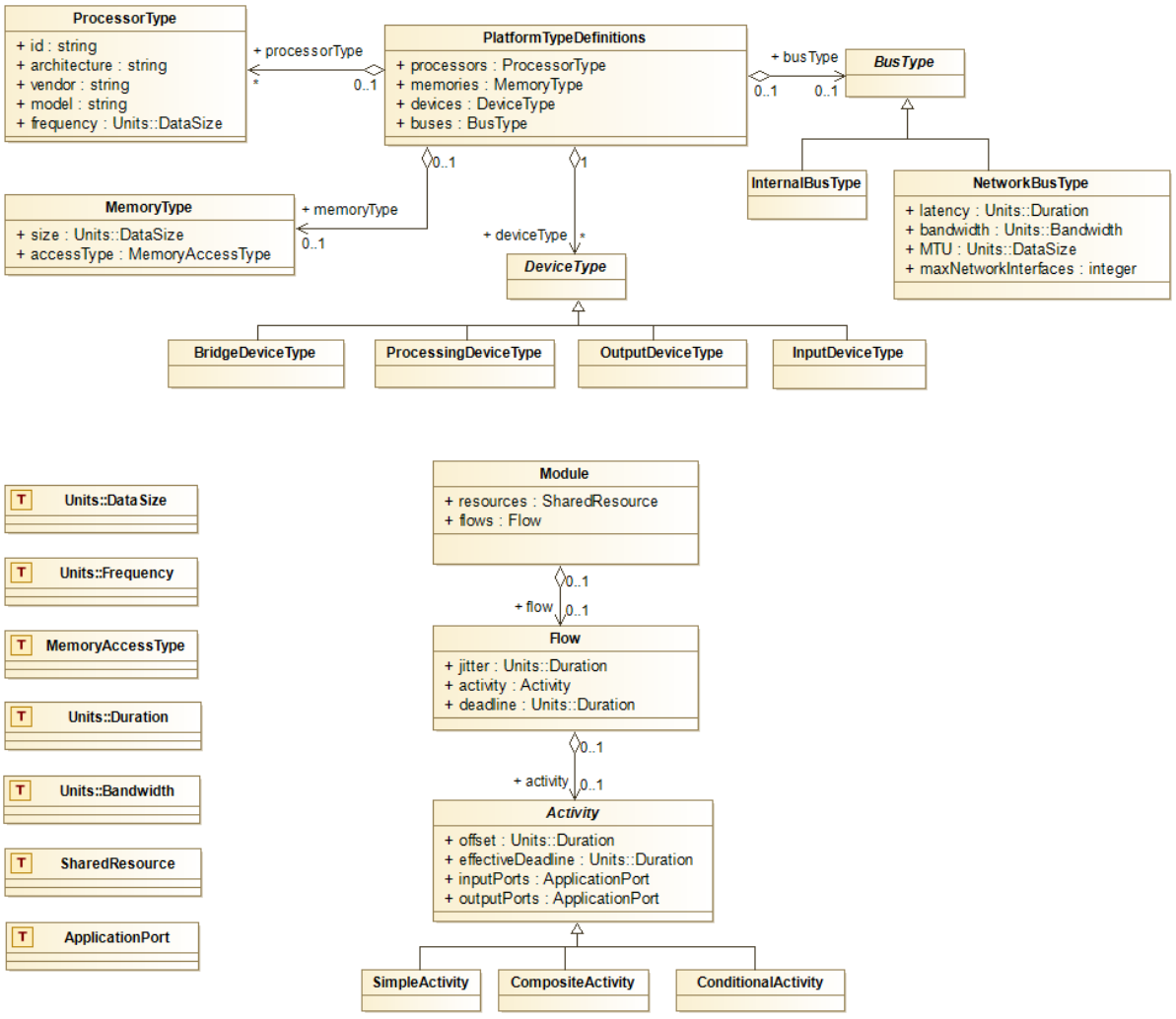

Figure 31 Platform and Application Model

PlatformTypeDefinitions

This is a container class which represents all the hardware components in a cyber-physical system.

ProcessorType

ProcessorType describes a processing device, including details of the type of processor and its clock frequency.

MemoryType

MemoryType describes a memory component, including its size and type of access mode.

DeviceType

Device type is an abstract class for representing all types of devices in a cyber-physical system.

BridgeDeviceType

BridgeDeviceType is a communications bus which interconnects two or more different networks.

ProcessingDeviceType

ProcessingDeviceType represents a processor in a cyber-physical system.

OutputDeviceType

An output port, usually connected to a communications bus.

InputDeviceType

An input port, usually connected to a communications bus.

BusType

BusType is an abstract class which encapsulates the different types of communications buses in a cyber-physical system.

InternalBusType

InternalBusType is a communications bus that is internal to a processing device.

NetworkBusType

NetworkBusType represents a communications bus or network. This class contains fields such as latency, bandwidth, message transfer size (MTU) and the maximum number of interfaces that may be attached to the bus.

Module

A Module is a container for objects that describe the application software running on a processor in a cyber-physical system. It contains a list of the shared resources controlled by the processor as well as a list of all the software flows of the system.

Flow

A Flow is a software unit. This class contains timing details of the flow as well as its activation type (e.g.sporadic, periodic, etc.). A GFlow also contains a list of activities or tasks, and how these are interconnected with each other.

Activity

An Activity is a base class for the most basic type of execution activity in application software. Activities have timing constraints (deadlines) and may be connected to input and output ports.

SimpleActivity

A basic single activity.

CompositeActivity

A CompositeActivity is an Activity that is composed of other activities that may be interconnected.

ConditionalActivity

A special type of activity that is only executed when a certain condition is satisfied.

6.3.8 TEK

In this section, we present the data models of TEK.

6.3.8.1 Testing modelling elements

Testing data are used for the use case scenarios TEK_UCS_01 “Design choices verification”, which deals with functional verification of the models and with design space exploration, and for TEK_UCS_02 “Run-time verification”. Currently, the data (descriptions of case tests, test results, tracing data such as creation/modification and execution data, and author name) are in an Excel workbook together with the requirements. The structure is such that testing data can be exported in a standard format, e.g. XML, and from there transformed to other formats.

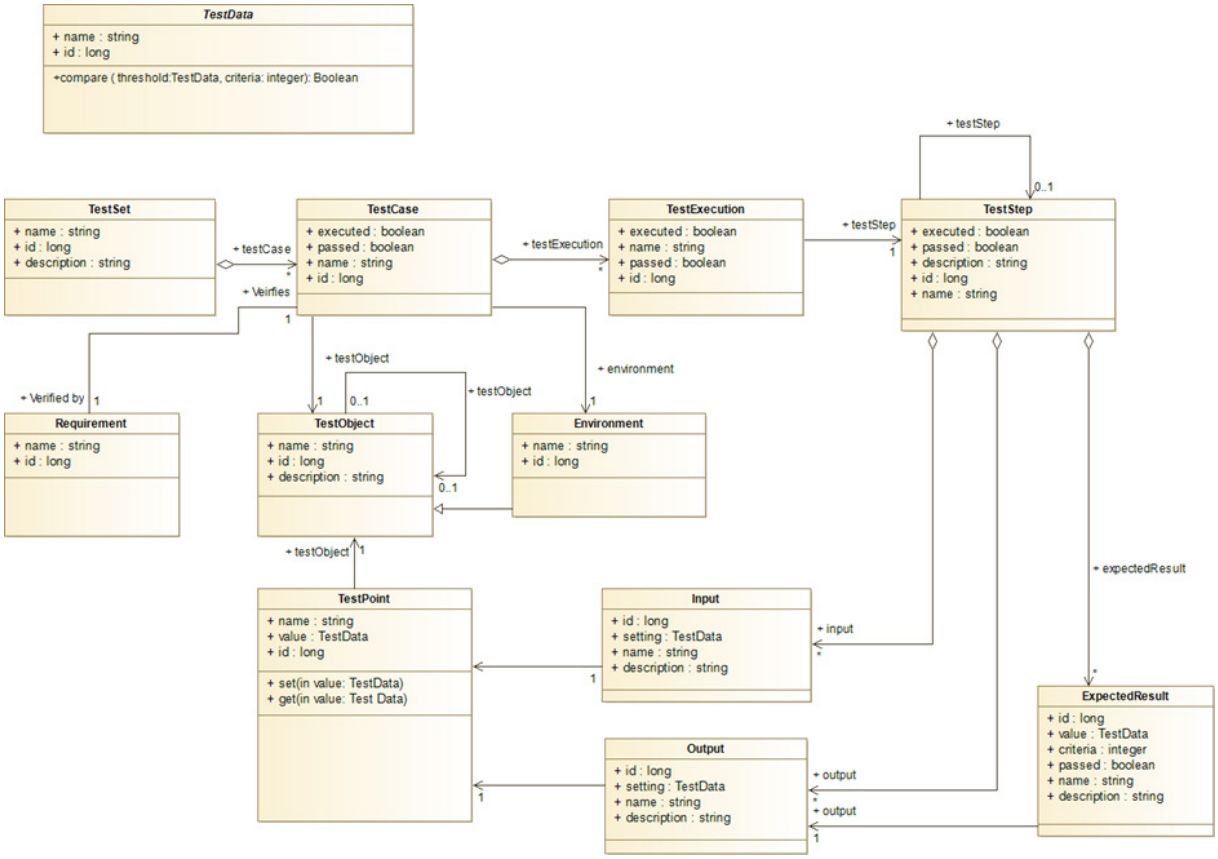

Figure 32 Testing modelling elements

TestData

TestData is the super-class from which all needed data types are derived.

TestSet

A TestSet object is a collection that can serve as a container for multiple TestCase objects that are related to each other, for example, for the level of requirements that they verify.

TestCase

A TestCase object is a collection of TestExecution objects. When the test case measures and compares against a threshold a quantity that depends on a parameter (for example, "power greater than 1 Watt for the temperature that varies from -40 to + 85 °C"), multiple test executions are needed to sample the parameter range.

TestExecution

A TestExecution object is a sequence of TestStep objects.

TestStep

A TestStep object has a collection, which can be empty, of Input objects and a collection, which can be empty, of Output objects.

Finally, it has a collection, which can be empty, of ExpectedResult objects.

Requirement

The requirement is satisfied when the status of all test steps of all test executions is “passed”. There is one TestCase object per Requirement object, though a TestCase object may encompass an indeterminate number of TestExecution objects.

TestObject

The uppermost object of the test, on which the test case acts, is a TestObject object that can be iteratively composed by other TestObject objects and that is associated with the TestCase object.

Environment

The environment, in which the objects of the test stay, is of the kind Environment that is derived by TestObject, consequently all pre/post-conditions can be established/checked by the first/last test step.

TestPoint

A TestPoint object is one directed test point of an object of the test (instrumentation can be required): it can be an output where the measures are taken, or alternatively an input.

Input

Input objects: the members setting specifies the value that is set on the associated input test point at the beginning of the test step.

Output

Output objects specify the associated output test points where the measures are taken successively.

ExpectedResult

ExpectedResult objects: at the end of the test step, according to the member criteria (at most, equal, at least), the member value is compared against the member sample of the associated Output objects. When one of the expected results isn’t verified, the test step fails, the test execution fails and halts, the test case fails, and the requirement isn’t verified.

6.3.8.2 DataSet modelling elements

Monitoring data are analysed for detecting and classifying anomalies of the software system or of the physical systems the latter acts on. Classification and detection are enabled by AI-based components. The equipment currently considered is an automotive inverter. The measured quantities are time-stamped voltages, currents, and temperatures; some of them require an interface circuit to adapt their ranges to the sensors. The measured data are filtered by a front end. In the end, the on-board computing platform obtains the monitoring data for local or remote usage. The logical model of the data (see Figure 34) is used in the use case scenarios TEK_UCS_03 “Operating life monitoring”. The model is suitable for the training phase too, and for this, it takes into account that the monitoring data can be generated through simulation. The final system can use a reduced model.

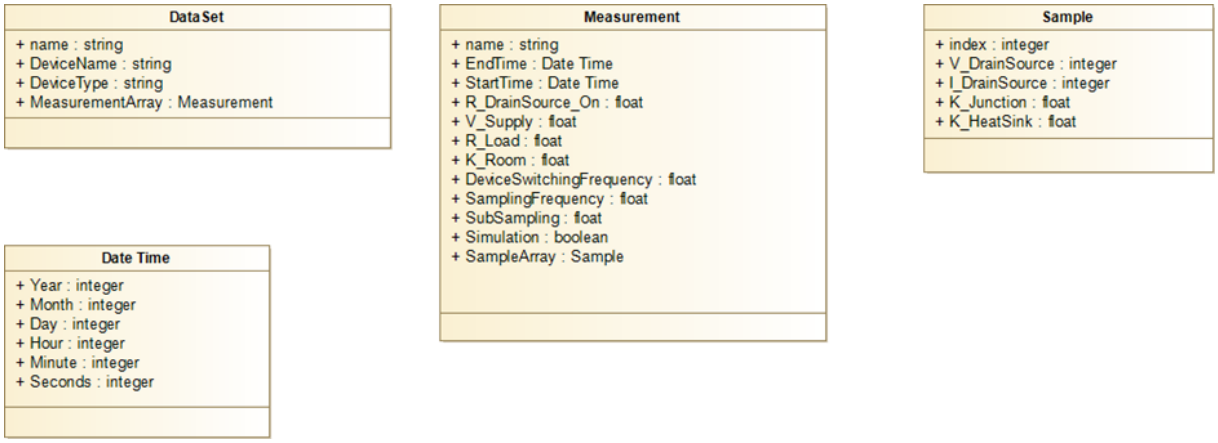

Figure 33 DataSet modelling elements

DataSet

Data are stored in a database that is a collection of DataSet objects (the database physical design is implementation dependent). A DataSet object is the collection of all measurements that are taken on a given device. The measurements are recorded in the Measurement array that is a member of DataSet.

Measurement

The Measurement object is the collection of the measures taken during a measurement session. The measures are recorded in the Sample array that is a member of Measurement. The measurement session is identified by the member “name” and spans the time interval from StartTime to EndTime. The measures are taken periodically as specified by the members SamplingFrequency and SubSampling. The measures can be generated by simulation, as specified by the member Simulation. The other members of Measurement specify the conditions under which the measurement session is executed.

Date Time

The very simple DateTime class was added because not all modelling tools have “date” and “time” among primitive types.

Sample

The members of a Sample object are the values that result from a multidimensional measure (note: the member Index is a progressive number incremented for every measure of the sequence).

6.3.9 VCE

In this section, we present the data model of VCE.

This diagram shows the concepts involved in the Volvo Use Case. In particular, it defines the data exchanges across the partner's solutions integrated in a solution architecture for the Volvo Use Case on Modeling and Simulation of a Dumper vehicle, its variants, and its subcomponents.

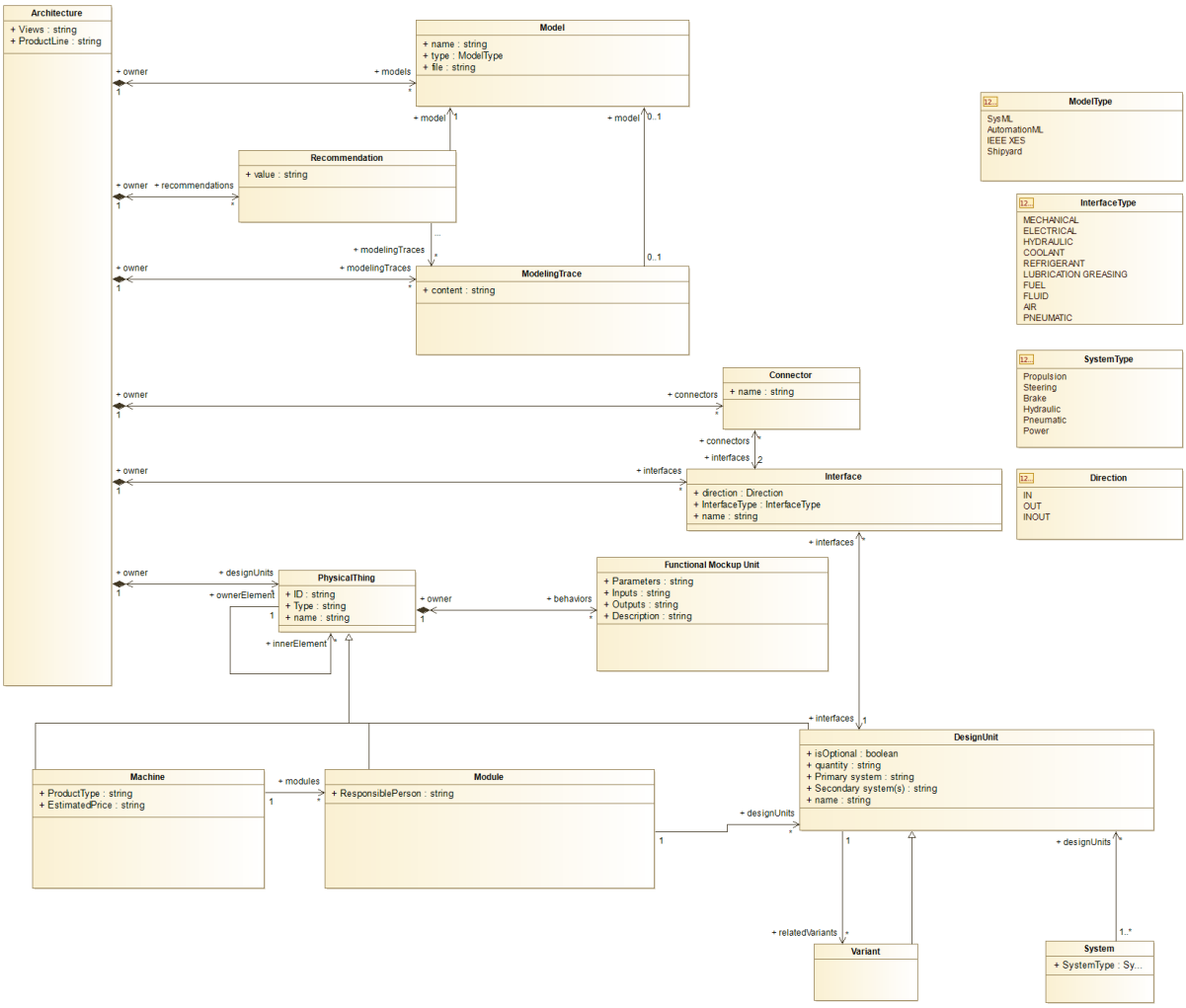

Figure 34 VCE UC Overall Data Model

Architecture

The architecture is made up of various connectors, interfaces, and design units. The architecture contains different views, two were inputs for the AIDOaRt project, while the project develops other types of Views through the solution architecture developed by the collaborative partners. The views that were used as inputs were one in Excel and one in Visio. The Visio view was similar to a drawing of the considered system, while the Excel view was a table that contained something similar to a Bill of Materials. In the Bill of Materials, information regarding the variant configuration of the different product line options. Each Architecture regards a particular product line.

Connector

A connector connects a pair of interfaces defined by the PhysicalThings of the Architecture.

Interface

Each design unit contains at least one interface to other design units. These interfaces describe how the different design units are connected and have an appropriate type and direction.

Recommendation

Each of the recommendable elements can be given a recommendation regarding the next action for a user. This is given in terms of string values which detail the recommended next action. Usually, this is presented as a list of several potential actions.

Functional Mockup Unit

A Functional Mock-up Unit (FMU) is a software library that can be created using the Functional Mock-up Interface (FMI) standard [FMI]. The FMI is a free standard that defines a container and an interface to exchange dynamic simulation models using a combination of XML files, binaries and C code, distributed as a ZIP file[1]. It is supported by 170+ tools and maintained as a Modelica Association Project [Modelica]. FMUs simulate the behaviour of PhysicalThings in a given Architecture.

ModelingTrace

A modelling trace is an ordered set of elements that correspond to an action performed in the modelling editor by an end user.

DesignUnit

The design unit is the lowest level description of an element in an architecture description. It corresponds to the parts that make up a sub-system, and each design unit contains interfaces with connectors with appropriate types. Furthermore, each unit could be optional for a design, and as well could be a variable unit and, in such cases, would have two or more variants linked to it. Each Design unit also has a designated primary system it is attached to and could have a number of secondary systems it is related to as well.

Variant

A variant is a design unit which has at least 2 or more choices that can be picked when implementing the particular design unit.

System

The various design units are assigned to systems. Each system has a predefined type, such as electrical, power, hydraulic, etc. Furthermore, each of the design units might be connected to several systems, and a distinction is made between what is the primary system for each design unit, and any potential secondary systems. A secondary system could be another system that takes the input or output of a particular design unit.

PhysicalThing

Physical thing is the abstract class of Design unit, Module, and Machine. Each physical thing has an ID and a type of the aforementioned classes.

Machine

A machine is the combination of different modules that together make up the machine under observation. A machine has an identification for which product type it is. Furthermore, it has an

estimation of the total cost of all components in the system, which will vary depending on the variant selected and which of the optional design units are used in the design.

Module

A module is simply a set of design units that together create a sub-system, particular instances of this could be a brake system or energy system. A module is something that has clearly defined borders for the rest of the system.

Model

The model artefacts represent an abstraction of the system under study. The model is identified by a name and an enumerated type.

6.3.10 WETMO

In this section, we present the data model of WETMO.

UML class diagram of the Westermo Use Case. The diagram describes classes and their relations that are of relevance to nightly testing and DevOps of industrial communication equipment.

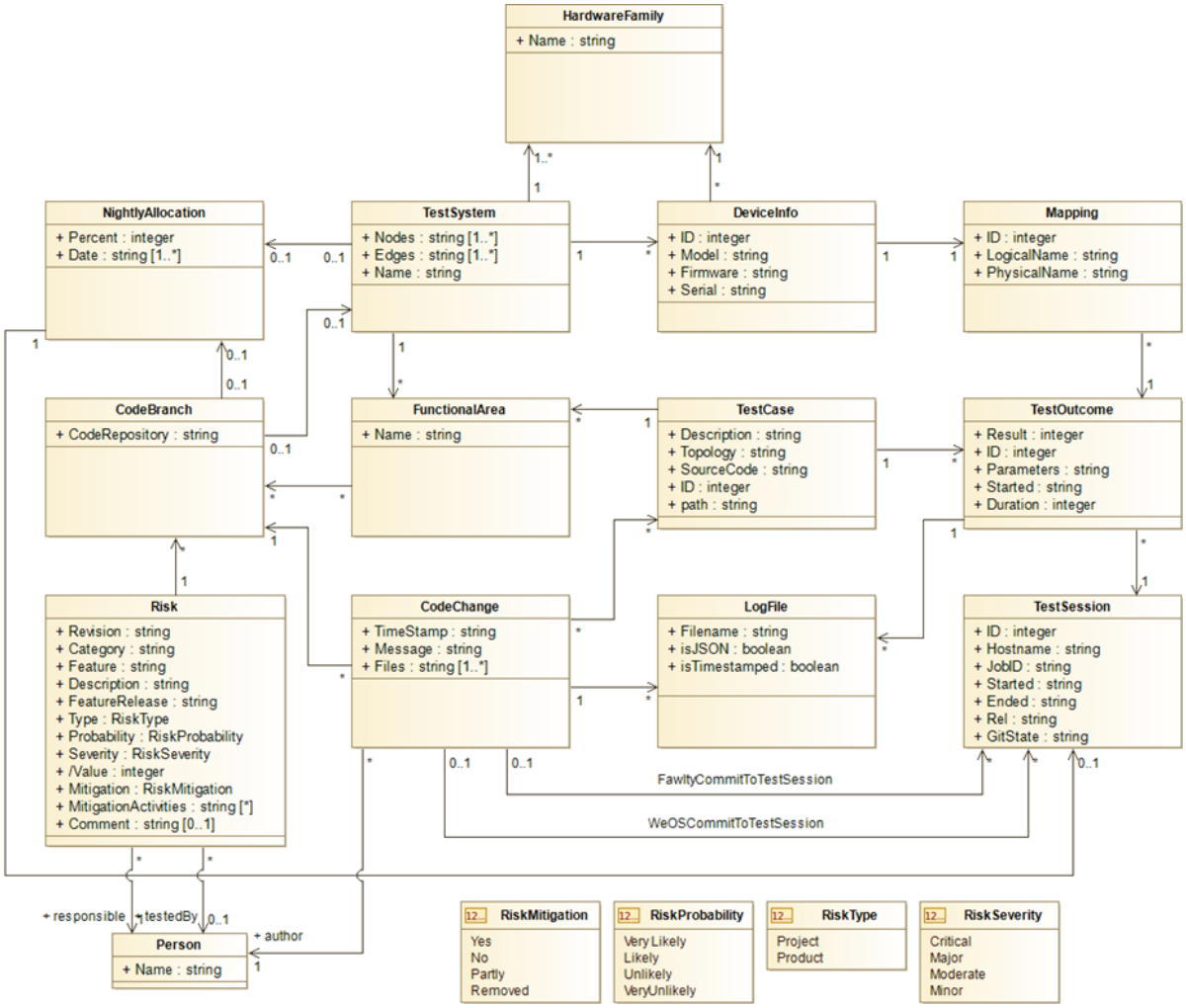

Figure 35 Westermo Class Diagram

Risk

Spreadsheet from risk workshops regarding releases of the software under test. In the sheet, for each release (code branch) of the software under test, risks are identified, tracked and mitigated.

LogFile

In the Westermo use case, several different types of log files are used, e.g. compilation logs, test execution logs, etc.

TestSystem

A test system topology contains information on the nodes in test systems (e.g. hardware version, serial number, etc.), as well as the links between nodes (copper ethernet, optical, serial, etc.).

TestSession

A test session can be thought of as a suite of test cases run on a certain test system, at a certain point in time, with a certain revision of the Fawlyt test framework, a certain revision of WeOS, the software under test. Only test sessions that are allocated can start.

TestOutcome

A test outcome represents when we run a certain test case on a certain test system at a certain point in time, as well as what the verdict was.

TestCase

A test case contains a description of what we want to test such that a human or the test framework could perform it. There is almost always source code for the automated test case, as well as a test case topology description on requirements from the test case on the test system (not all test cases can run on all test systems).

CodeChange

A code change describes the difference of one or more source code file(s), as well as information on who did the code change, when, and a plain text description of why.

DeviceInfo

DeviceInfos describe hardware used for testing. General information such as model is stored, as well as information on individual hardware devices (serial number).

Mapping

Mappings describe how devices were used in test cases.

FunctionalArea

A functional area represents a type of feature or communication protocol in WeOS, e.g. firewall, DHCP, etc. Test cases typically belong to one or more functional areas, but may not belong to any one in special cases. Not all test systems can run test cases for all functional areas (e.g. tests for serial protocols may require hardware with a serial port).

HardwareFamily

A hardware family represents a group of hardware products that share most components and that are only different in minor ways.

CodeBranch

A code branch is both a sub-project for a release and a way to organise code. Often, many code changes belong to each code branch. Sometimes new code branches introduce new functional areas, meaning that there is some mapping between code branches and functional areas -- not all areas can be tested with all branches. Also, not all hardware products are supported for all code branches.

NightlyAllocation

The NightlyAllocation is a representation of what code branches we run on what test systems, how much, and at what date intervals, E.g. one week, we may run branch X on test system A 50% of the night, and on test system B 30% of the night. Branch Y has another allocation, and branch Z is not allocated for testing since it has been completed and merged into the main branch. Each night, the NightlyAllocations decide what test sessions we start. If allocations are changed many times in one day, only the latest settings will trigger test sessions, so some allocations will never trigger test sessions.

Person

A person in this context is a member of the team. She could be the author of a code change that is to be tested, and/or responsible for mitigating a risk that has been identified for a code branch. If reported, the person who performed the testing for a risk usually is the same person responsible for the risk mitigation.

6.4 AIDOaRt Generic Data Model

In this section, we present the data models of the AIDOaRt Generic Data Model.

6.4.1 Requirements Engineering

The Requirements Engineering data model contains generic elements to represent the functional and non-functional requirements of a system, derived from an analysis of the stakeholders and context needs. A requirement could be a refinement of customer requirements, which are directly stated by a customer. A requirement is expressed from the point of view of the system.

Data models from Use Case providers ABI and TEK contain definitions related to the Requirements Engineering phase. Their elements also enrich the AIDOaRt generic entities.

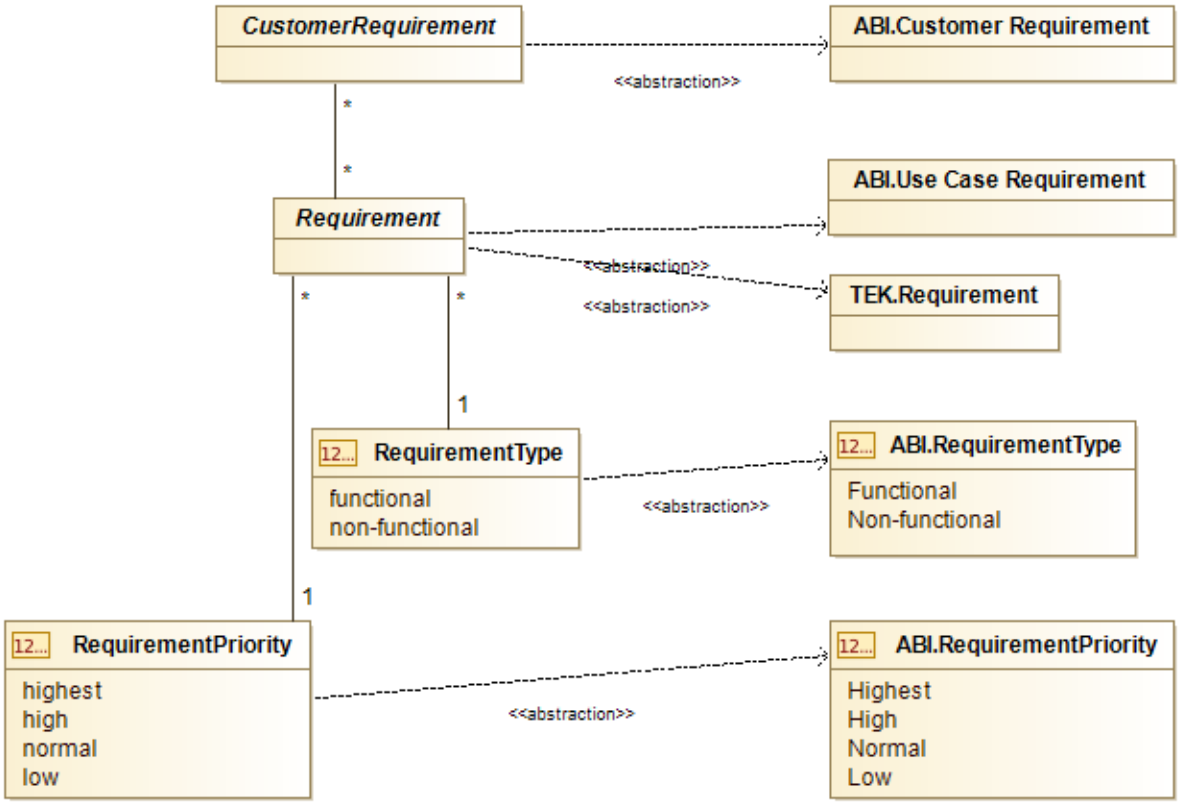

Figure 36 AIDOaRt Generic Data Model - Requirements Engineering Class Diagram

CustomerRequirement

A customer requirement represents the high-level specification of a need/constraint expressed by a customer. These requirements motivate customers to buy a product or service.

Requirement

A requirement is the specification, in terms of features or other software-/hardware-related needs that satisfy customer requirements.

A requirement is of one type: functional (it should be implemented as a functionality of the product) or non-functional (it should be taken into account when designing the final system so that it fulfils these quality constraints).

A requirement has a priority, which will facilitate the process of selecting and implementing requirements according to business and customer preferences.

6.4.2 Modelling

Modelling is an essential activity in an MBRE approach, in which an abstract representation of a system, an element of a system, or its behaviour, is depicted and implemented, and becomes its central asset.

Note the difference between "modelling a system" and "modelling its behaviour". In a few words, modelling a system is creating a visual representation of the system and its elements to, for instance, support making architectural design decisions or automating code generation. Modelling a system behaviour, on the other hand, may imply the implementation, training and evaluation of an AI/ML algorithm to recognise patterns and make predictions or decisions.

AVL, BT and VCE use models of different nature, and in different development phases, in their use cases. Their models are specific to their respective application domains.

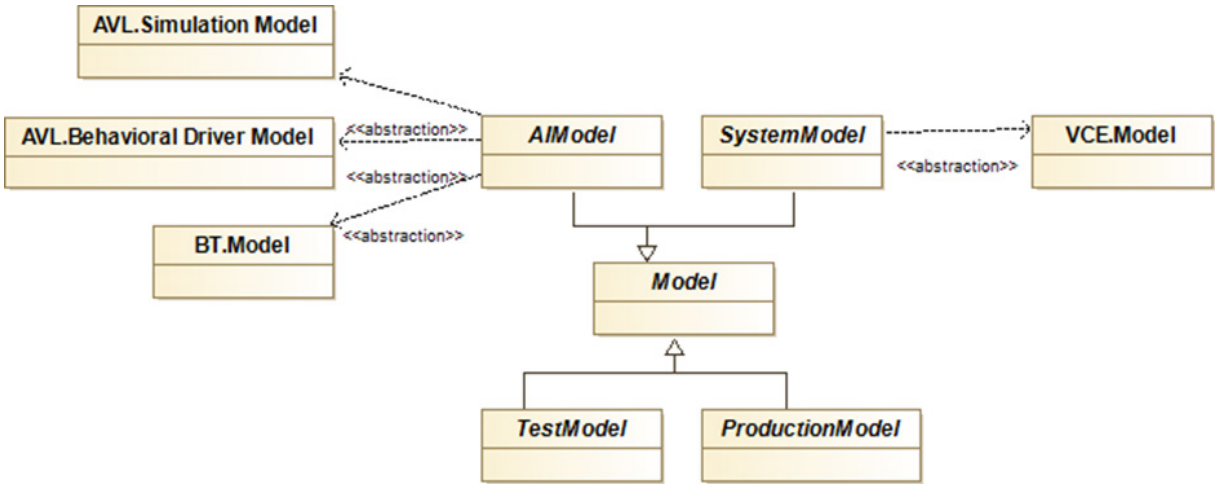

Figure 37 AIDOaRt Generic Data Model - Modelling Class Diagram

Model

A model is a simplified description or a mathematical/abstract representation of a system, an element of a system, or a process.

TestModel

A model is used in a testing phase to help simulate the actual conditions and behaviour of the system, in a controlled, simplified environment. Fine-tuning a model in the test phase is essential to get to the best model regarding its performance and accuracy.

ProductionModel

A model in production is used to perform real-time calculations and make predictions based on both observed and historical data. Such a model is useful to automate actions or make recommendations to the user of the system.

SystemModel

A system model is mainly used for driving the design and construction of a complex system, thanks to the use of modelling tools, in an MDE approach.

AIModel

An AI/ML model recognises patterns and assists in making simulations, predictions, recommendations and decisions.

6.4.3 Testing

Testing software and AI models is an essential step in the development lifecycle so that the team is able to detect bugs early and resolve them prior to a production release, or identify potential enhancements in the parameterization and configuration of AI models and systems.

There are several tools for automating testing, which eliminate the human risk of skipping errors, and require the definition and management of test cases/scenarios. The Testing data model contains elements for representing the definition and automation of tests and the adapting nature of the AI models (and other physical or logical elements) being tested.

AVL, CAMEA, TEK and WMO have their concrete test data models, specific to their particular domains.

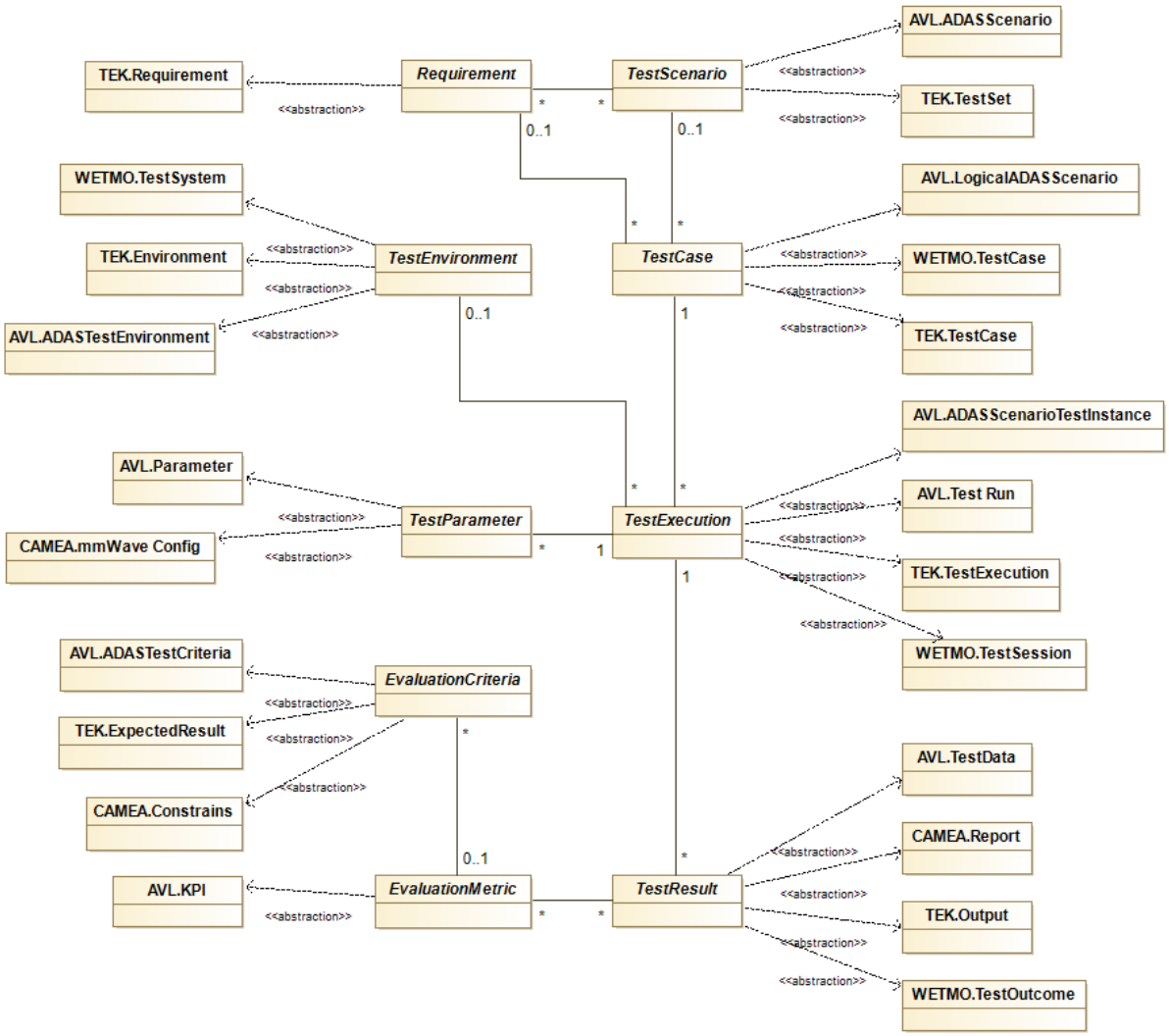

Figure 38 AIDOaRt Generic Data Model - Testing Class Diagram

TestCase

A test case is a set of actions executed to verify a particular feature or functionality of the system. A test case usually consists of test steps and test input data, and are part of a test scenario.

TestEnvironment

The test environment represents the conditions in which test cases are executed. It usually emulates the actual context in which the system could be placed in production, under controlled variables.

TestExecution

A text execution is a specific instance of a test case which ran in a testing environment, given input testing data and other testing configuration parameters. As a result, a set of data is generated.

TestResult

The execution of tests generates testing output data that is further analysed. This data is put in contrast with expectations and/or criteria that will discriminate if the test has been successful.

TestParameter

A series of test parameter values are tried in order to fine-tune successive tests, with the aim of achieving the best configuration of the system. They could be used in production as a baseline to monitor the variation of actual data.

EvaluationMetric

A set of metrics is defined to evaluate whether the output data from a test execution conforms to the expectations of the test case, and is aligned with the business goals.

An evaluation metric is not only used in test environments but also in production, to continuously monitor the performance of the system, based on actual collected data.

TestScenario

A test scenario provides a high-level point of view of the actions to perform a test. It corresponds to any functionality that can be tested and contains a set of test cases which helps the testing team to determine the status of the product.

Requirement

A requirement is the specification, in terms of features or other software- / hardware-related needs that satisfy customer requirements.

A requirement is of one type: functional (it should be implemented as a functionality of the product) or non-functional (it should be taken into account when designing the final system so that it fulfils these quality constraints).

A requirement has a priority, which will facilitate the process of selecting and implementing requirements according to business and customer preferences.

EvaluationCriteria

The evaluation criteria must be fulfilled by the test result or the data collected in production. In the scenario where a criterion is not met, it would be necessary to apply changes to the system or re-generate tests, to make sure that the system performance is still aligned to business needs.

6.4.4 Monitoring

In this section, we present the data models of Monitoring.

6.4.4.1 AIDOaRt Generic Data Model - Monitoring Class Diagram

Monitoring a system in production allows teams to respond to any degradation in the customer experience. The huge number of devices, sensors and actuators that interact with each other makes extremely important the need for continuous and (semi)automated monitoring.

In the Monitoring data model, the data that is collected from the sensors is measured against a set of evaluation metrics that allow, whenever it is possible, to automatically evaluate if an anomaly or a degradation of the system has occurred. Those metrics are set according to a series of KPIs that define the target performance of the system.

AVL, BT, PRO, TEK and VCE corresponding monitoring data models enrich the AIDOaRt generic elements with specific attributes and elements for their respective scenarios.

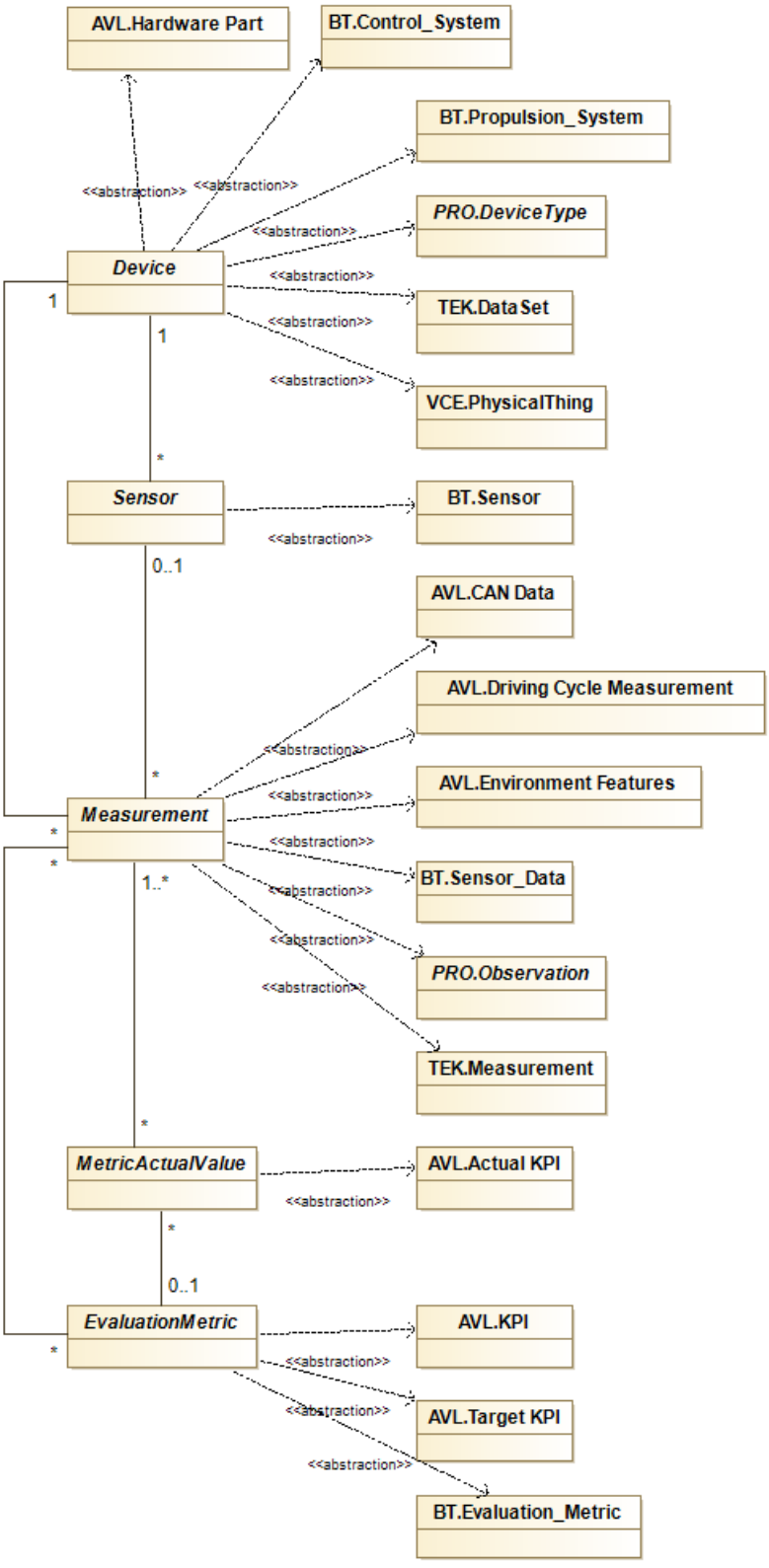

Figure 39 AIDOaRt Generic Data Model - Monitoring Class Diagram

Sensor

A sensor is a device that continuously collects data from the context and sends it to the system in order to be analysed. There is a large variety of sensors, specialised in quantitatively measuring different characteristics of the environment.

Measurement

A measurement is an instance of data that has been observed from the environment. Its structure and data, data types and units are specific to the domain.

Observation

This class is the central axis of the diagram because it is in charge of listing the different sensors that are deployed in the area of interest. Therefore, most of the schematic elements are interconnected with this class.

MetricActualValue

The actual value of a metric is calculated from the measurements made by the system, and is put in contrast to the corresponding defined evaluation metrics. When an actual measurement implies that a threshold set by an evaluation metric is surpassed or not achieved, it will lead to a further analysis of the system's performance and adequacy to business needs.

EvaluationMetric

A set of metrics is defined to evaluate whether the output data from a test execution conforms to the expectations of the test case, and is aligned with the business goals.

An evaluation metric is not only used in test environments but also in production, to continuously monitor the performance of the system, based on actual collected data.

Device

A device is a physical or logical element of the system. A device may contain sensors to collect data from the environment, and other devices or elements (e.g. actuators).

6.4.4.2 AIDOaRt Generic Data Model - Log Monitoring Class Diagram

Log monitoring is a supporting activity for system monitoring. Logs contain useful information that is collected throughout the system, throughout all its development phases. Monitoring is an activity that is not only performed in the production, with actual observed data but also in other development phases such as coding (for instance, to detect lines of code that introduce bugs) and testing (to support the assessment of the system being tested).

HIB and WMO manage and evaluate logs in their respective use cases.

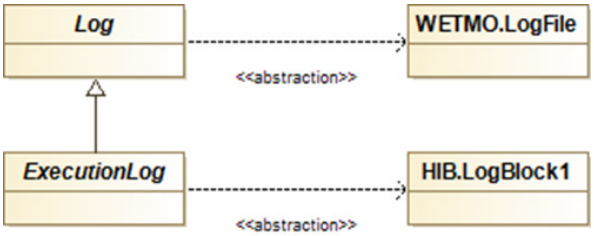

Figure 40 AIDOaRt Generic Data Model - Log Monitoring Class Diagram

Log

Logs are information that is collected in several phases of development (e.g. coding, testing, monitoring). Logs could be used to trace errors or to support an analysis of the system.

ExecutionLog

Execution logs are a concrete sub-type of logs. They are collected in production and provide additional information regarding the actual use of the system.

6.5 Conclusions and Next Steps

The mega-model is the result of an invaluable, combined effort between all partners of the consortium. UC providers have their area of expertise, and this is reflected in Figure 42 where we see the partners who contributed to each of the AIDOaRt generic layer sub-diagrams with their own data model elements.

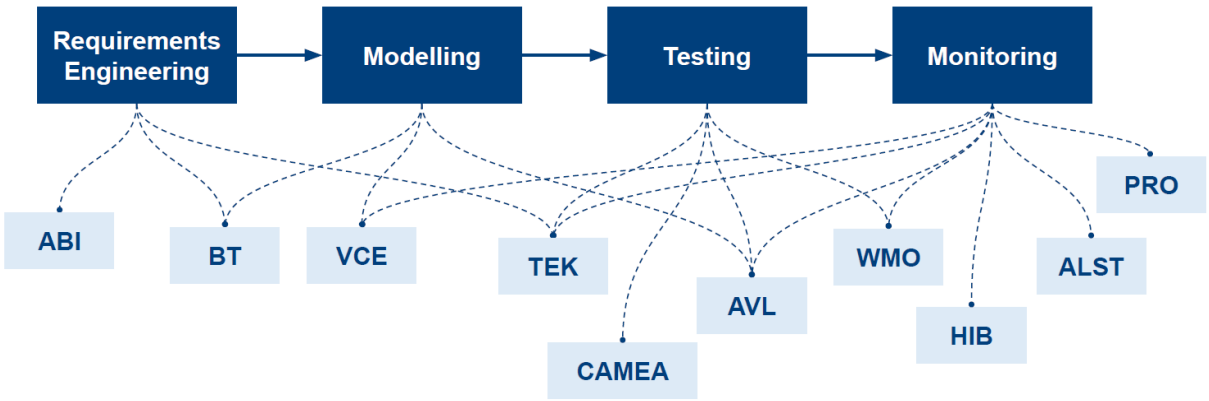

Figure 41 Mapping of UC partners who provided any concrete data element to an AIDOaRt phase

We have finally identified and specified 21 generic elements. It can be highlighted the popular interest in the Testing and Monitoring phases, which led to a more diverse definition of generic elements in these areas. Synergies and shared interests between partners are more probable to happen there. We have, on average:

- 2,5 concrete elements per generic one in the Requirements phase;
- 2,56 concrete elements per generic one in the Testing phase; and
- 3,4 concrete elements per generic one in the Monitoring phase (excluding log monitoring ones).

The average calculation is not straightforward in the Modelling phase since we did an additional abstraction exercise which resulted in more generic elements than concrete ones.

We have presented the first version of the mega-model and look forward to iterating on it as needed throughout the rest of the project. The first set of mega-modelling activities has been completed timely enough so that the project still can make use of the resulting output asset to boost its advances. As with any data model, the mega-model is a live asset that will continuously evolve in response to changes in the context of the scenarios it is representing, or adapt to new necessities from UC partners. In subsequent deliverables corresponding to other work packages, we expect to describe to what extent the partners have used and mainly benefited from the mega-model. We hereby encourage partners whose UC data models have entities mapped to the abstract elements of the mega-model to review commonalities with other partners, and evaluate a potential collaboration.

In this latter sense, a first potential collaboration has already been pointed out, which involves partners Westermo and HI Iberia, as they share an interest in regard to the activity of log monitoring. This has been spotted thanks to the connection established in the mega-model between Westermo's and HI Iberia's data models (see Figure 41). There it can be seen the generic entities Log and ExecutionLog connecting similar elements from WMO's and HIB's respective concrete data models.

We expect that partners will make the most of this partnership and the existing synergy between their concerns and also that the mega-model will be refined and further enriched.

7 Conclusion

This last deliverable of WP2 reports labour in the general context of data related processing in its three tasks, design time and runtime data collection, internal data representation, and data cleaning, analysis, and management. The status of the different date related solutions for the AIDOaRt Core Tool Set components and the way they can work to solve the challenges posted in the various use cases that need the Data Engineering Tool Set developed in AIDOaRt is here reported using the mappings to the components defined in the AIDOaRt conceptual Framework as a means to organise their presentation, something that we expect shall facilitate their reutilization to future readers.

As a relevant contribution, this final version of this Data Collection and Representation report proposes a mega-model as a comprehensive data modelling managing tool, a generic data representation model that is expected to be implemented and reused in a variety of stages of the DevOps process. This is constructed as an abstract summary of the data models used in the use cases with the aim of providing a common, agreed-upon, global data representation that can serve as a foundation for MDE-based activities throughout the various components and usages of the AIDOaRt framework. The modelling elements in the mega-model are organised according to aspects like requirements engineering, modelling, testing, monitoring, and the management of logs.

Further exploitation, usage, and evaluation of this mega-model will be done considering its ability to formalise and automate: the collection, storage, harmonisation, filtering, clustering and cleaning of data, the management of logs, weight the quality of data, model DevOps workflows, requirements, tests, or even general parameters of physical objects to support digital twins, for example. Relevant advances on these concerns may be reported in the remaining deliverables of other work packages, mainly in those of the integration work package (WP5).

8 Bibliography

[AIDOART-D1.4] AIDOaRt consortium: Architecture Specification Final Version, 2022. *(Access restricted to the AIDOaRt project's consortium)*

[AIDOART-D2.2] AIDOaRt consortium: Data collection and representation - Intermediate Version, 2022. *(Access restricted to the AIDOaRt project's consortium)*

[AIDOART-D5.3] AIDOaRt consortium: AIDOaRt Integrated Framework - Intermediate Version, 2023 *(Access restricted to the AIDOaRt project's consortium)*

[FMI] Functional Mock-up Interface. https://fmi-standard.org/

[ISO-11898-1] International Organization for Standardization, "Road vehicles – Controller area network (CAN) – Part 1: Data link layer and physical signalling," International Organization for Standardization, ISO Standard 11898-1, 2015.

[Modelica] Modelica Association. https://modelica.org/