OPEN PLATFORM FOR REALIZING ZERO DEFECTS IN CYBER PHYSICAL MANUFACTURING

D4.2 – Methodologies and first implementations

Version	1.0
WP	4
Delivery Date	6 Dec 2023
Dissemination level	PU
Deliverable lead	LMS
Authors	LMS, INTRA, AIMEN, IPB
Reviewers	INTRA, MSI
Abstract	Definition of the methodologies used in the developed software applications and in the openZDM platform. This document elaborates on the methodologies applied in the development process of the first version of the openZDM project's software components and its platform business layer. Furthermore, it defines the metadata model of the Asset Administration Shell.
Keywords	Software development methodologies, Asset Administration Shell metadata model, platform integration, software user experience
License	This work is licensed under a Creative Commons Attribution-No Derivatives 4.0 International License (CC BY-ND 4.0). See: https://creativecommons.org/licenses/by-nd/4.0/

Dissemination Level:	
PU	**Public, fully open**
SEN	Sensitive, limited under the conditions of the Grant Agreement
Classified R-UE/EU-R	EU RESTRICTED under the Commission Decision No2015/444
Classified C-UE/EU-C	EU CONFIDENTIAL under the Commission Decision No2015/444
Classified S-UE/EU-S	EU SECRET under the Commission Decision No2015/444
Type	
R	**Document, report (excluding the periodic and final reports)**
DEM	Demonstrator, pilot, prototype, plan designs
DEC	Websites, patents filing, press & media actions, videos, etc.
DATA	Data sets, microdata, etc.
DMP	Data management plan
ETHICS	Deliverables related to ethics issues.
SECURITY	Deliverables related to security issues
OTHER	Software, technical diagram, algorithms, models, etc.

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or EUROPEAN HEALTH AND DIGITAL EXECUTIVE AGENCY (HADEA). Neither the European Union nor the granting authority can be held responsible for them.

Version History

Version	Date	Owner	Author(s)	Changes to previous version
0.1	2023-07-31	LMS	LMS	Outline
0.5	2023-07-24	LMS	LMS, AIMEN, INTRA	Content provided
0.9	2023-09-24	LMS	LMS, INTRA	Revised version
1.0	2023-12-08	LMS	LMS	Final draft

Table of Contents

List of Abbreviations & Acronyms

AAS	:	Asset Administration Shell
AASX	:	Eclipse AASX Package Explorer
CIM	:	Common Information Model
CSS	:	Cascading Style Sheets
DIMOFAC	:	Digital Intelligent MOdular FACtories [1]
DST	:	Decision Support Tool
DT	:	Digital Twin
DTT	:	Digital Twin Toolset
HTTP	:	Hypertext Transfer Protocol
ICT	:	Information and Communication Technologies
IDS	:	International Data Spaces
IDTA	:	Industrial Digital Twin Association
KPI	:	Key Performance Indicator
LCA	:	Life Cycle Assessment
MIP	:	Mixed Integer Programming
ML	:	Machine Learning
MQTT	:	Message Queuing Telemetry Transport
NDI	:	Non-destructive instrument
RAMI4.0	:	Reference Architectural Model Industry 4.0
RFR	:	Random Forrest Regressor
SLSQP	:	Sequential Least Squares Programming
UI	:	User Interface
UML	:	Unified Modeling Language
ZDM	:	Zero-Defect Manufacturing

List of Figures

List of Tables

Executive Summary

The purpose of this document is to report the main methodologies followed during the development of the initial version of the openZDM applications and briefly present their first implementation. In line with the project scope, the openZDM applications developed, are focused on achieving defect identification, reduction, and utterly proactive quality control in manufacturing.

Furthermore, the openZDM applications are part of the business layer of the AAS-enabled openZDM platform. The main applications include a (i) Digital Twin Toolset (DTT), allowing the creation and execution of hybrid digital twins, (ii) several data-driven quality assessment modules, and (iii) a Decision Support Tool (DST), enabling recommendations for defects reduction, as well as the generation, simulation and evaluation of alternative what-if scenarios, resulting either to recommendations to the user or to direct control of relevant production assets, through the openZDM platform.

The methodologies used to create the main openZDM applications and the platform which exposes them are described in section 2, while section 3 presents their first implementations, as they stand by M18 of the project lifetime.

The document also discusses about the AAS data model and metamodel used in openZDM to model all necessary identified assets in its use cases. The methodology and implementation of both the data model and metamodel are also detailed in sections 2 and 3, respectively.

Lastly, Section 4 concludes with the identified risks and challenges related to the realization of the openZDM components and Section 5 outlines the lessons learned and recommendations for practitioners.

1. Introduction

The openZDM project aims at providing an innovative open platform that combines advanced ICT solutions and innovative non-destructive inspection systems, to enable inline quality assessment and proactive quality control towards zero-defect manufacturing.

A high-level representation of the overall openZDM reference architecture is provided in Figure 1, illustrating all expected functionalities of the overall openZDM system. In the context of each use case, the architecture will be instantiated according to the specific needs, constraints, and expectations of each industrial demonstrator, aiming to extensively test and validate the project outcomes and finally provide the final integrated openZDM system with its integrated set of applications for quality monitoring and proactive control.

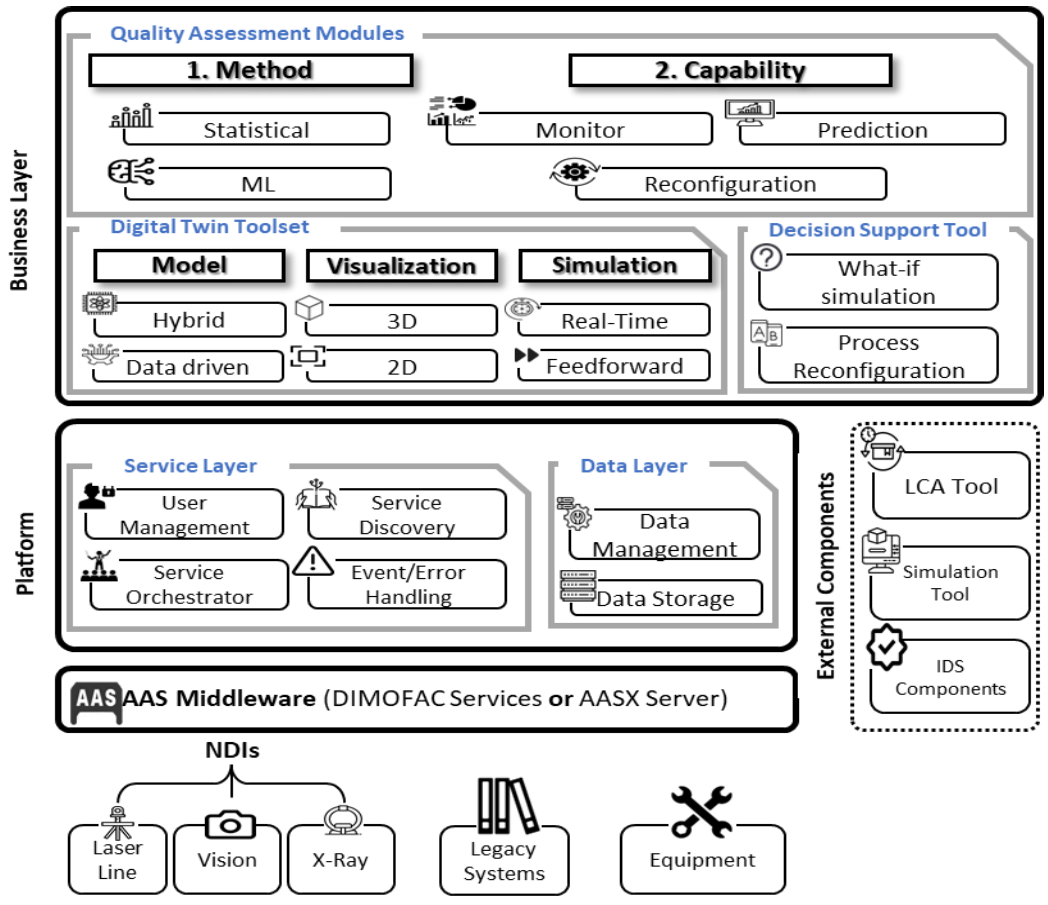


Figure 1: High-level representation of the reference openZDM system

In this context, a number of key software applications are developed in the project including the following:

1. An AAS-driven open platform, which integrates and exposes all other applications. Open APIs are included in the platform to support connection with third-party tools, some identified in the context of the project, such as LCA tools, as well as possible future ones, supporting the scalability of the solution. Also, the use of AASs, and in particular the envisioned AASs of type 3, are expected to provide extensive interoperability with industrial systems in the middle to long-term, as well as facilitate proactive quality control strategies being applied at the edge. The platform based on AASs, may either used as a middleware either a solution coming from the DIMOFAC project or the AASX server, putting the focus on the proper implementation of the AAS models and less on the middleware used.

Furthermore, the platform is coupled with support and data management services, as illustrated in Figure 1.

2. A set of tools to create and load digital twins, named DTT, coupled with the required services for simulation, feedforward execution and synthetic data generation. The tool supports the creation of data-driven digital twins, to physics-based as well as hybrid, whose high-level architecture is presented in section 2.3.2. Also, alternative visualisations are included, addressing versatile needs and purposes.

3. Several data-driven quality assessment modules, accessible through the openZDM platform, are in charge of data analysis and defects identification as well as prediction. The tools use data-driven methods, statistical and machine learning to allow for defect monitoring, analytics visualisation and facilitating defect prediction.
4. A decision support tool, built on top of the quality assessment modules and the digital twins, in charge of generating recommendations for reducing defects, and/or improving the sustainability of the production process, either as a whole or partially. Furthermore, this tool can create and evaluate alternative what-if scenarios executed on top of the digital twin(s) and with the support of the quality assessment modules. The DTS is also expected to allow for the collection of user feedback and accordingly constrain its future recommendations, as well as impact the performance of the digital twin and quality assessment modules.

In addition, the openZDM platform supports integration with NDIs deployed at the edge, legacy systems, equipment and external components. In addition, it supports secure data sharing through the integration of an IDS connector.

2. Methodologies

2.1 Data modelling

The AAS data model in the context of the openZDM project has followed the relevant standards and regulations as well as the experience of consortium partners from previous projects. There are 3 types of AAS: AAS Type 1 (Passive) is a serialized file (XML or JSON) which contains static asset information. This type of AAS is the basis for AAS type 2 and type 3. The AAS Type 2 (Active), is host-based and contains static and dynamic asset information. Finally, the AAS type 3 (reactive), can communicate with other AAS or software components through peer-to-peer communication protocols. The openZDM project's goal is to develop, deploy and use type 3 AASs. For its development, the following 3-step methodology has been used:

- **Step 1**: Define the assets involved in each use case and the information that each of them can provide (output data) and the information they require (input data) for the use case they belong to. This information is classified into Rest data and Motion Data. As a result, a document with the list of assets to be modelled by AAS is obtained.
- **Step 2**: Define the Asset Administration Shell structure for each asset identified for each of the use cases. In this step, the information related to the asset is separated into sub-models (according to the domain to which the information belongs). The Industrial Digital Twin Association (IDTA) has developed some submodel templates for AAS in general, such as submodels for TechnicalData, HierarchicalStructures, ContactInformation, and DigitalNameplate, among others. Of these submodels, their suitability for the different use cases developed in the openZDM project will be evaluated. If no submodels are found that fit the use cases presented in openZDM, these templates will have to be designed by the project. Once the structure of the submodels is defined, the AASs are created using the AASX Package Explorer tool [2]. As a result, an AASX file is obtained that can be used as AAS type 1.
- **Step 3**: Define the communication of AASX files with external elements. In this step, the modelled AASs are hosted on the platform developed in the project for future interaction with other AASs.

2.2 Metadata modelling

The main objective of Industry 4.0 is to facilitate the integration and cooperation of the assets involved in a manufacturing process. An asset is any object that has value for a company (machines, sensors, products, drawings, work cells, among others). For this integration and cooperation to be feasible, assets must be described and connected virtually. The Platform Industrie 4.0 [3] has developed the "Reference Architecture Model Industrie 4.0" RAMI 4.0, in order to be able to describe all the technical characteristics of an asset throughout its life cycle (Figure 2). RAMI 4.0 approaches the description of an asset by means of a three-dimensional model: Layers, in which the asset architecture is described in terms of properties, lifecycle & value stream, which is used to describe an asset at a given point in its lifecycle (from design to final disposition) based on the IEC62890 standard, and hierarchy levels, which allows functional models to be assigned to specific levels based on DIN EN 62264-1 and DIN EN 61512-1 standards [4].

One of the technologies used in Industrie 4.0 is the digital twin. A digital twin can be implemented using the Asset Administration Shell (AAS) [5]. The Asset Administration Shell is a central concept for representing the technical functionalities of an asset according to RAMI 4.0 and enabling interoperability.

The Platform Industrie 4.0 has proposed an AAS metamodel which defines mandatory and optional attributes for all AAS in UML language, as can be seen in Figure 3.

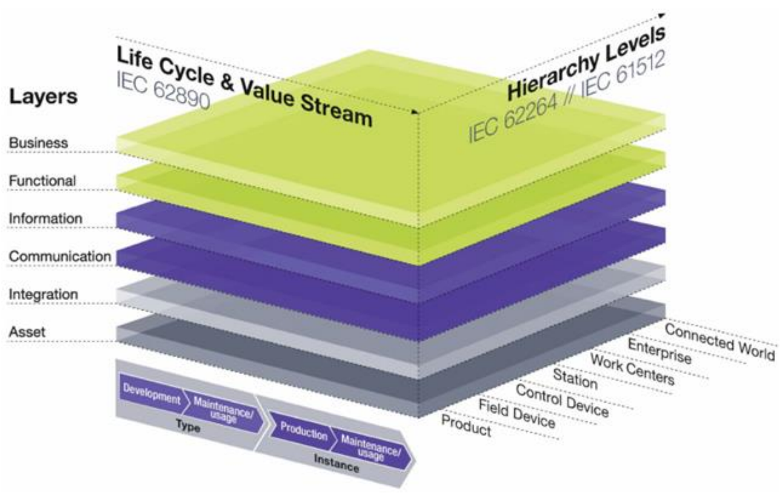


Figure 2: RAMI 4.0 Architecture

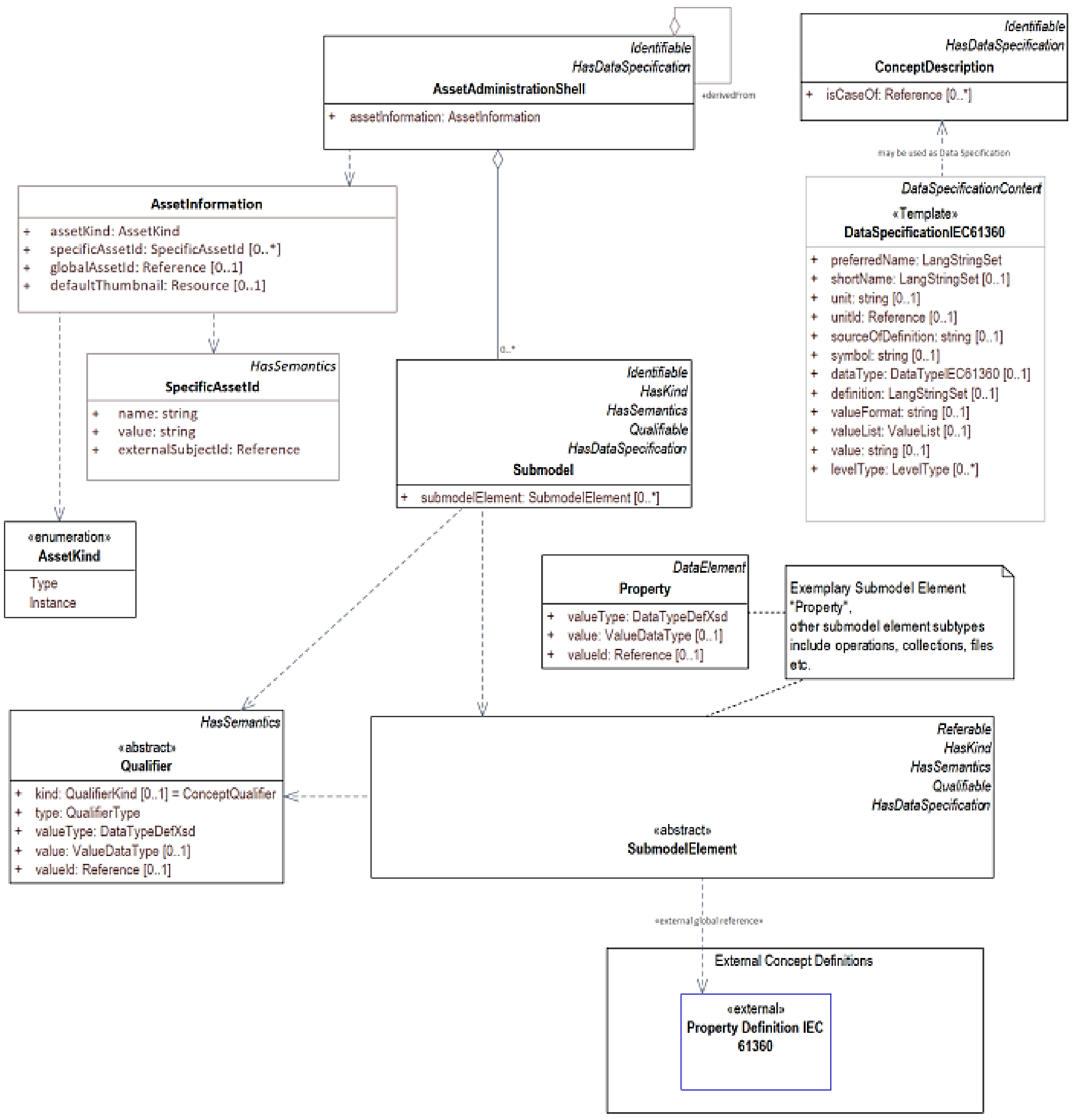

Identifiable
HasDataSpecification
AssetAdministrationShell
+ assetInformation: AssetInformation
+derivedFrom
Identifiable
HasDataSpecification
ConceptDescription
+ isCaseOf: Reference [0..*]
may be used as Data Specification
DataSpecificationContent
«Template»
DataSpecificationIEC61360
+ preferredName: LangStringSet
+ shortName: LangStringSet [0..1]
+ unit: string [0..1]
+ unitId: Reference [0..1]
+ sourceOfDefinition: string [0..1]
+ symbol: string [0..1]
+ dataType: DataTypeIEC61360 [0..1]
+ definition: LangStringSet [0..1]
+ valueFormat: string [0..1]
+ valueList: ValueList [0..1]
+ value: string [0..1]
+ levelType: LevelType [0..*]
AssetInformation
+ assetKind: AssetKind
+ specificAssetId: SpecificAssetId [0..*]
+ globalAssetId: Reference [0..1]
+ defaultThumbnail: Resource [0..1]
HasSemantics
SpecificAssetId
+ name: string
+ value: string
+ externalSubjectId: Reference
0..*
Identifiable
HasKind
HasSemantics
Qualifiable
HasDataSpecification
Submodel
+ submodelElement: SubmodelElement [0..*]
«enumeration»
AssetKind
Type
Instance
DataElement
Property
+ valueType: DataTypeDefXsd
+ value: ValueDataType [0..1]
+ valueId: Reference [0..1]
Exemplary Submodel Element "Property", other submodel element subtypes include operations, collections, files etc.
HasSemantics
«abstract»
Qualifier
+ kind: QualifierKind [0..1] = ConceptQualifier
+ type: QualifierType
+ valueType: DataTypeDefXsd
+ value: ValueDataType [0..1]
+ valueId: Reference [0..1]
Referable
HasKind
HasSemantics
Qualifiable
HasDataSpecification
«abstract»
SubmodelElement
«external global reference»
External Concept Definitions
«external»
Property Definition IEC 61360

Figure 3: Overview metamodel of the Asset Administration Shell

The lack of systematic means to integrate the different data lifecycles of different devices leads to so-called solution silos problems, where solutions for specific use cases lack interoperability [6]. To address this problem, it is necessary to develop a common information model that allows the representation of data throughout its life cycle in a language common to all use cases. In the openZDM Project, the AAS metamodel is used as the Common Information Model (CIM) for all use cases. The AAS Metamodel allows the representation of assets throughout their life cycle by means of sub-models. Each submodel allows the description of a specific domain of the asset by means of properties and relationships. The AAS metamodel provides a set of classes and relationships that make possible the creation of submodels for the description of the different assets in the different use cases in the openZDM project. The different domains of the asset can be described by means of properties, relationships, operations, events, collections or lists of these, among others. Existing standards or consortium specifications provide concepts that can be used to define properties within a submodel. Submodels based on standards can be considered such as templates which guarantee interoperability [5] and facilitate the modelling of assets.

The openZDM project develops a platform to implement quality management practices in 5 use cases of different types of production processes, with different types of data stored in different sources. In its current form, the AAS Metamodel version 2 was used [7].

2.3 Digital Twin Toolset

The DTT application is capable of both creating and loading previously created DTs. A created DT should follow ISO 23247 [8], according to which the fundamental part of a DT is its model. A model in openZDM can be either data-driven, physics-based or hybrid. In the context of data-driven models, high-quality and quantity historical data are used to create and train ML models through which the behaviour of the asset can be replicated in the virtual space. In openZDM to describe an asset's behaviour through a data-driven model, two ML techniques have been used:

1. The first is the symbolic regression ML algorithm. Through this algorithm, an equation that links the input and target variables is generated that is based on the provided to it historical data.
2. The second algorithm used is the logistic regression. The logistic regression algorithm was used to describe input parameters which are of continuous type with categorical target values. Logistic regression was used to describe production parameters and the probability of the presence or not of a defect on a product.

In the context of physics-based models an asset behaviour is given through, usually, complex mathematical equations that describe the physical phenomena and provide insight on the behaviour of the asset. Physics-based models have the ability of explainability however they can be less accurate than a data-driven model. Thus, in openZDM and in the DTT, a combination of the two has been developed, a Hybrid DT model, where the accuracy of the physics-based model is iteratively corrected through the data-driven model. A schema of the methodology followed during the construction of a hybrid model can be seen in Figure 4.

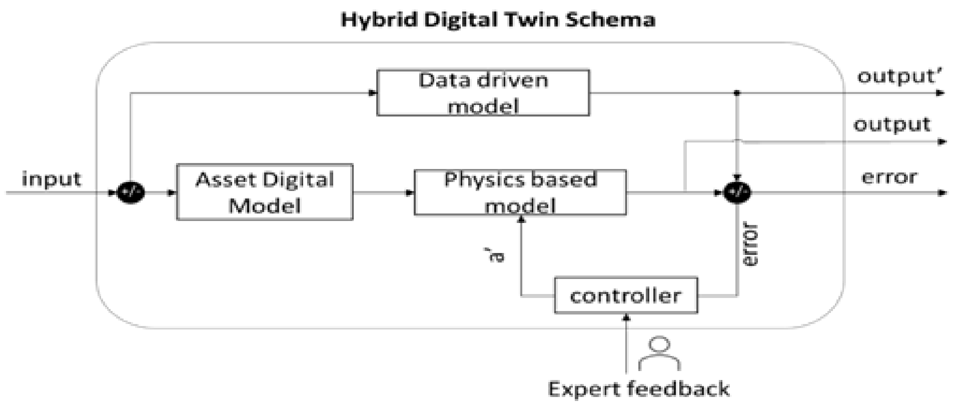


Figure 4: The hybrid digital twin model schema used in the DTT

A hybrid digital twin model is composed of a digital replica of an asset, a data-driven model replicating the under-consideration behaviour of the physical asset, and a physics-based model using differential equations to model its behaviour. The created data-driven model provides the data-driven behaviour using ML algorithms and the physics-based model describes the asset's behaviour using a set of mathematical equations. Each behavioural model has its output and using the deviation between them an error is calculated and controlled to Improve the accuracy and realism of the digital twin. The control approach aims to receive feedback from the user and constrain the behavioural models accordingly. Thus, through multiple iterations, the accuracy of the DT is expected to have significantly improved and in particular the physics-based model which may facilitate explainable recommendations to be generated.

According to ISO 23247, a functional DT requires the synchronization of the status of the digital twin with the status of the corresponding physical asset. In the DTT this is handled through the utilization of industry-standard communication protocols, like MQTT, in order to ensure fast and reliable communication between the physical asset and the asset in cyber space.

Furthermore, an integral part of the DTT is the visualization part and its reporting component. In the DTT two methodologies are followed which have been adapted to suit both the visualization and reporting needs. Visualization is handled both in a dashboard environment through graphs and charts (for descriptive and diagnostic analytics visualization) and in a three-dimensional environment using 3D models of the assets under study. Additionally, the production results and simulation results are reported through the visualisation component.

2.4 Data-driven Quality Assessment Modules

The data-driven quality assessment modules can use statistical tests, machine learning and data preprocessing methodologies to provide predictive, monitoring, diagnostic and descriptive analytics for defect identification, prediction, and quality assessment. The different modules developed in openZDM to achieve this purpose can be grouped based on their respective capabilities to a) monitoring, b) predictive and c) process reconfiguration. Figure 5 depicts a classification of the analytical tools based on their objectives and with the respective algorithms that are employed (note that Figure 5 is not complete but illustrative, particularly with respect to the algorithms column).

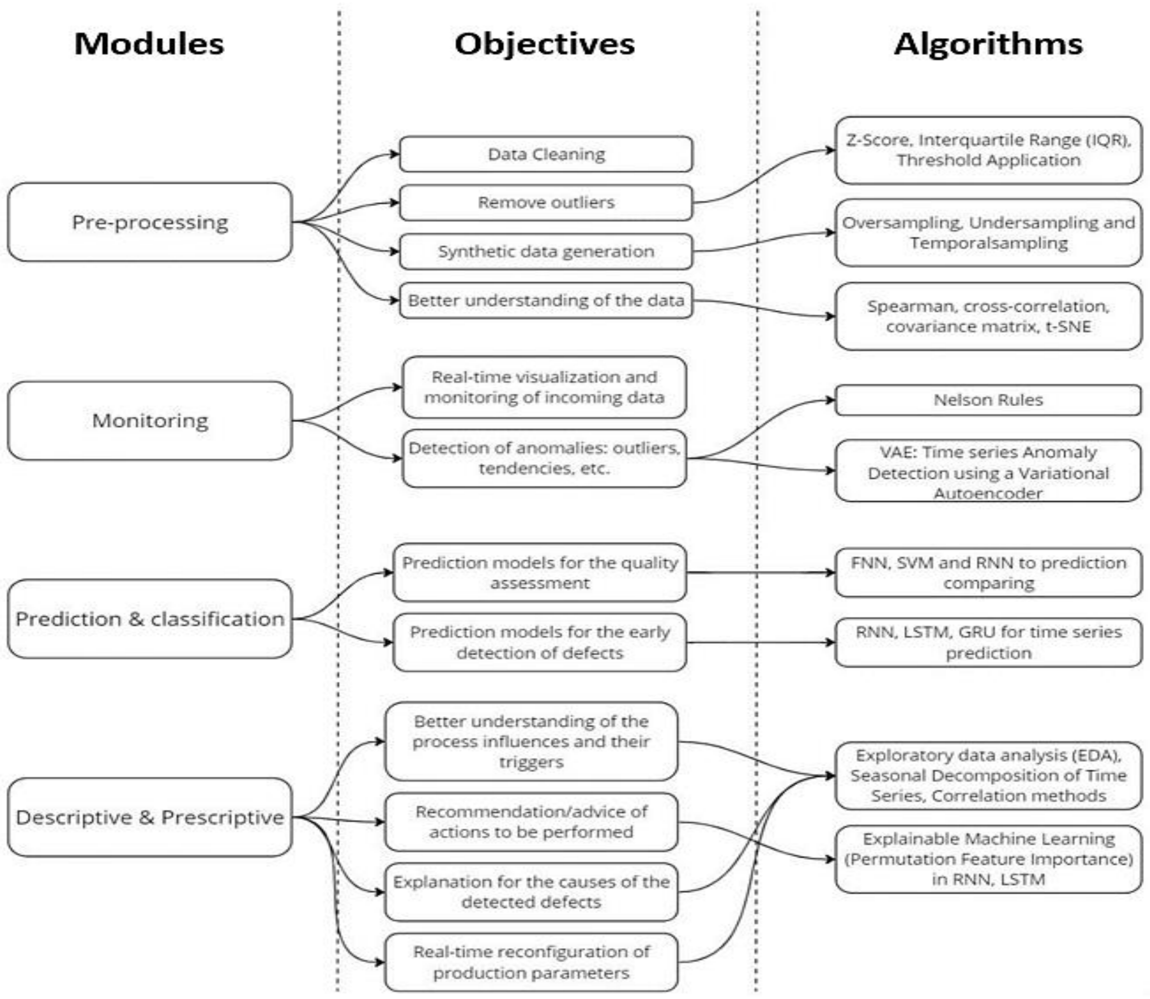

Modules
Objectives
Algorithms
Pre-processing
Data Cleaning
Remove outliers
Synthetic data generation
Better understanding of the data
Z-Score, Interquartile Range (IQR), Threshold Application
Oversampling, Undersampling and Temporalsampling
Spearman, cross-correlation, covariance matrix, t-SNE
Monitoring
Real-time visualization and monitoring of incoming data
Detection of anomalies: outliers, tendencies, etc.
Nelson Rules
VAE: Time series Anomaly Detection using a Variational Autoencoder
Prediction & classification
Prediction models for the quality assessment
Prediction models for the early detection of defects
FNN, SVM and RNN to prediction comparing
RNN, LSTM, GRU for time series prediction
Descriptive & Prescriptive
Better understanding of the process influences and their triggers
Recommendation/advice of actions to be performed
Explanation for the causes of the detected defects
Real-time reconfiguration of production parameters
Exploratory data analysis (EDA), Seasonal Decomposition of Time Series, Correlation methods
Explainable Machine Learning (Permutation Feature Importance) in RNN, LSTM

Figure 5: Data-driven quality assessment modules classification based on main objective/functionality.

To begin with, several data preprocessing modules are considered in the context of the openZDM project towards providing quality insights. The data analysis process requires however data of certain quantify but also quality. For example, an ML model cannot predict a defect that is never encountered in its training dataset. As such, the methods incorporated include but are not limited to the following:

- imputation techniques for the removal of missing values;
- data transformation techniques for variable creation;
- visualization techniques for exploratory data analysis;
- the Nelson rules technique for detection of outliers in manufacturing process data.

Additionally, statistical test modules have been developed which are capable of better understanding the association between variables and gaining inference (drawing meaningful statements and information on the underlying structure of the data). In further detail, the techniques used in the developed statistical test modules include:

- Pearson's, Spearman's and point biserial correlation techniques for the identification of correlation between variables;
- Chi-squared test for association identification between categorical variables;
- one way ANOVA for association identification between continuous and categorical variables.

Lastly, different predictive modules have been developed. These modules utilize the pre-processed data coming from the data preprocessing modules and the modules capable of conducting statistical tests. The predictive modules are focused on providing proactive quality control through early prediction of defects in a production line. In more detail, the models incorporated in the quality assessment modules include:

- MultiTarget Support Vector Regressor, Random Forest Regressor for capable of making predictions of multi-target variables;
- XGBoost, Decision Tree Classifier and kNN Classifiers capable of predicting defects.

2.5 Decision support tool for alternative process configuration

The DST for alternative process configuration in openZDM is responsible for identifying the optimal set of parameters to be configured to achieve a user input improvement in use case-specific KPIs. The DST follows a generic approach that can be adapted in every use case that is connected. The DST is connected to the DTT in order to acquire the input parameters and KPIs that should be improved. Additionally, the DST gets from the DTT the equation that links the input parameters and the output parameters that should be improved, as well as the relevant KPIs.

The DST is responsible for conducting feedforward simulation. In order to conduct the simulation, the DST takes as input the user-defined range of values for the input parameters, the value at which the parameter is incrementally increased, and the frequency at which a parameter can change, thus simulating a real-world production where changes to process parameters are not instant and a time horizon which indicates the future window until which the simulation will be conducted. Lastly, the DST is provided by the user with the targeted improvement value for the cost/benefit indicators.

The core of the DST uses an optimization algorithm. Multiple optimization algorithms are being taken into consideration in order to identify the most optimal algorithm for a specific application. Currently, a methodology using the SLSQP algorithm has been implemented into the DST. Through this algorithm, the DST can use the equations provided by the DTT and optimize its parameters to generate an optimal set of features that result in the user-defined KPI improvement. The optimisation algorithm is a gradient-based optimization method. This methodology was selected due to its capability of optimizing both non-linear and linear mathematical models that use continuous variables. Through this generic methodology, the DST is capable of optimising a methodology while being totally decoupled by a specific use case. Additionally, further methodologies are being investigated to be implemented into the DST, which include the MIP algorithm, the Deep Counterfactual Regret Minimization and the Sequential Game Models methodologies. Lastly, the what-if scenario results are presented to the user in the UI of the DST. A visual representation of the followed methodology in the DST can be found in Figure 6.

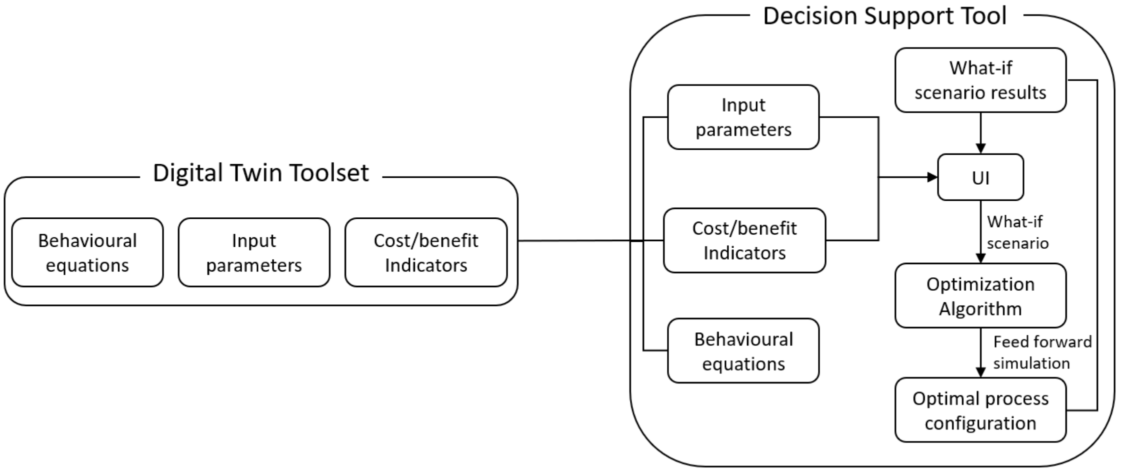


Figure 6: Methodology followed in the DST

2.6 Platform

The openZDM project follows a waterfall development methodology which starts from requirements elicitation, where all stakeholders identify the functionality to be fulfilled by the solution and its specification is defined. Next, the design phase describes the components of the solution and their behaviour, technologies to be adopted, and interfaces between the various components. After the design phase is completed the implementation of the solution begins, here all components are developed and integrated to realize the desired behaviour. After the implementation, the verification phase begins where the solution is tested against the specification to identify deviations and potential issues. The last phase is maintenance of the solution where updates are issued to enhance the solution or fix issues uncovered during its operation. Figure 7 illustrates the approach.

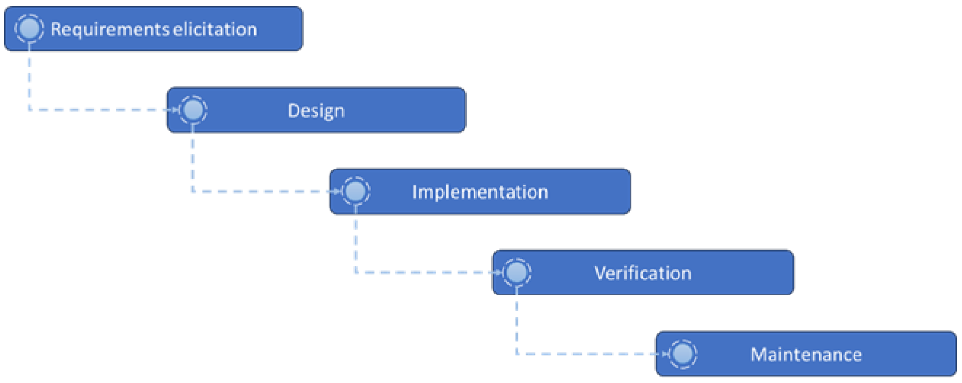


Figure 7: Waterfall development methodology of the openZDM platform

The implementation phase in addition adopts the prototype methodology, where developers work on a prototype version of the final product, gather end-user feedback evaluate it, and gradually add functionalities according to the design. More specifically, during the first development phase first versions of the data model, digital twin toolset, data-driven quality assessment modules, decision support tool for alternative process configuration, and platform services were developed and deployed together with existing components such as Asset Administration Shell components, the identity management service, and the message broker. As the openZDM platform relies on data coming from the shop floor AAS were used to represent digitally all different types of assets on the shop floor, such as products, machines, and non-destructive inspection systems. This on one hand requires modelling the assets as AAS (see chapters 2.1, 2.1, 3.1.1, and 3.1.2 for more details) and on the other the configuration of appropriate AAS components that realize the standardized AAS API [9]. The application of AAS enables the platform to connect new assets in a flexible way using mainly configuration and modelling.

The openZDM platform has been implemented using a layered architecture approach supporting a modular configuration with high independence and capable of operating using a cloud deployment, full deployment at the edge or hybrid. The use of docker containers and their management as a cluster through technologies like Kubernetes or docker swarm ensures the scalability and resilience of the deployed functionalities. In addition, the modular approach followed allows for the deployment of redundant modules and along with the error handling mechanisms facilitates fault tolerance and increased robustness. In addition, robustness is further facilitated by the project's integration approach where all the platform modules will be individually tested before being deployed. Moreover, the platform will support the monitoring of individual components using established software components such as Prometheus [10]. Lastly, the platform maintainability was also considered during the design phase employing the following strategies:

1. Increased modularity: enables the removal of components as required in order to eliminate bugs and vulnerabilities. It also reduces the impact of code changes on other parts of the platform.
2. Capability enhancement: In order to facilitate capability enhancement all APIs of the platform provide descriptions using OpenAPI Specification [11].
3. Ease of shop-floor asset integration: Using the Asset Administration Shell approach and taking advantage of the AAS middleware features new assets can be introduced in the platform using modelling practices detailed in sections 2.1 and 2.2 along with adding configuration in the AAS middleware.

3. First implementations

3.1 Models

3.1.1 Data model

As a result of step 1 of section 2.1, the assets to be modelled for each use case inside the openZDM project were identified. Table 1 summarizes this information.

It is possible that other assets are also going to be modelled according to needs that may arise, so this table may be updated in the next stages of the project.

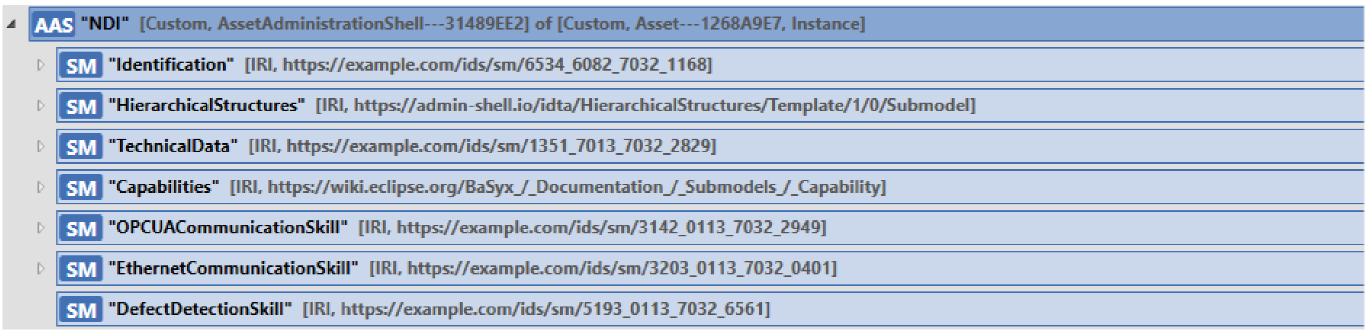
Table 1: Quantity of assets to be modeled per use case

Use Case	Assets identified
Trailing arm production	15
EV battery production	8
Vehicle body Shop and final assembly	5
Melamine surfaced board manufacturing	4
Bottle manufacturing	2

As an example, some submodels of an NDI belonging to one of the project use cases will be detailed below.

NDI AAS Submodels

In Figure 8 all the AAS submodels of an NDI developed in openZDM are shown. In the sections below, each submodel will be explained.

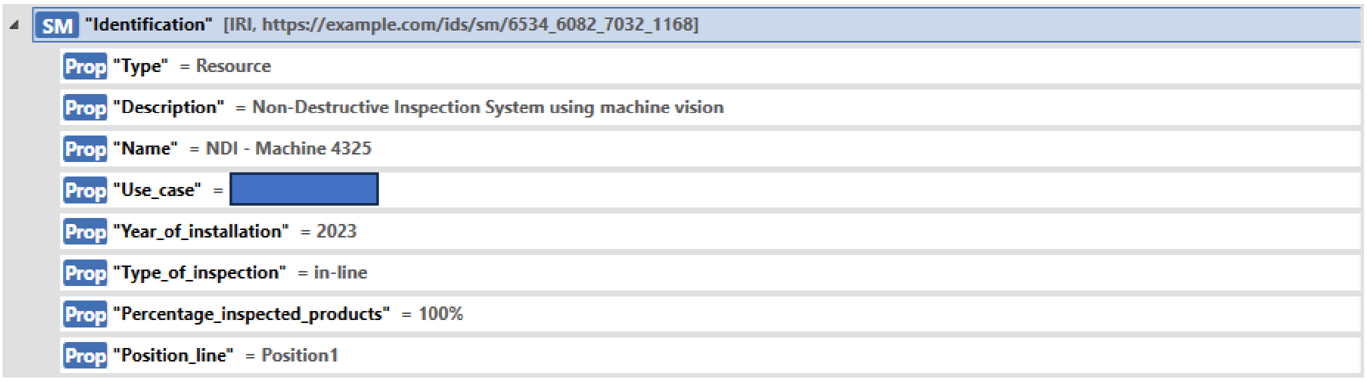


AAS "NDI" [Custom, AssetAdministrationShell---31489EE2] of [Custom, Asset---1268A9E7, Instance]
SM "Identification" [IRI, https://example.com/ids/sm/6534_6082_7032_1168]
SM "HierarchicalStructures" [IRI, https://admin-shell.io/idta/HierarchicalStructures/Template/1/0/Submodel]
SM "TechnicalData" [IRI, https://example.com/ids/sm/1351_7013_7032_2829]
SM "Capabilities" [IRI, https://wiki.eclipse.org/BaSyx_/_Documentation_/_Submodels_/_Capability]
SM "OPCUACommunicationSkill" [IRI, https://example.com/ids/sm/3142_0113_7032_2949]
SM "EthernetCommunicationSkill" [IRI, https://example.com/ids/sm/3203_0113_7032_0401]
SM "DefectDetectionSkill" [IRI, https://example.com/ids/sm/5193_0113_7032_6561]

Figure 8: openZDM developed NDI's AAS submodels

Identification submodel

The aim of this model is to provide information related to the asset, its name, the use case in which it participates, a brief description of the asset, the year of installation, and its location in the production line. The identification submodel can be seen in Figure 9.

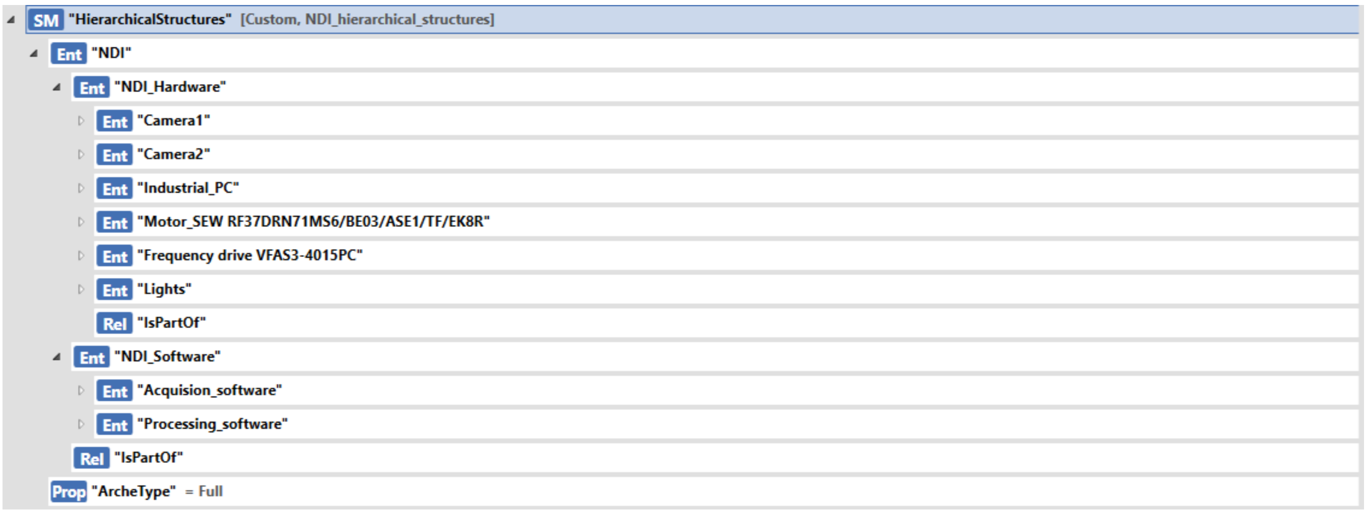


```
SM "Identification" [IRI, https://example.com/ids/sm/6534_6082_7032_1168]
    Prop "Type" = Resource
    Prop "Description" = Non-Destructive Inspection System using machine vision
    Prop "Name" = NDI - Machine 4325
    Prop "Use_case" = [redacted]
    Prop "Year_of_installation" = 2023
    Prop "Type_of_inspection" = in-line
    Prop "Percentage_inspected_products" = 100%
    Prop "Position_line" = Position1
```

Figure 9: Identification submodel

Hierarchical Structures Submodel

This submodel can be used to describe the components that make up the NDI and the relationships between them. In the openZDM project, vision-based NDIs are comprised of hardware and software. The hardware is composed of cameras and their accessories, PCs and other devices specific to each use case. The software comprises the acquisition and processing of the images. IDTA provides a template for this submodel [12], which can be seen in Figure 10.

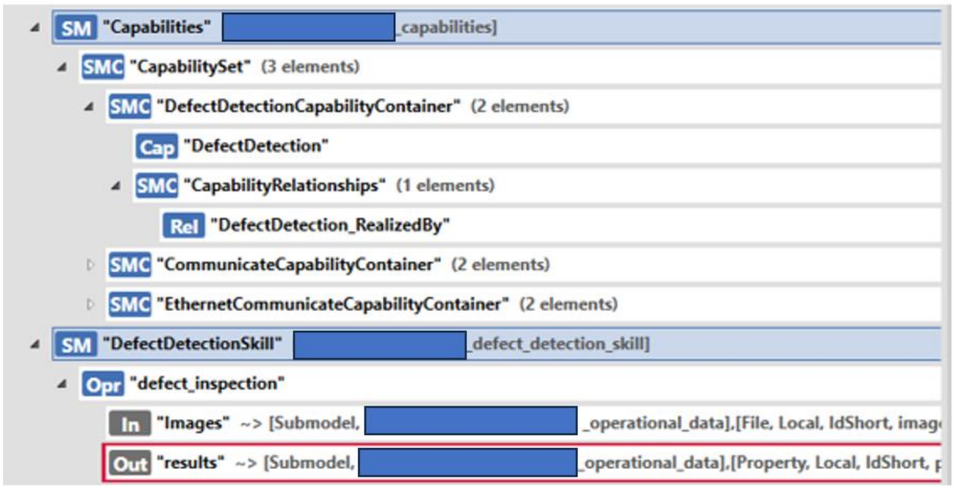


```
SM "HierarchicalStructures" [Custom, NDI_hierarchical_structures]
    Ent "NDI"
        Ent "NDI_Hardware"
            Ent "Camera1"
            Ent "Camera2"
            Ent "Industrial_PC"
            Ent "Motor_SEW RF37DRN71MS6/BE03/ASE1/TF/EK8R"
            Ent "Frequency drive VFAS3-4015PC"
            Ent "Lights"
            Rel "IsPartOf"
        Ent "NDI_Software"
            Ent "Acquision_software"
            Ent "Processing_software"
            Rel "IsPartOf"
    Prop "ArcheType" = Full
```

Figure 10: Hierarchical structures submodel

Capabilities and Skill Submodels

The capability submodel describes the capabilities that the asset can perform. The capabilities are implemented in the skill submodel by means of an operation that contains input and output operation variables. The input variables refer to the information required by the asset to implement its capability. While the output variables are the results of the implementation of the asset's skill. One of the NDI’s capabilities is defect detection, which is shown in Figure 11, by means of the DefectDetection capability. This capability is implemented in the skill submodel called Defectdetectionskills, in which the input operations are described as the images taken of the product and the input operation is the result of the evaluation of those images.



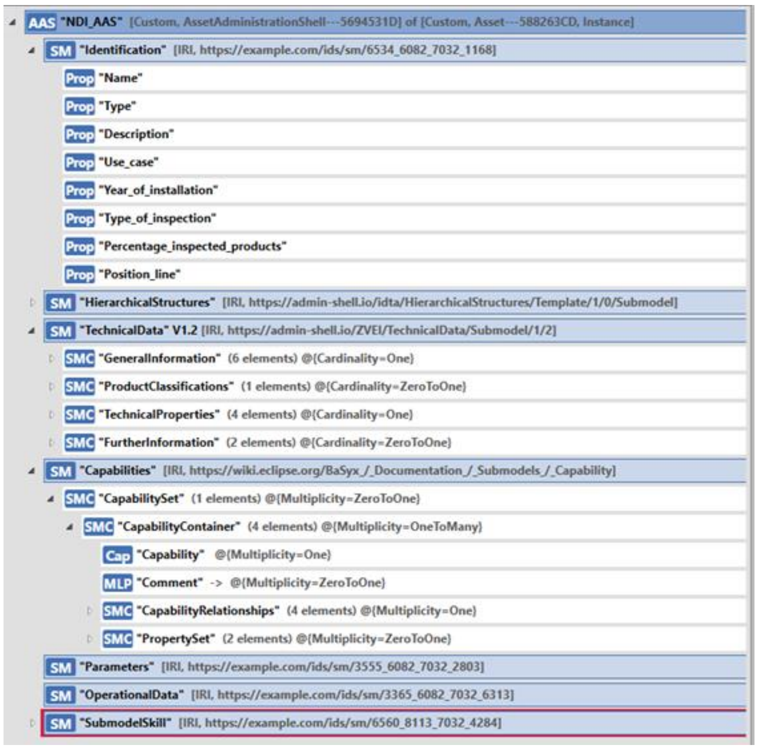
```
SM "Capabilities" [redacted]_capabilities]
  SMC "CapabilitySet" (3 elements)
    SMC "DefectDetectionCapabilityContainer" (2 elements)
      Cap "DefectDetection"
      SMC "CapabilityRelationships" (1 elements)
        Rel "DefectDetection_RealizedBy"
    SMC "CommunicateCapabilityContainer" (2 elements)
    SMC "EthernetCommunicateCapabilityContainer" (2 elements)
SM "DefectDetectionSkill" [redacted]_defect_detection_skill]
  Opr "defect_inspection"
    In "Images" ~> [Submodel, [redacted]_operational_data],[File, Local, IdShort, imag
    Out "results" ~> [Submodel, [redacted]_operational_data],[Property, Local, IdShort, p
```

Figure 11: Capabilities and skills submodels

3.1.2 Metadata model

A Common Information Model (CIM) is a shared semantic model focused on extracting value from data. The CIM organizes information about the managed environment through a set of classes with properties and associations that provide a well–understood conceptual framework. The CIM approach allows the construction of new models to represent new environments or systems requirements The AAS metamodel fits the CIM definition since the different domains of the asset or new environments in which it performs can be represented by new submodels, and therefore is used as CIM in the openZDM project.

Once the assets have been identified, the submodels that compose them are created based on a classification of the information provided by the assets. There are no mandatory sub-models for AAS, however, assets must be uniquely identifiable. In the approach developed in the openZDM project, a submodel called "Identification" is created, in which this unique identifier is modelled as a property called "Name" and value "assetname".



```
AAS "NDI_AAS" [Custom, AssetAdministrationShell---5694531D] of [Custom, Asset---588263CD, Instance]
  SM "Identification" [IRI, https://example.com/ids/sm/6534_6082_7032_1168]
    Prop "Name"
    Prop "Type"
    Prop "Description"
    Prop "Use_case"
    Prop "Year_of_installation"
    Prop "Type_of_inspection"
    Prop "Percentage_inspected_products"
    Prop "Position_line"
  SM "HierarchicalStructures" [IRI, https://admin-shell.io/idta/HierarchicalStructures/Template/1/0/Submodel]
  SM "TechnicalData" V1.2 [IRI, https://admin-shell.io/ZVEI/TechnicalData/Submodel/1/2]
    SMC "GeneralInformation" (6 elements) @{Cardinality=One}
    SMC "ProductClassifications" (1 elements) @{Cardinality=ZeroToOne}
    SMC "TechnicalProperties" (4 elements) @{Cardinality=One}
    SMC "FurtherInformation" (2 elements) @{Cardinality=ZeroToOne}
  SM "Capabilities" [IRI, https://wiki.eclipse.org/BaSyx_/_Documentation_/_Submodels_/_Capability]
    SMC "CapabilitySet" (1 elements) @{Multiplicity=ZeroToOne}
      SMC "CapabilityContainer" (4 elements) @{Multiplicity=OneToMany}
        Cap "Capability" @{Multiplicity=One}
        MLP "Comment" -> @{Multiplicity=ZeroToOne}
        SMC "CapabilityRelationships" (4 elements) @{Multiplicity=One}
        SMC "PropertySet" (2 elements) @{Multiplicity=ZeroToOne}
  SM "Parameters" [IRI, https://example.com/ids/sm/3555_6082_7032_2803]
  SM "OperationalData" [IRI, https://example.com/ids/sm/3365_6082_7032_6313]
  SM "SubmodelSkill" [IRI, https://example.com/ids/sm/6560_8113_7032_4284]
```

Figure 12: AAS template designed for NDIs of the openZDM Project

In the openZDM project, 11 NDIs are used in order to detect non-conformities in the production stages in the 5 use cases. These NDIs are mainly based on vision systems (composed of cameras and processing systems). To date, the IDTA has not standardized submodels for this type of asset, therefore in openZDM, the submodels are created with the information required to satisfy the use cases. For this purpose, a template has been designed in order to express the different NDIs in a uniform way (Figure 12).

The capabilities submodel describes the activities or operations that an asset is capable of performing. Through this submodel, an asset can make itself known in a distributed environment. The IDTA has standardized a submodel to specify the capabilities of an asset. The capabilities of an asset are realized by means of Skills, therefore, the AAS is composed of one or more submodels representing these skills. Skills sub-models are composed of operations with inputs (generally parameters or external information received by the asset) and outputs that can be operational data of the asset.

The asset modelling process must guarantee the interoperability of the modelled assets with other assets and other environments, therefore the structure of the submodels may change throughout the openZDM project in order to comply with standards specific to each asset domain. A list of the current standardised submodels can be found in Table 2. Additionally, an AAS creating the openZDM AAS template of a generic robot can be seen in Appendix I.

Table 2: Standardization status of the NDI asset submodels

Submodel Name	Standardized	Institution
Identification	NO	-
HierarchicalStructures	YES	IDTA
TechnicalData	YES	IDTA
Capabilities	YES	IDTA
Parameters	NO	-
OperationalData	NO	-
SkillSubmodel	NO	-

3.2 Applications

3.2.1 Digital Twin Toolset

openZDM's DTT application is focused on providing the ability of its user to create a new DT from scratch or load an already created DT. To realize the DTT application a full-stack architecture was utilized. In addition, the tools, frameworks, programming languages and databases used can be found in Table 3.

Table 3: Technologies used in the implementation of the DTT application

Type	Purpose
Programming languages	
JavaScript	JavaScript was used in the development of the UI of the application and for backend programming.
Python 3.9	Python, version 3.9, was used in the development of the data-driven and physics-based models.
Tools	
Node-RED	Node-RED has been integrated into the DTT application in order to provide the user with the ability to create the workflow of a DT.
Grafana	Grafana has been integrated into the DTT application in order to allow the user to create the dashboard of the DT.
Process Data Generator	The Process Data Generator node is used inside the Node-RED environment for process data generation.
Frontend frameworks	
React.js	React.js was the frontend framework used in order to build the UI of the DTT application.
Backend frameworks	
Express.js	Express.js was used as the backend server of the UI part of the DTT application
Flask	Flask was used as a backend framework responsible for handling HTTP requests between Node-RED and the data-driven models.
Databases	
InfluxDB	InfluxDB is part of the DTT application, allowing the user to save time-series data.
MongoDB	MongoDB is responsible for storing large files of various formats (.glb models, .pkl models etc) using GridFS.
Redis	Redis is responsible for storing incoming real-time data.
Other	
Three.js	Three.js is a frontend library and was used to create the component responsible for the 3D visualisation of an asset's .glb model.
CSS	CSS was used to style the UI of the DTT application.
HTTP	HTTP is a communication protocol which was used to communicate between the frontend and backend of the DTT application.
Docker	Docker as a containerization technology, was used for containerisation

In the DTT application, it all begins with the landing page of the app through which the user selects to either create or load a previously created DT. In order to create a DT, the user can select to retrieve the names of the assets that should be modelled either through the openZDM's platform AASs or through a manual definition process. To define an asset, apart from its name the application, requires the user to provide the data-driven model (in a .pkl format) that describes its behaviour, provide its .glb 3D model for visualising the asset and insert the MQTT topics where real-time data is published to in the openZDM platform's MQTT broker. This can be seen in Figure 13.

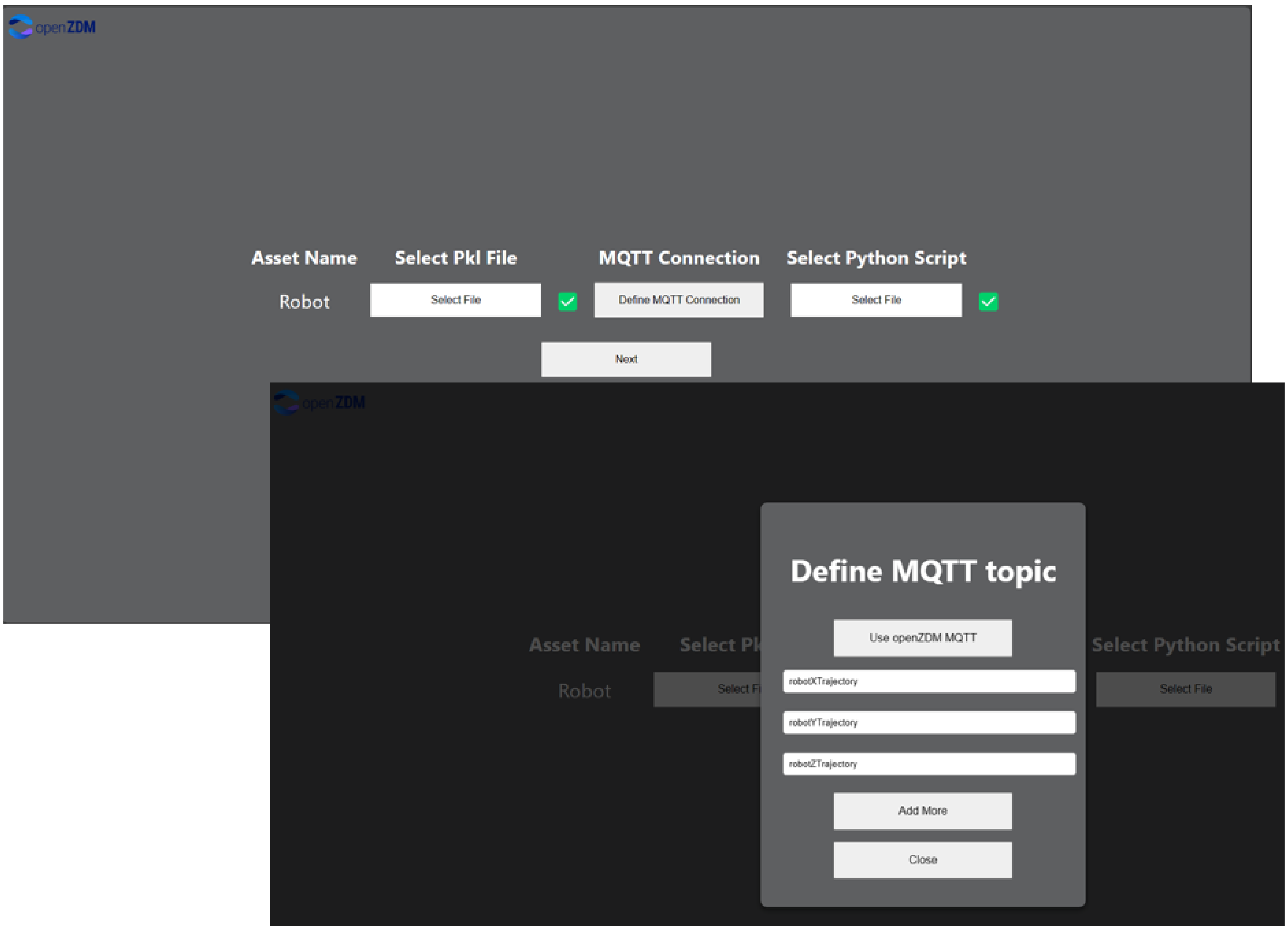


Figure 13: Asset definition component of the DTT application

The application provides the user with the capability to insert new communication topics to the MQTT broker and the application subscribes to the defined by the user MQTT topics, with a unique key is created for each subscribed topic in the Redis database. Inside each key, the corresponding real-time data is saved. Furthermore, a crucial step to create a DT inside the DTT application is the workflow definition which is handled by Node-RED, where the user connects the data-driven model with the real-time data from the Redis database and builds any additional data analytics functionality on top of it as shown in Figure 14 where the workflow of a robot has been created. Following the workflow definition, the user needs to set up the 3D visualization scene using the respective component built using Three.js and the DT dashboard using the Grafana component as can be seen in Figure 15.

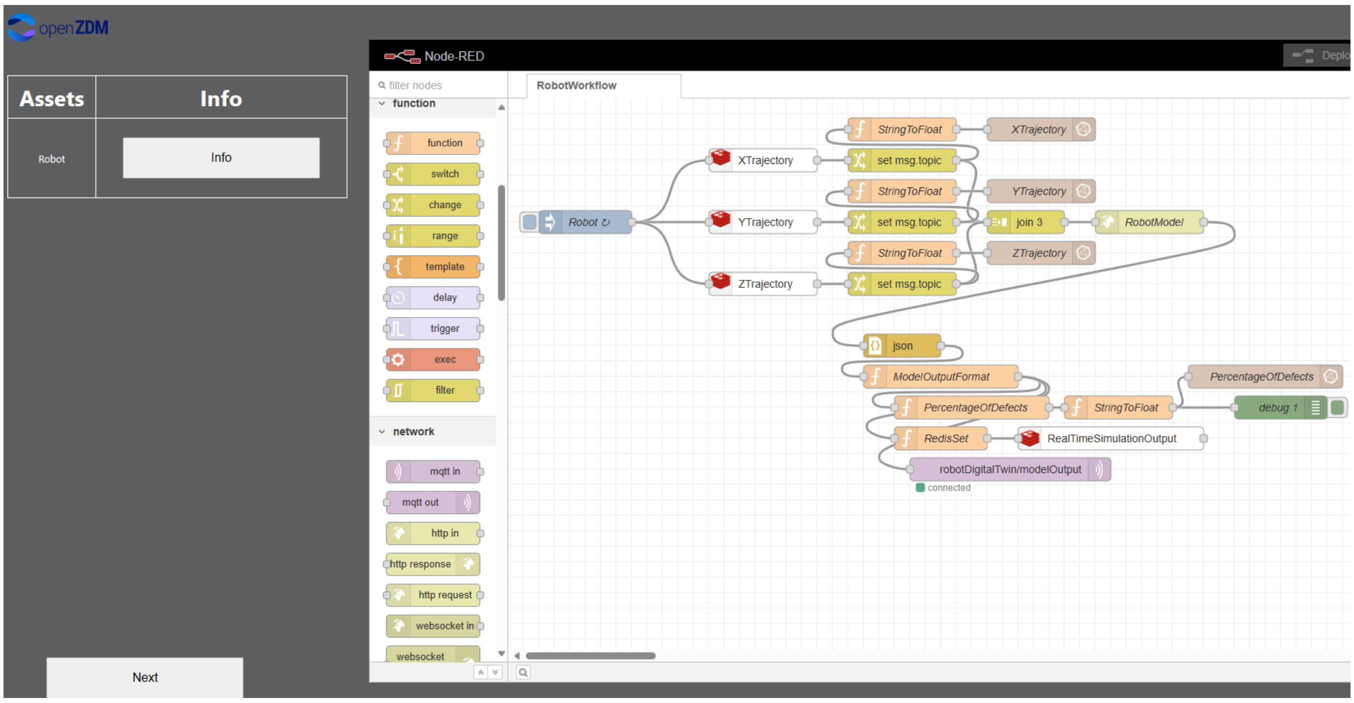


Figure 14: The workflow definition component in the DTT application

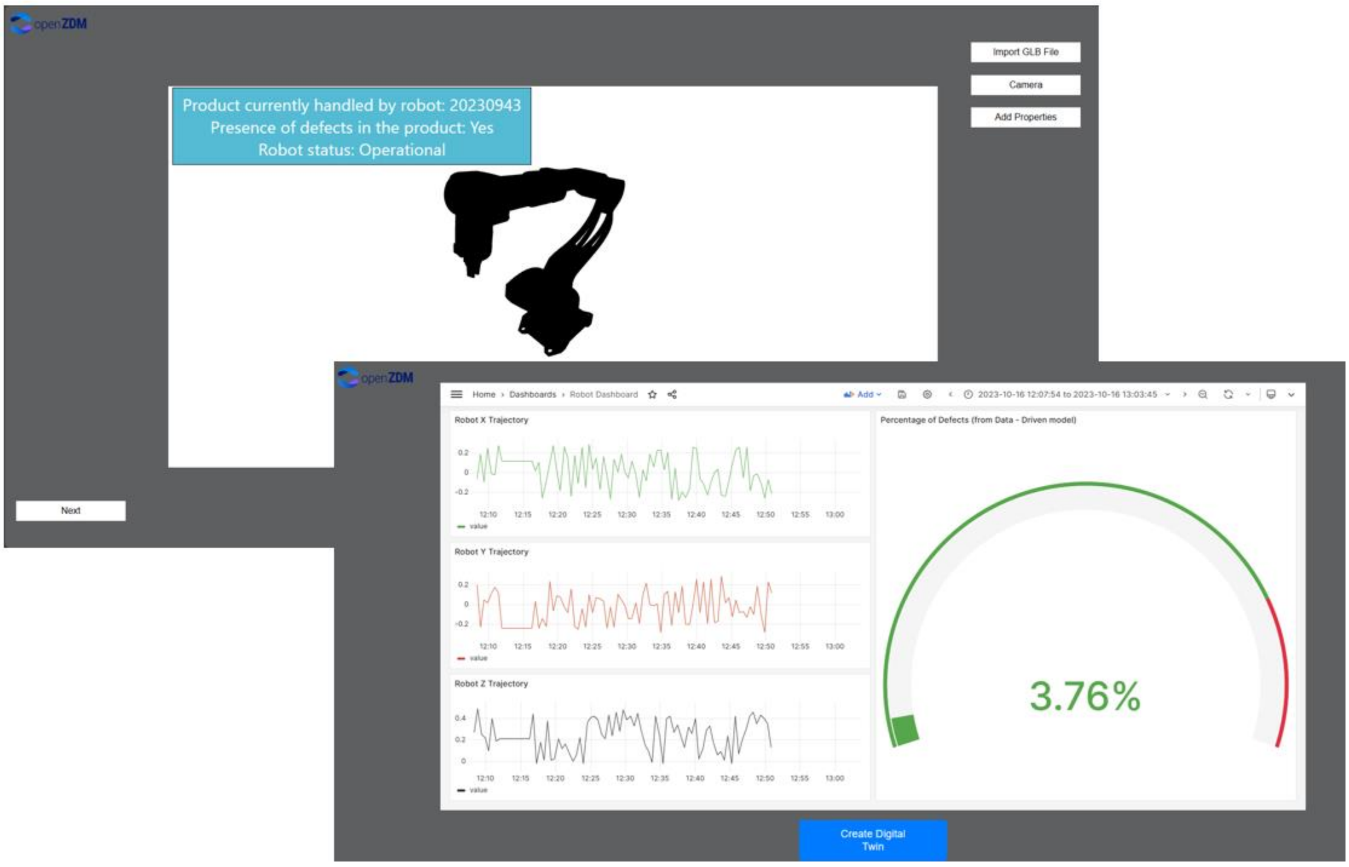
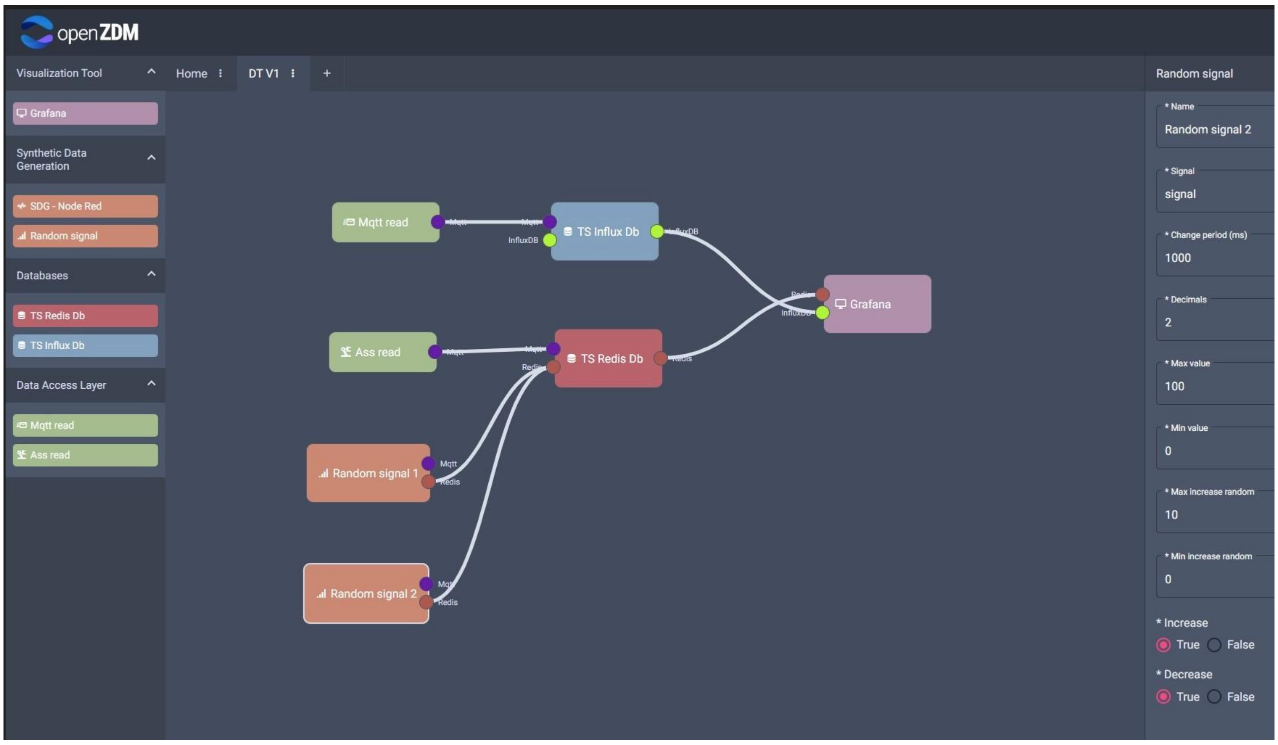

openZDM
Import GLB File
Camera
Add Properties
Product currently handled by robot: 20230943
Presence of defects in the product: Yes
Robot status: Operational
Next
Robot X Trajectory
Robot Y Trajectory
Robot Z Trajectory
Percentage of Defects (from Data - Driven model)
3.76%
Create Digital Twin

Figure 15: The 3D visualisation and dashboard creation component in the DTT application

Furthermore, the goal is to update the first implementation of the DTT to also include the configurator of the toolset. The configurator will be responsible for allowing the user to link all the different microservices required to build a DT according to his/her needs. Currently, an initial version of the UI of the configurator has been created and it provides a canvas that allows the user to connect some basic microservices as can be seen in Figure 16. Lastly, the DTT that is currently under development can be considered TRL 3. The plan for the DTT application is to demonstrate it in an operational environment making it TRL 7 by the end of the project.



Random signal

* Name: Random signal 2

* Signal: signal

* Change period (ms): 1000

* Decimals: 2

* Max value: 100

* Min value: 0

* Max increase random: 10

* Min increase random: 0

* Increase: [✓ True] False

* Decrease: [✓ True] False

Figure 16: The DTT's configurator

3.2.2 Data-driven Quality Assessment Modules

The data-driven quality assessment modules are responsible for providing a set of tools for monitoring and analysing collected shop floor data using the methodologies presented in section 2.4. The modules development was conducted with Python and were containerized using Docker. To connect and run the different modules, a choreography approach has been followed as depicted in Figure 17 where several modules are connected to each other to form a service that provides a form of descriptive analytics with temperature data.

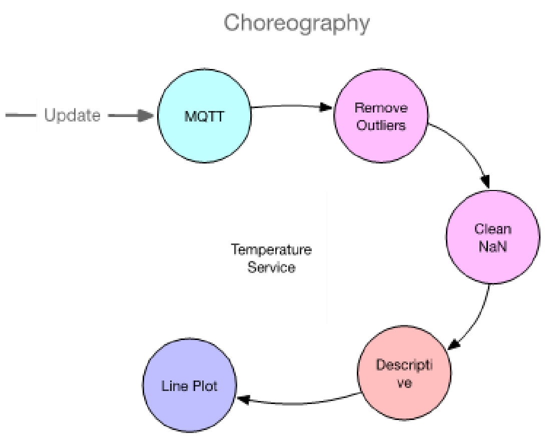


Figure 17: Choreography approach adopted for the connection and execution of data-driven quality assessment modules

A user can access the connected modules from a web application developed using JavaScript and the React.js framework. When the user accesses the application, he/she is presented with a landing page and an option to choose based on the two different methodologies, statistical and ML, in which under each methodology the capabilities of each module can be selected (monitor, prediction and process reconfiguration). In its current state, real-time data can be accessed through the openZDM MQTT broker, and in a similar manner as in the DTT application the user is requested to provide the MQTT topics where the module should subscribe to receive real-time data as it can be seen in the top image of Figure 18. As soon as the user has defined the MQTT topics, a selection between the available models to carry out the data analysis can be made as seen in the bottom image of Figure 18, where through real-time data and the selected RFR model, the user can make predictions and visualize them through graphs presented in the UI. Lastly, currently, the application is on TRL 3, with the intention to demonstrate it in an operational environment of TRL 7 by the end of the openZDM project.

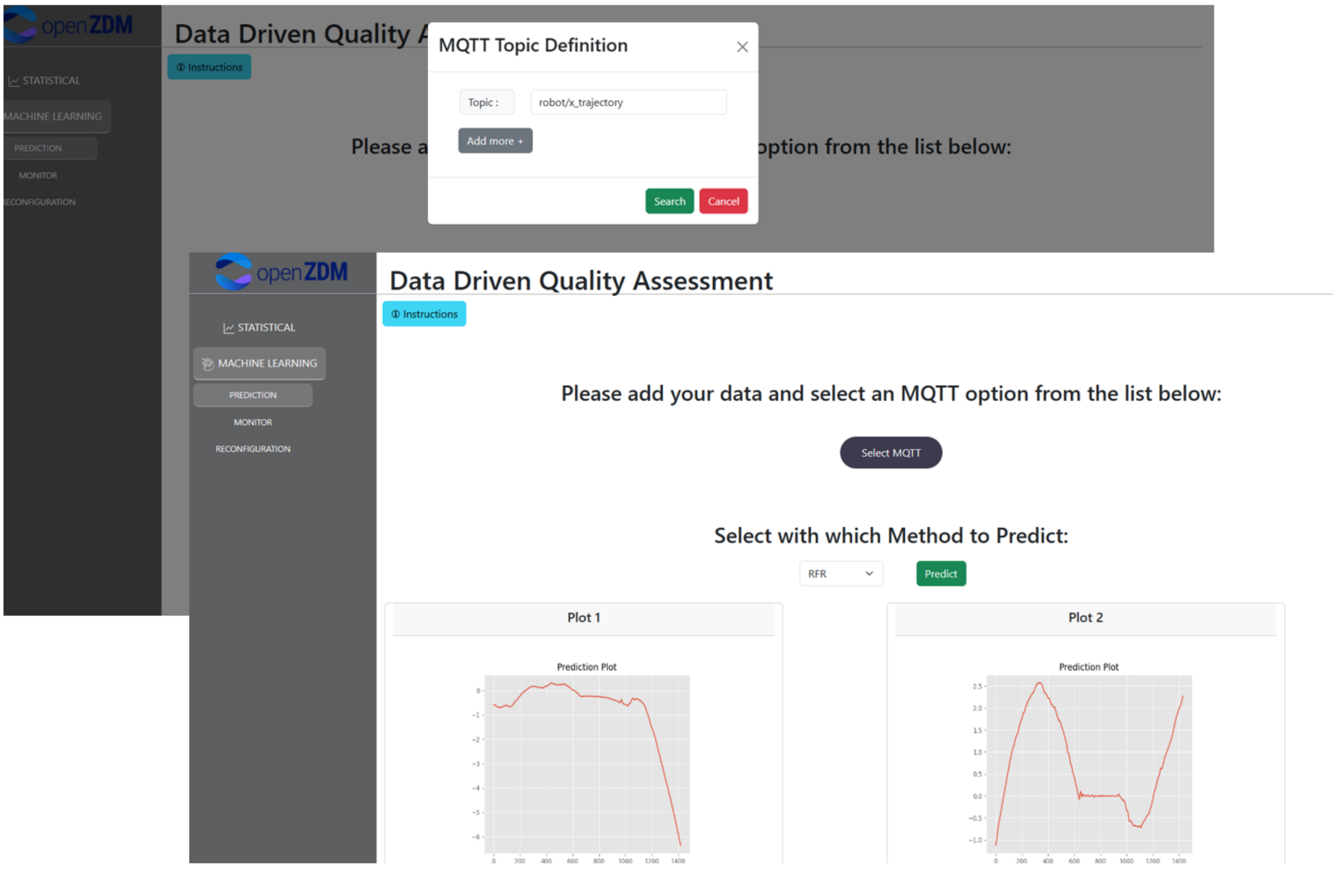


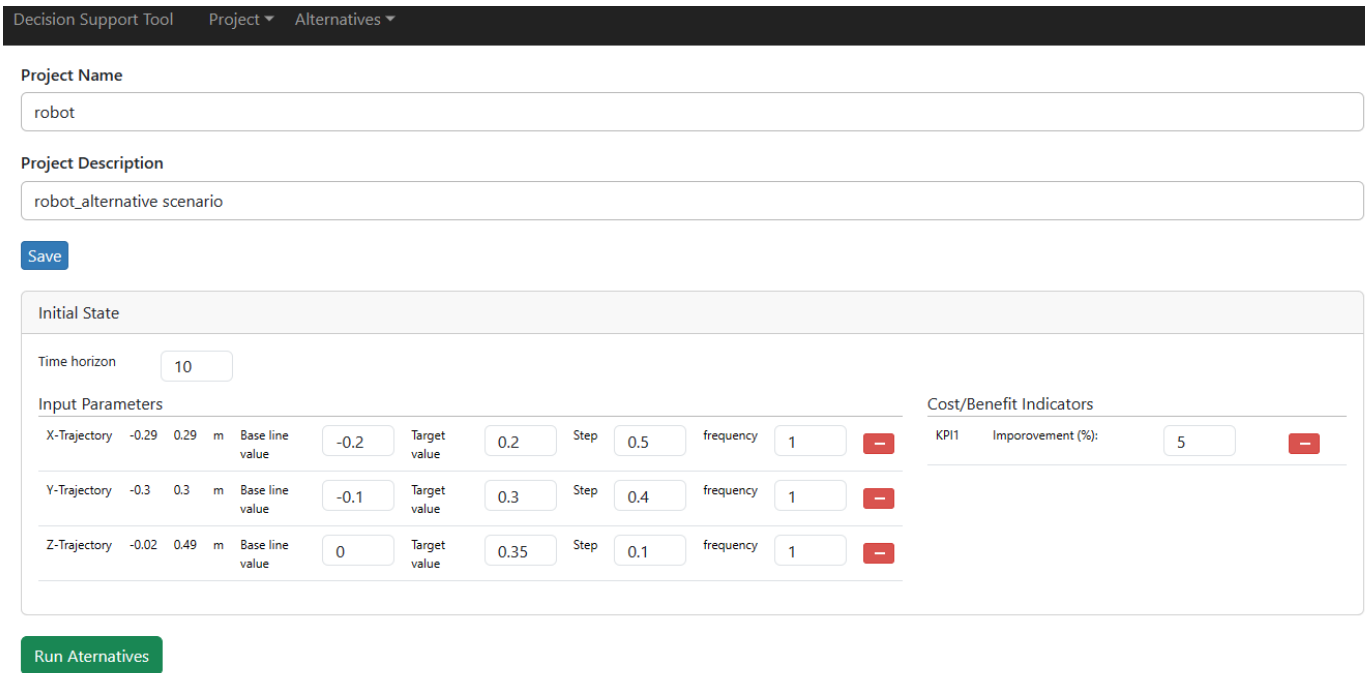
Figure 18: The UI of the data-driven quality assessment modules application

3.2.3 Decision support tool

The DST in openZDM is responsible for the generation, execution and evaluation of what-if scenarios. The backend implementation of the DST was handled with the use of Java and Spring. The UI of the tool has been implemented by using Jakarta Server Pages (formerly known as JavaServer Pages) in order to create the web page part of the application. In addition, the styling of the application has been handled with Bootstrap. Furthermore, the communication between the DST and the DTT uses standardised communication protocols, mainly HTTP requests. The same implementation approach was used in the communication between the different components of the DST (frontend and scenario evaluation component using the SLSQP algorithm). Lastly, a MySQL database was used in the DST for data storage and the application was containerized using Docker.

When accessing the application, the user is presented with the option of creating a new project or accessing an existing one from the DST database. When creating a new project, the user is automatically presented with the input parameters that can be included in the what-if scenario generation process taken from the DTT. The user inputs the minimum and maximum range of the search area for each feature, the step at which the input is increased or decreased during the search, and the frequency at which each input goes to its next step. Additionally, from the DTT, the cost/benefit indicators are taken and presented to the user who can insert a target value of improvement. The user also inputs the time horizon, which indicates how far in the future the desired improvement in the cost/benefit indicator should be achieved, and this can be seen in Figure 19 where the user defines the what-if scenarios to be executed for a robot. After the execution of the what-if scenarios, the results are returned to the UI and presented to the user. Lastly, in the DST application, the user is also capable of retrieving past what-if scenario results

Technologies used in the development of the DST are already available in the market, like the MySQL database used. However, the current version of the DST is on TRL 3 with the intention of getting to TRL 7 at the end of the openZDM project.



Decision Support Tool Project ▾ Alternatives ▾

Project Name

robot

Project Description

robot_alternative scenario

Save

Initial State

Time horizon 10

Input Parameters

X-Trajectory	-0.29	0.29	m	Base line value	-0.2	Target value	0.2	Step	0.5	frequency	1
Y-Trajectory	-0.3	0.3	m	Base line value	-0.1	Target value	0.3	Step	0.4	frequency	1
Z-Trajectory	-0.02	0.49	m	Base line value	0	Target value	0.35	Step	0.1	frequency	1

Cost/Benefit Indicators

KPI1	Imporovement (%):	5

Run Aternatives

Figure 19: The creation of an alternative scenario through the DST application

3.3 Platform

The platform provides the services that support all applications developed in the project (see chapter 3.2), the integration of shop floor assets, security mechanisms, and a user interface that acts as a single point of entry for the users. As illustrated in Figure 1 the platform services include data management and storage, error/event handling and notifications, user management, service orchestration and discovery, and the Asset Administration Shell middleware for integration with shop floor assets. As it is evident the different modules of the platform are deployed in a distributed fashion using several resources. Figure 20 illustrates a potential deployment of components. Server infrastructure I-IV represent physical resources but may be broken down into more than one physical resource. For example, the Business layer applications may each be deployed on separate physical resources. Edge devices are foreseen to be deployed remotely at the premises of the end users and are not considered part of the infrastructure of the platform.

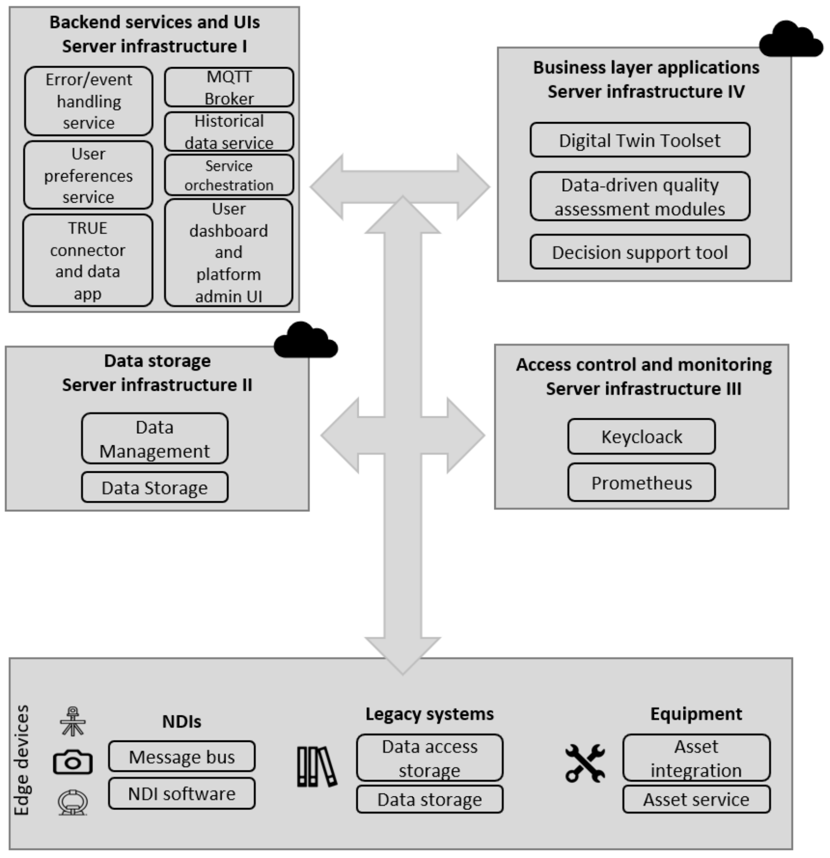


Figure 20: The openZDM platform deployment

The current version of the platform is deployed using virtual machines, following the configuration in Figure 20. Using this cloud approach enhances the platform's flexibility in upgrading the resources available to it (for example in Server infrastructure II more storage space can be added to facilitate the needs of the platform, or upgrading the hardware resources such as RAM, CPU, etc.). This approach also provides guaranteed levels of service, automated backups and data recovery mechanisms.

The main data flow of the platform comes from the edge layer. The main approach is to connect these devices with the AAS middleware. This can be achieved in several ways. The primary way used in openZMD is to publish information in the MQTT Broker this information is intercepted by the AAS middleware and fed into the AAS models. In addition, historical data is recorded both on value changes of the AAS models and the data published in the MQTT Broker. Another way to handle data from the edge layer is to use a connector deployed with the edge devices that are capable of monitoring data changes in a given asset and publishing these changes to AAS middleware using its standardized RESTful API. The AAS middleware provides additional integration capabilities such as utilizing OPCUA for integrating with shop floor assets.

The platform's data both historical and MQTT Broker messages are available to the Business layer applications. The historical information can also be shared using the dataspace approach by deploying an appropriate dataspace connector and data app. For development purposes, the IDS TRUE connector has been selected and an appropriate app will be developed to expose data of the platform to data spaces.

This platform's data is used to deliver monitoring, prediction, and other functionalities as detailed in section 3.2. All these functionalities are woven together into the platform user interface. All the services and user interfaces are secured using Keycloak [13] and the OAuth2 standard. Different authentication workflows are applicable based on the authentication scenario required. For example, user interfaces will redirect users to Keycloak to provide their credentials and then they are redirected back to the platform's user interface along with all relevant information required for getting appropriate authorization tokens for accessing backend services. In contrast, services may directly access authorization token required by other services using their own credentials. Keycloak also acts as the primary user management component of the platform.

The platform's users are organized into the roles of administrators and operators, each role having different access levels. In addition, the platform user interface is responsible for providing uniform access to all the other application user interfaces that provide specific functionalities for the users, such as the Digital Twin Toolset. Depending on the user role the user interface has a different structure as illustrated in Figure 21.

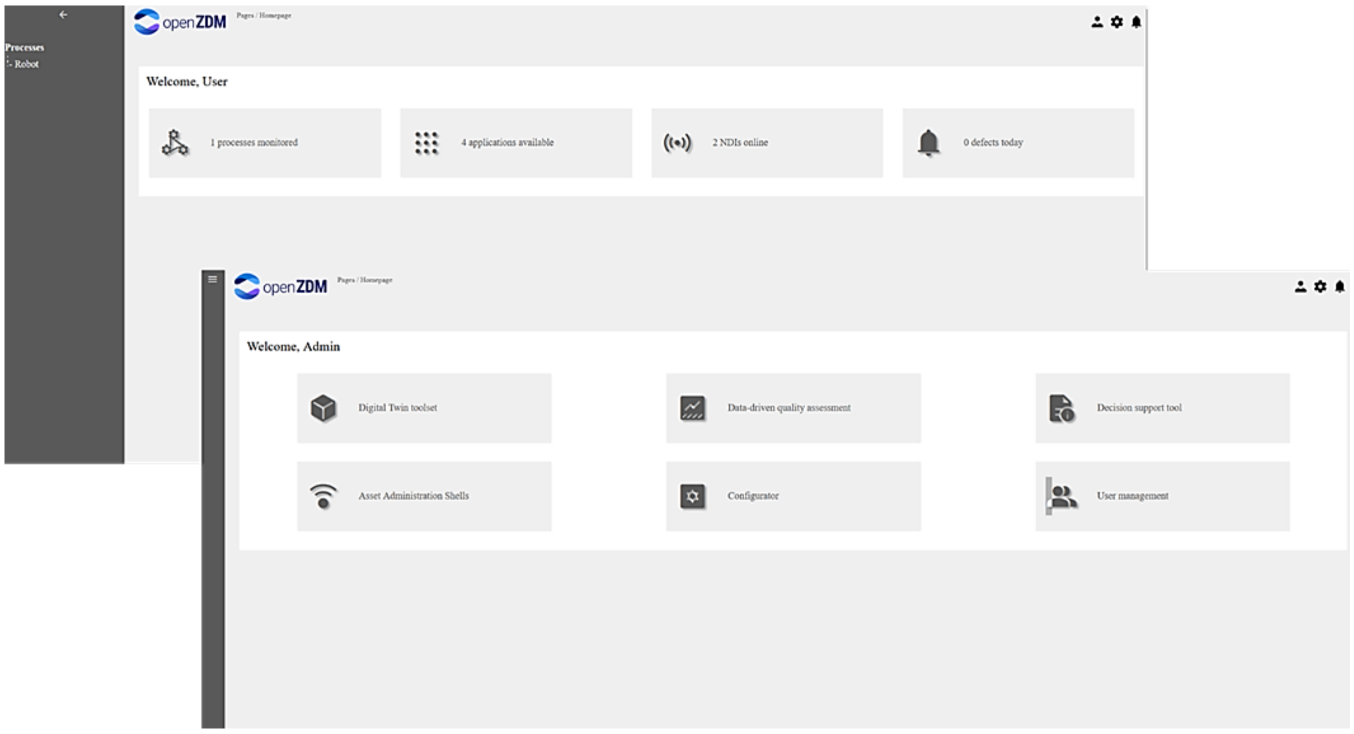
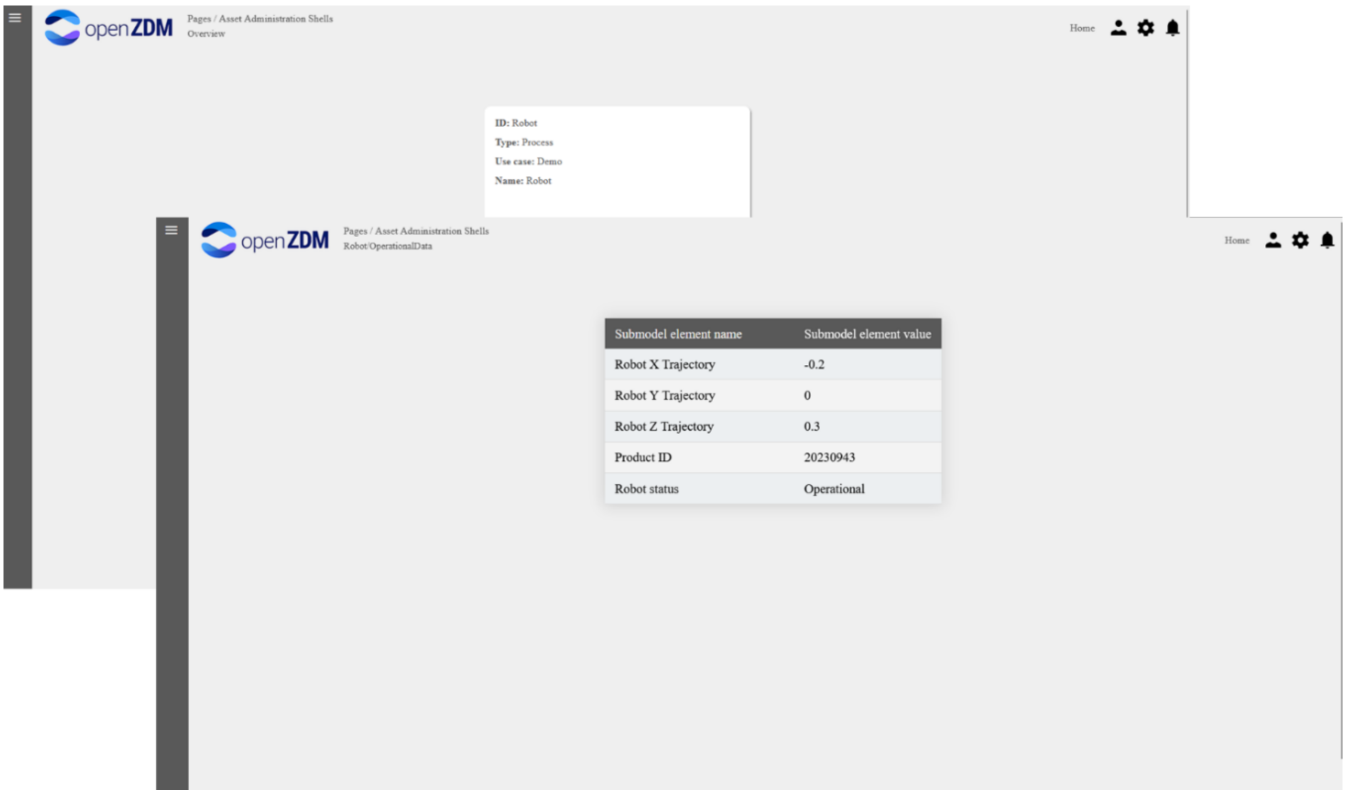


Figure 21: Role-based user interfaces of the openZDM platform

The administrator role (Figure 21 bottom right) has access to mainly configuration functionalities that control what is visualized to the operator users. These include management of applications (see the applications section 3.2), visualization of Asset Administrator Shells, and user management. On the top left of Figure 21 the operator, landing page is simplified to only provide some overview information. In order to achieve these functionalities, the user interface is integrated with Keycloak and is supported by a back-end service that retrieves data from the AAS components of the platform.

The existing AAS models, characterised as type-2 AASs, currently used in the openZDM platform are based on knowledge acquired from the DIMOFAC project, following the existing standards published by VDI/VDE [14], [15], [16], [17] and IEC [18]. These components provide the mechanism for registering new AAS, managing the AAS and their submodels, integrating them with data from shop floor assets, and sorting historical data.

While the AAS components are backend services their data can be accessed by the platform administrator. This role has access to all Asset Administration Shells that have been registered in the system. In the example of Figure 22: Viewing Asset Administration Shell registered in the platform as an administrator a single AAS representing the process carried out by a robot is registered in the system, and the values of the submodel named 'OperationalData' are also visualized. Operators also have access to the data of AAs for example in Figure 21 at the top left the processes menu is opened and the listed processes are reflected based on the AAS. Here AAS are filtered based on the user profile properties such as user group, users belonging to different user groups will have access to different AAS.

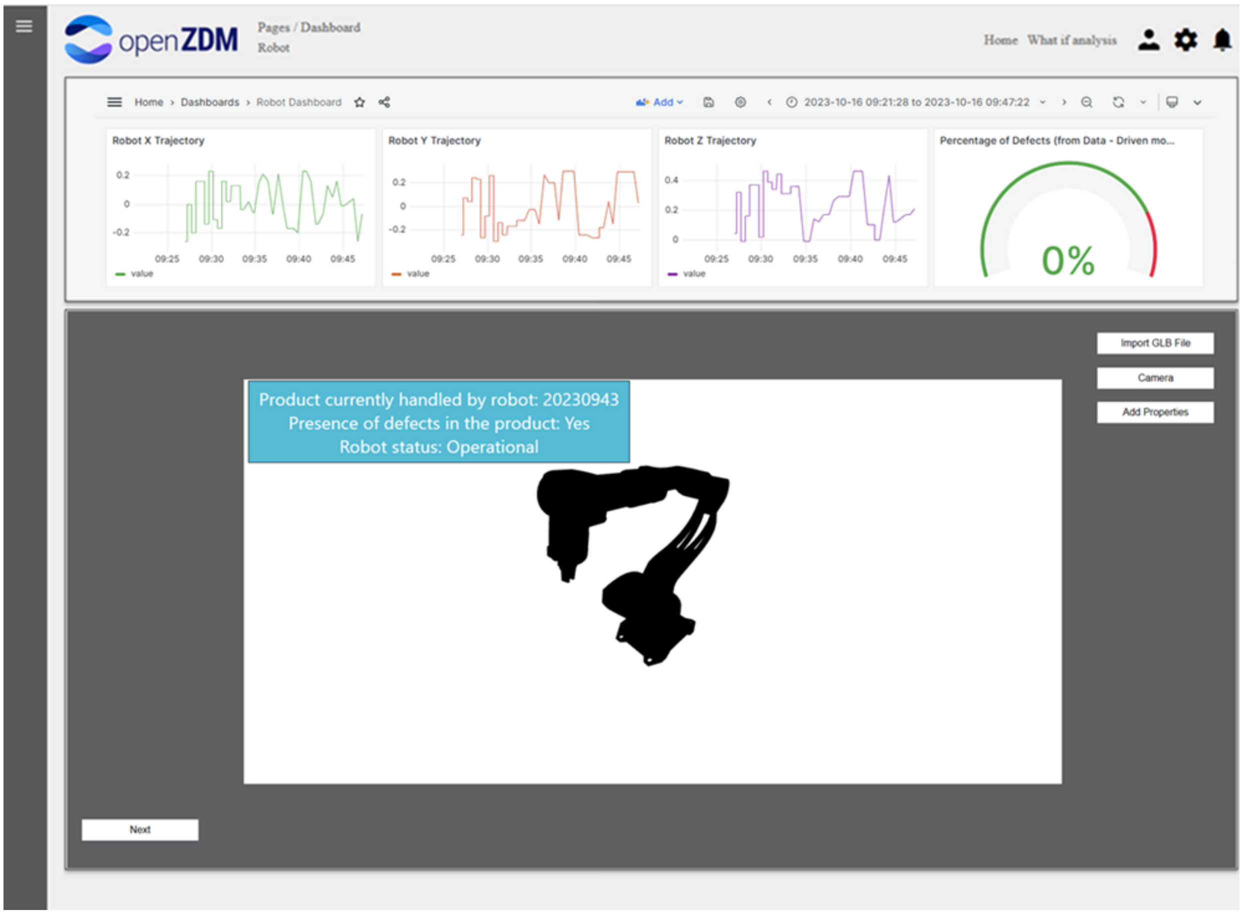

openZDM
Pages / Asset Administration Shells
Overview
Home
openZDM
Pages / Asset Administration Shells
Robot/OperationalData
Home

ID: Robot
Type: Process
Use case: Demo
Name: Robot

Submodel element name	Submodel element value
Robot X Trajectory	-0.2
Robot Y Trajectory	0
Robot Z Trajectory	0.3
Product ID	20230943
Robot status	Operational

Figure 22: Viewing Asset Administration Shell registered in the platform as an admin

The operator can use the side menu to access more details on the available processes by using specific applications. For example, in Figure 23 the Digital Twin application is loaded in the operator user interface.



Product currently handled by robot: 20230943
Presence of defects in the product: Yes
Robot status: Operational

Figure 23: Operator view with the DTT application opened

In order for the aforementioned functionalities to work the data of the robot executing a process is required. The process execution is simulated, and the data is fed to the platform's MQTT broker. Using an appropriate configuration, the AAS components of the platform intercept the data and feed into the process AAS registered in the platform. As new data are published the process AAS is constantly updated. The platform also stores the historical values of the process AAS. The same approach is utilized to connect the assets of the end-users in the platform where in the place of the simulated robot data is coming from industrial automation components, end-user services such as Azure IoT hub, and software developed in the project such as Non-Destructive Inspection components.

In terms of deployment, the platform consists of a mix of components that can be deployed separately, thus facilitating the scalability of the platform, and supporting different deployment scenarios. Currently, the platform is deployed on the cloud using Hetzner infrastructure and supporting components such as Non-destructive inspection components deployed on the shop floor of the end-users. In addition, the on-premise deployment has also been applied, where the platform components are deployed on the infrastructure of the end users.

As it is evident the platform consists of several components with a mix of 'Off-the-shelf' components and new components implemented during the project. For the individual components, several Technology Readiness Levels are applicable. For example, components such as Keycloak are available on the market whereas the user interface which is currently under development can be considered TRL 3. The plan for the platform as a whole is to be demonstrated in the operational environment making it TRL 7 by the end of the project.

4. Risks and Challenges

During the identification of the methodologies and their implementation risks and challenges were identified. To begin with, since AAS is in its early stages of maturity as a technology, the documentation around its standards and specifications is lacking, especially in the field of type 3 AAS. Fundamental aspects of the type 3 AAS implementation, such as if the contract negotiation is applicable for communication between assets of the shop floor with one another. Additionally, currently, a challenge that should be overcome in the future for the implementation of type 3 AAS is if type 3 is meant to be used as software for asset control. Through future research in the openZDM project and future specifications released from standardization authorities, type 3 AAS will be implemented by the end of the project.

Additionally, challenges were faced during the creation of pre-trained predictive Industry 4.0 software solutions (such as generic predictive analytical models) which were integrated into the data-driven quality assessment modules application. The main challenge derives from the complexity of data storage in most manufacturers, whose data was used to train the pre-trained algorithms and the security barriers that must be overcome before data can be accessed. Often, there is more than one service responsible for storing data, and data is typically securely stored, requiring specific authorization for access. This complexity in data storage poses a burden in the development of predictive Industry 4.0 software that relies heavily on high-quality historical data. Collecting data becomes more demanding due to the possible complexity of the data storage architecture used by manufacturers. Moreover, while tight data-sharing policies have their benefits, overcoming such security measures has proven to be a time-consuming process, increasing the development time required for predictive software. Lastly, in the context of developing predictive Industry 4.0 software solutions, another challenge was identified, while modern manufacturing heavily relies on acquiring as much data as possible from the shop floor, analysing this data to find correlations and patterns can be a time-consuming task.

Furthermore, an additional risk was identified during the creation of a hybrid DT model. The model relies heavily on the performance of the physics-based model. This poses a risk during the creation of the hybrid model since creating a physics-based model of adequate accuracy is not straightforward, especially when the under-study asset is comprised of complex physical phenomena.

5. Lessons learned

The adoption of the AAS as a data model in openZDM can provide valuable insights. A lesson that can be learned from this adoption is that while the AAS as a data model, in general, is currently in its early stages of maturity as a technology, it is a powerful way of increasing interoperability across manufacturers and companies. The adoption of AAS, especially the modelling of products as AASs, and through the inclusion of the product's lifecycle in specific AAS submodels, the collection and sharing of product data throughout its lifecycle is possible. However, the technical specification on how to properly implement it and its benefits against other approaches remain unclear, at least in the short term and considering manufacturers. Furthermore, the implementation of the type 3 AAS, as well as its reported characteristics for distributed peer-to-peer communication and the incorporation of an interaction protocol for bidding procedures are two aspects that are not clear on their benefits for shopfloor operations, as well as their technical implementation, is open to interpretation lacking clarity for the development teams.

Last, a point that will require particular attention is the service orchestration approach adopted by the openZDM platform. At the moment two different approaches are followed, a) choreography and b) the use of a central orchestrator. The selection of an approach or their mix will be a challenge in the implementation of the final openZDM system as well as the implementation of the other applications.

6. Conclusions

This deliverable provides an overview of the openZDM system, implemented by M18. A reference architecture and implementation are expected to be provided and further used in the context of exploitation activities, after being instantiated and deployed for testing and validation in 5 industrial use cases.

The main components have been outlined with the methodologies followed as well as the first implementations of those components. The combination of those components, NDIs, and external systems, through the openZDM platform, is expected to provide manufacturers with a set of tools to identify and assess the quality of their production lines and proactively control them towards reducing defects and improving the sustainability of their lines.

Future work will aim at the mitigation of the identified risks and challenges and will build on top of the lessons learned to deliver the updated version of the openZDM platform and its integrated components, with respect to the project's workplan.

7. References

[1] Home - DIMOFAC [WWW Document], n.d. URL https://dimofac.eu/ (accessed 12.4.23).

[2] GitHub - admin-shell-io/aasx-package-explorer: C# based viewer / editor for the Asset Administration Shell [WWW Document], n.d. URL https://github.com/admin-shell-io/aasx-package-explorer (accessed 12.4.23).

[3] What is Industrie 4.0? [WWW Document], n.d. URL https://www.plattform-i40.de/IP/Navigation/EN/Industrie40/WhatIsIndustrie40/what-is-industrie40.html (accessed 12.4.23).

[4] DIN SPEC 91345:2016-04 “Referenzarchitekturmodell Industrie 4.0 (RAMI4.0) / Reference Architecture Model Industrie 4.0 (RAMI4.0) / Modèle de reference de l’architecture de l’industrie 4.0 (RAMI4.0)”, ICS 03.100.01; 25.040.01; 35.240.50, April 2016. [Online]. Available: https://www.beuth.de/en/technical-rule/din-spec-91345-en/250940128

[5] 2 S. Bader, E. Barnstedt, H. Bedenbender, B. Berres, M. Billmann, and M. Ristin, “Details of the Asset Administration Shell - part 1 : the exchange of information between partners in the value chain of Industrie 4.0 (Version 3.0RC02),” Federal Ministry for Economic Affairs and Climate Action (BMWK), Berlin, May 2022. doi: 10.21256/zhaw-27075.

[6] S. Malakuti, P. Juhlin, J. Doppelhamer, J. Schmitt, T. Goldschmidt and A. Ciepal, "An Architecture and Information Meta-model for Back-end Data Access via Digital Twins," 2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Vasteras, Sweden, 2021, pp. 1-8, doi: 10.1109/ETFA45728.2021.9613724.

[7] “Asset Administration Shell Specification - Part 1: Metamodel Schema”.

[8] “ISO 23247-1:2021(en), Automation systems and integration — Digital twin framework for manufacturing — Part 1: Overview and general principles.” Accessed: Nov. 20, 2023. [Online]. Available: https://www.iso.org/obp/ui/en/#iso:std:iso:23247:-1:ed-1:v1:en

[9] Interoperability at Runtime – Exchanging Information via Application Programming Interfaces (Version 1.0RC02)

[10] Prometheus, “Prometheus - Monitoring system & time series database.” Accessed: Dec. 08, 2023. [Online]. Available: https://prometheus.io/

[11] “OpenAPI Specification - Version 3.0.3 | Swagger.” Accessed: Dec. 08, 2023. [Online]. Available: https://swagger.io/specification/

[12]“submodel-templates/published/Hierarchical Structures enabling Bills of Material/1/0 at main · admin-shell-io/submodel-templates,” GitHub. Accessed: Nov. 24, 2023. [Online]. Available: https://github.com/admin-shell-io/submodel-templates/tree/main/published/Hierarchical%20Structures%20enabling%20Bills%20of%20Material/1/0

[13]Keycloak [WWW Document], n.d. URL https://www.keycloak.org/ (accessed 12.4.23).

[14]VDI/VDE 2193 Blatt 1:2020-04 Language for I4.0 Components - Structure of messages

[15]VDI/VDE 2193 Blatt 2:2020-01 Language for I4.0 components - Interaction protocol for bidding procedures

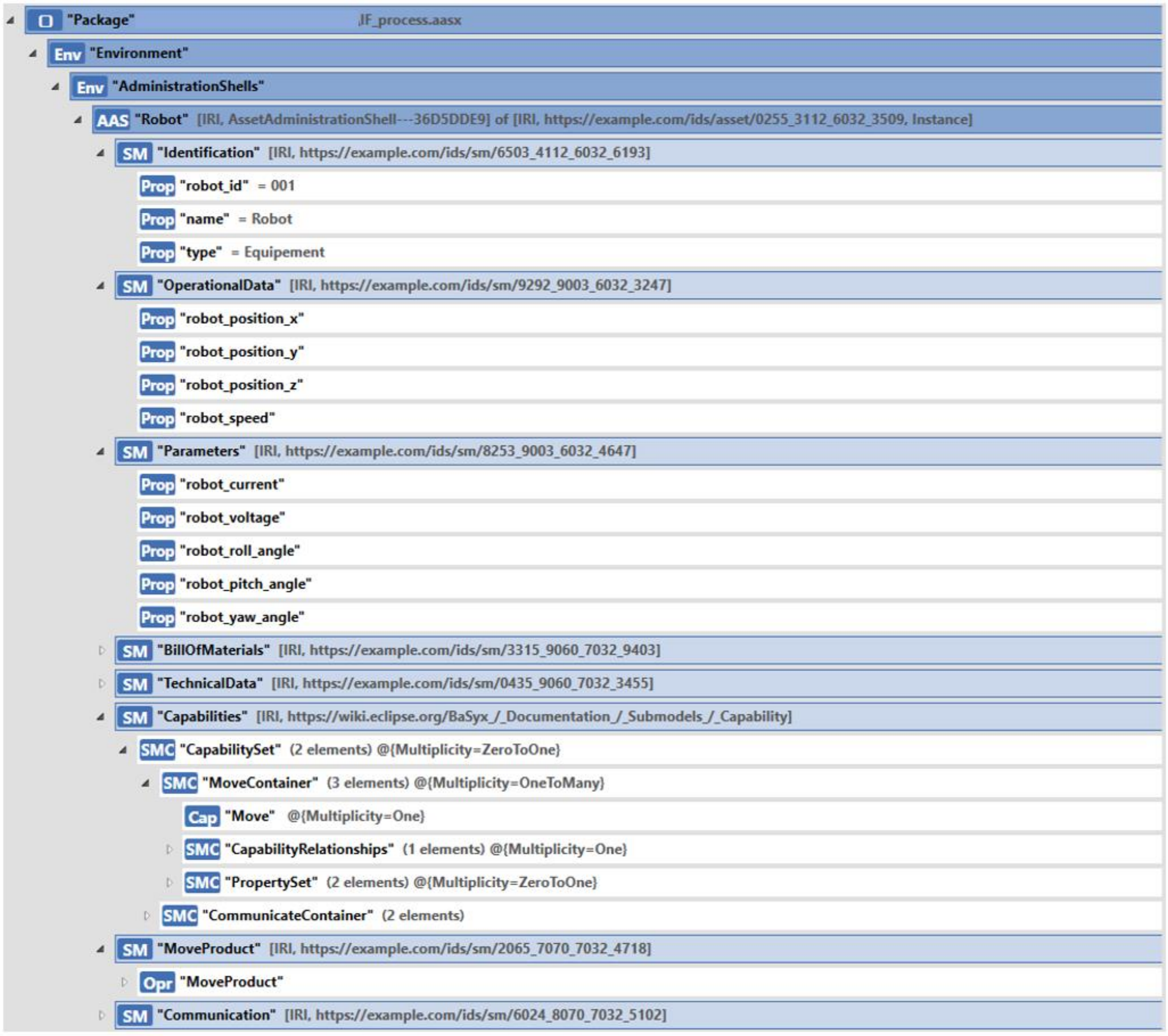
[16]VDI/VDE 4004 Blatt 1:2022-07 Testing of networked i4.0 systems - Rough planning of distributed test processes

[17]VDI/VDE 4004 Blatt 2:2022-12 - Draft Testing of networked I4.0 systems - Minimum requirements for the digital capture of planning information in distributed test processes

[18]OVE EN IEC 63278-1:2022-06-15 - Draft Asset Administration Shell for industrial applications - Part 1: Asset Administration Shell structure (IEC 65/925/CDV)

Appendices

Appendix I The AAS of a robot created using the openZDM template



```
◢ "Package"   .IF_process.aasx
  ◢ Env "Environment"
    ◢ Env "AdministrationShells"
      ◢ AAS "Robot"  [IRI, AssetAdministrationShell---36D5DDE9] of [IRI, https://example.com/ids/asset/0255_3112_6032_3509, Instance]
        ◢ SM "Identification"  [IRI, https://example.com/ids/sm/6503_4112_6032_6193]
            Prop "robot_id"  = 001
            Prop "name"  = Robot
            Prop "type"  = Equipement
        ◢ SM "OperationalData"  [IRI, https://example.com/ids/sm/9292_9003_6032_3247]
            Prop "robot_position_x"
            Prop "robot_position_y"
            Prop "robot_position_z"
            Prop "robot_speed"
        ◢ SM "Parameters"  [IRI, https://example.com/ids/sm/8253_9003_6032_4647]
            Prop "robot_current"
            Prop "robot_voltage"
            Prop "robot_roll_angle"
            Prop "robot_pitch_angle"
            Prop "robot_yaw_angle"
        ▷ SM "BillOfMaterials"  [IRI, https://example.com/ids/sm/3315_9060_7032_9403]
        ▷ SM "TechnicalData"  [IRI, https://example.com/ids/sm/0435_9060_7032_3455]
        ◢ SM "Capabilities"  [IRI, https://wiki.eclipse.org/BaSyx_/_Documentation_/_Submodels_/_Capability]
          ◢ SMC "CapabilitySet"  (2 elements) @{Multiplicity=ZeroToOne}
            ◢ SMC "MoveContainer"  (3 elements) @{Multiplicity=OneToMany}
                Cap "Move"  @{Multiplicity=One}
              ▷ SMC "CapabilityRelationships"  (1 elements) @{Multiplicity=One}
              ▷ SMC "PropertySet"  (2 elements) @{Multiplicity=ZeroToOne}
            ▷ SMC "CommunicateContainer"  (2 elements)
        ◢ SM "MoveProduct"  [IRI, https://example.com/ids/sm/2065_7070_7032_4718]
          ▷ Opr "MoveProduct"
        ▷ SM "Communication"  [IRI, https://example.com/ids/sm/6024_8070_7032_5102]
```

Figure 24: The AAS model of a generic robot created using the openZDM AAS template.

openZDM