ZDMP: Zero Defects Manufacturing Platform

WP4: Technical Challenge: Requirements, Specifications, and Standardisation

D4.4a: Functional Specification - Vs: 1.0.3

Deliverable Lead and Editor: Tim Dellas, ASC

Contributing Partners: All Partners from WP 5,6,7,8, 9,10

Date: 2020-05

Dissemination: Public

Status: EU Approved

Abstract

This deliverable is the functional specification of ZDMP. The document is structured following the components identified in the architecture specification of ZDMP in D4.3a. This document is the basis for the technical specification, although iteratively build with it.

Grant Agreement: 825631

Document Status

Deliverable Lead	Tim Dellas, ASC
Internal Reviewer 1	Christian Melchiorre, SOFT
Internal Reviewer 2	Juan Pardo and Sandra Vilaplana Llin, CETECK
Internal Reviewer 3	Stuart Campbell, ICE
Type	Deliverable
Work Package	WP4: Technical Challenge: Requirements, Specifications, and Standardisation
ID	D4.4a: Functional Specification and Update
Due Date	2019-07
Delivery Date	2020-05 – This is an interim update to the submitted M9 version due to some issues detected
Status	EU Approved

Project Partners:

For full details of partners go to www.zdmp.eu/partners

Executive Summary

D4.4a Functional Specification is the resulting artefact of the task T4.4 Functional Specification. The task involved further specifying software tasks, which was started with T4.3 Architectural Principles and Design.

T4.4 focusses on the following aspects:

- Understanding the features provided by the base components of WP5-6 and the more specific features of the process quality and product quality components of WP7-8, which were broken into modules in T4.3
- Addressing the vApps (the applications to be developed on top of the ZDMP platform) in the same way as other components, as those will be created to solve the use cases in ZDMP
- Focussing for all the software parts on features, the dimensions What, Why, Who, Where and When and looking at which requirements are covered by a certain function

In summary, the functions of all necessary software modules needed to solve the use cases were detailed for the whole consortium to be evaluated and understood, and the main results of the work going into this massive document are the reflections and thoughts the describing consortium partners had in defining these functions and understanding the requirements documents. Still, this document remains as a functional catalogue to be further consulted throughout the project.

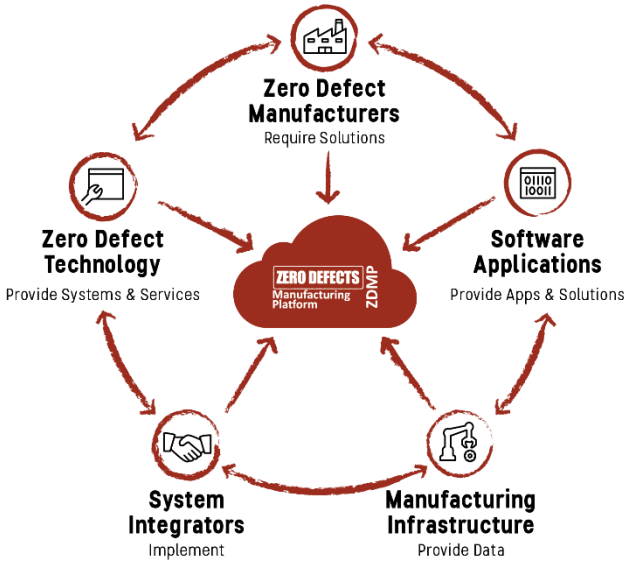
Table of Contents

0 Introduction

0.1 ZDMP Project Overview

ZDMP – Zero Defects Manufacturing Platform – is a project funded by the H2020 Framework Programme of the European Commission under Grant Agreement 825631 and conducted from January 2019 until December 2022. It engages 30 partners (Users, Technology Providers, Consultants and Research Institutes) from 11 countries with a total budget of circa 16.2M€. Further information can be found at www.zdmp.eu.

In the last five years, many industrial production entities in Europe have started strategic work towards a digital transformation into the fourth-industrial revolution termed Industry 4.0. Based on this new paradigm, companies must embrace a new technological infrastructure, which should be easy to implement for their business and easy to implement with other businesses across all their machines, equipment, and systems. The concept of zero-defects in the management of quality is one of the main benefits deriving from the implementation of Industry 4.0, both in the digitalisation of production processes and digitalisation of the product quality.



To remain competitive and keep its leading manufacturing position, European industry is required to produce high quality products at a low cost, in the most efficient way. Today, manufacturing industry is undergoing a substantial transformation due to the proliferation of new digital and ICT solutions, which are applied along the production process chain and are helping to make production more efficient, as in the case of smart factories. The goal of the ZDMP Project is to develop and establish a digital platform for connected smart factories, allowing to achieve excellence in manufacturing through zero-defect processes and zero-defect products.

ZDMP aims at providing such an extendable platform for supporting factories with a high interoperability level, to cope with the concept of connected factories to reach the goal of zero-defect production. In this context, ZDMP allows end-users to connect their systems (ie shop-floor and Enterprise Resource Planning systems) to benefit from the features of the platform. These benefits include product and production quality assurance amongst others. For this, the platform provides the tools to allow following each step of production, using data acquisition to automatically determine the functioning of each step regarding the quality of the process and product. With this, it is possible to follow production order status and optimise the overall processes regarding time constraints and product quality, achieving the zero defects.

0.2 Deliverable Purpose and Scope

The purpose of this document “D4.4 Functional Specification” is to closer define the functions and their dimensions What, Where, When, by Whom and Why in close resemblance to user stories, to closer inspect the stakeholder of the function, the context the function operates in, as well as timely and specially restrictions in terms of process (eg this function needs to be activated at the customer site close to the machine after a particular other function was executed by a worker with manager access rights). Also, a sequence diagram is used to define the main functions which are more complex in nature than a request response relationship, to understand the implications on other parts of the system.

Specifically, the DOA states the following regarding this Deliverable:

O4.4 To create a functional specification of all components

T4.4	**Functional Specification**			**ASC**	**M5-7, 46-48**
D4.4ab	Functional Specification and Update	R	CO	7, 48	RDI2 & 8

This task will deliver a Functional Specification document to provide an in-depth definition of the functionalities/behaviours of all ZDMP components. It will explain how related requirements will be fulfilled. It is an important means to measure the outcome of the individual tasks and the overall project. Interactions between ZDMP components will be detailed and depicted to guide the overall flow of functionalities and data between components. Components will be split into subcomponents and defined via a unified approach. Based on this, it will define the missing functionalities and implementation needs, which are the foundation for the work to be performed in the further RTD work packages as well as the Technical Specification.

0.3 Target Audience

The primary target for this document is the partners involved in technical WPs 5,6,7 and 8. However, it will be of wider relevance to all partners giving an overview of the project's components. Additionally, as this is a publicly available document, this document may be of use to the wider scientific and industrial community including other publicly funded projects and anyone interested in collaborations with ZDMP.

0.4 Deliverable Context

This document is a key deliverable in providing the architectural overview of the technical sections of the ZDMP project. Its relationship to other documents is as follows:

Primary Preceding documents:

- **D2.1 Vision Document:** The vision represents the consensus of the ZDMP partners and provides generic scenarios for which ZDMP should be able to implement
- **D2.3 Industry Scenarios and Use Cases:** This document provides specific use cases with respect to industry partners. The architecture needs to be applicable to be used in these industrial scenarios
- **D4.1 Requirement Specification:** Provides a list of requirements that were used in addition to the vision to help design the architecture
- **D4.2 User Mock-Ups:** Often considered part of the Functional Specification, in ZDMP mock-ups were created as an independent user-driven document
- **D4.3 Functional Specifications:** This document uses the components layouts and the architecture provided

0.5 Document Structure

This deliverable is broken down into the following sections:

- **Section 1: Context**: An introduction including a detailed description of documents that influence this document
- **Section 2: Functional Specification**: An overview of the targets of the Functional Specification and the description of the single sections explained for all components
- **Section 3: Developer Tier: Design-time**: The developer tier components: Application Designer, Security Designer, Data Harmonisation Designer, Orchestration Designer, AI-Analytics Designer and Process Optimisation Designer
- **Section 4: Enterprise Tier: Use-time**: The enterprise tier consists of higher-level components that facilitate the main zero defects functionality this includes Security Run-time, Autonomous Computing, Marketplace, Human Collaboration Application Run-time, and Inter-platform Interoperability
- **Section 5: Platform Tier: Run-time**: The platform tier consists of the main zero defects technology including Process Optimisation Run-time, Process Assurance Run-time, and Product Assurance Run-time. There are also core components that are needed to achieve the platform; Data Harmonisation Run-time, Monitoring and Alerting, Orchestration Run-time, AI-Analytics Run-time, Storage, and the Message Bus
- **Section 6: Edge Tier: Run-time**: This tier is for high-performance aspects of the zero defects manufacturing that must be located close to their data sources. It includes a Distributed Computing component to facilitate this and Data Acquisition, Digital Twin and Non-Destructive Inspection components for creating a zero defects solution
- **Section 7: zApps**: This section described the different zApps targeted to be built within WP9 and WP10 of ZDMP as a means of solving the use-cases on top of the ZDMP platform
- **Section 8: Conclusions**: An overview of the functional specification and further recommendations from this document

- Annexes:
  - **Annex A**: Document History

0.6 Document Status

This document is listed in the Description of Action as public as it has information that may be useful to a wider audience – for example other platforms such as project eFactory which will connect to elements of ZDMP or the subcall partners.

0.7 Document Dependencies

This document, D4.4a, is the initial version and main release of this document. An update, D4.4b, is produced at the end of the project. It will incorporate in-project changes (due to the reality of implementation) and be useful post project and in Zero Defects Limited as a reference.

0.8 Glossary and Abbreviations

A definition of common terms related to ZDMP, as well as a list of abbreviations, is available at http://www.zdmp.eu/glossary.

0.9 External Annexes and Supporting Documents

- None

0.10 Reading Notes

- None

0.11 Document Update

- The original M9 document was submitted and approved at the ZDMP 1st review. However, some minor issues were found and hence this document updated to maintain its coherency and because it is a public deliverable

1 Context

The ZMDP Project develops a Smart Zero Defects environment by deploying and networking an Intelligent and 'SME-friendly' Platform, Application Builder and Marketplace of developed functionality, applications, and services. The development of zero-defect solutions will help improve the quality (and profitability) within the manufacturing sector.

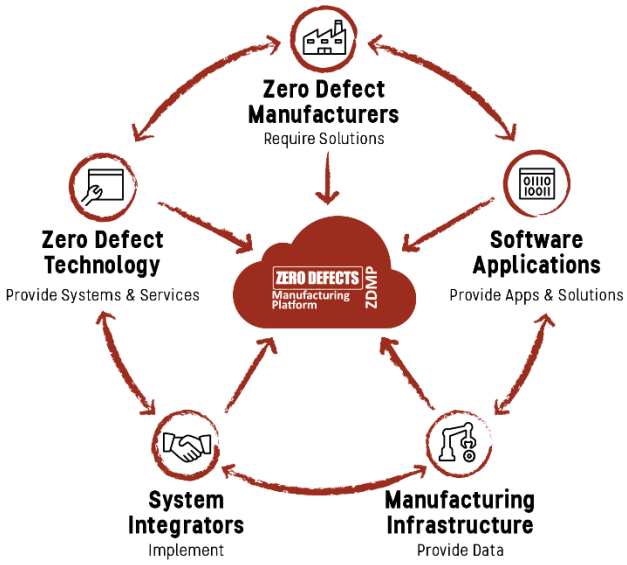


Figure 1: ZDMP Relevant Concepts

Figure 1 shows the main aspects that are linked by the platform. ZDMP allows for the linking of software applications, data, systems integrators, zero defect technology to manufactures. This link allows manufacturers to improve production by reducing defects and improving processes.

To put this document into perspective, this section will revisit the Inception and Vision document (D2.1), the Platform Benchmarking document (D2.2), the Industry Scenarios and Use-Cases document (D2.3), Manufacturing Reference Model Analysis document (D2.4), the Requirements document (D4.1) and User Mock-ups document (D4.2).

1.1 Existing ZDMP Deliverables

This document draws from many influences including the existing ZDMP documentation, such as the DOA, as well as discussions with partners and results from plenary and other meetings. The main influences have been previous documents from WP4: D4.1.a Requirements Definition, D.4.2.a User Mock-Ups, and D4.4a Architecture Definition, but also D2.2 Manufacturing Reference Model Analysis Document and D2.3 Industry Scenarios and Use Cases. Brief reminders of these documents are explained below but in-depth readers should examine them in greater detail.

1.1.1 D2.3 Industry Scenarios and Use Cases

D2.3 Industry Sector Scenarios and Use Cases is aimed to describe the functionality of ZDMP and of applications needed for each pilot. In turn, this then forms the basis of the specifications and requirements for the development of the ZDMP project. This document develops the characteristics of each of the four industrial scenarios: Automotive, machine tool, electronics, and construction.

The document describes how pilots are currently operated, what problems exist currently, as well as how they are expected to be improved with the new platform. In each pilot the user scenario and the problems to solve are identified. The pilot industrial cases were already summarized in the Architecture document.

The influence of the user's scenarios on this document is as follows:

- Giving context to the architecture regards to the ZDMP functions and stakeholders
- Describing the zApps to use as user facing applications, with their functions described in Section 7

1.1.2 D2.4 Manufacturing Reference Model Analysis Document

The D2.4, manufacturing and reference model, document makes recommendations for the architecture of this project. It recommends using IIRA and RAMI 4.0 architectural models [DIN+91345]. For the IIRA model, the document recommends using the "Implementation view". The component groups have been defined in the Architecture document and have similarly been used for the Functional Specification.

This document addresses specific recommendations from D2.4 Manufacturing Reference Model Analysis in the table of Figure 2. Note that many of these recommendations have been managed by the Architecture document D4.4.1 and only differences are explained.

Recommendation	Reference	Priority	Response
Use IIRA's Business, Usage, and Functional Viewpoints to refine the use case descriptions	IIRA	MEDIUM	The use-cases have played an important role in designing ZDMP, therefore this document describes the zApps described in the use-case documents as the technical representations of the use-cases in order to connect the use-cases to the rest of the architecture
Use IIRA implementation viewpoint to describe system implementation	IIRA	MEDIUM	This was addressed in Section 1.1.4 of the Architecture document and the structure was transferred to his document
Align the ZDMP high-level system architecture description to the three-tier IIoT architectural pattern	IIRA	HIGH	This was addressed in Section 1.1.4 of the Architecture document and the structure was transferred to his document
Use a multi-layered approach (such as the C4 model) to describe the system implementation at various levels of abstraction (eg context, containers, components, and code)	C4 model	MEDIUM	Level 1-3 are managed in the Architecture document D4.4.1. Level 4 is not outlined in document, but interfaces are given in the D4.5 Technical Specification, which is an online reference as it is subject to continuous change

Recommendation	Reference	Priority	Response
Identify data and control flows between containers, requirements, and interdependencies between physical and virtual connections at the container boundaries, also considered cyber-security standards (ISA 62443)	C4 model	MEDIUM	The connections between components were described in the Architecture document D4.4.1, but is extended in this document by presenting sequence diagrams for each of the components for all non-trivial interactions and including the zApps

Figure 2: Recommendation from D2.2

1.1.3 D4.1 Requirements Document

The ZDMP D4.1 Requirements Document identifies the main requirements of the project by analysing the use-cases.

D4.1a Requirements Annex is a living document and aims to assign responsibilities for these requirements. The requirements list labels requirement by ID and by zApps and components. It looks at dependencies of these requirements as well as prioritising them.

This document uses the insights gained from the Requirement Specification by:

- Validating the requirements by outlining the functions of the diverse components and zApps that will fulfil the named requirements
- Referencing the filled requirements in each function

1.1.4 D4.2 User Mock-ups Document

This document creates graphical designs for zApps and components. Although not expected to be the exact final designs of the user interfaces. It shows a vision of the platform and how it will be used. It brings together the complete picture of the ZDMP platform so that large numbers of partners can envision several types of users and how they will interact with the platform.

This document uses the following insights from the user-mock-ups:

- Visual understanding of the functions presented in this document

1.1.5 D4.3 Architecture Document

The Architecture document created the basic structure for this functional specification, as the separation between run time, design time, and the different tiers (Developer Tier, Enterprise Tier, Platform Tier and Edge Tier) were introduced.

The document uses the following insights from the Architecture:

- A structuring of the existing components

1.2 Future ZDMP Deliverables

Future deliverables of ZDMP will be directly influenced by this specification:

- **D4.4 Functional Specification Update:** As no upfront specification is perfect and can represent future reality realistically, this functional specification will be updated

towards the end of the project (month 48). It will thus represent changes to the functions that have occurred during the development of the project

- **D4.5 Technical Specification:** This is dynamic web utility to create a technical APIs for each component, interface, data models, and concrete data models. This will include the definitions of methods, parameters return values, and error handling to be used at the source code level. The specification takes the architecture and functional requirements as a basis and provides concrete interfaces between ZDMP software components, protocols, and class/packages structures, including definitions of methods and error handling together with the data models. It is derived from the Architecture document and Functional specification

2 Functional Specification

The current document is the ZDMP functional specification and it describes how the ZDMP platform will work from the user's perspective. As with any functional specification, this document does not deal with the technical aspects on how the software is implemented. Instead, it explains the features provided by the software, specifying its features and interactions. The mock-ups for the different UIs can be found in D4.2.

Technical aspects associated with the implementation are included in the architecture (D4.3), and the technical specification represents a living online interface specification (D4.5).

This functional specification is divided into the functionality and interactions that the user has with each ZDMP component, which are identified in parallel in the architecture definition (D4.3). The functional analysis per component and zApp has two perspectives each:

- **Behaviour and Functionality**: Containing a story map with the features and functionality offered and the user stories that need to be developed to implement that functionality
- **Interaction descriptions**: Describing for each component the set of interactions with other ZDMP components and users and describing the exchange of information flows critical for a unified ZDMP platform

In terms of behaviour and functionality, features describe the functionality at various levels of aggregation / abstraction.

The main description presents the feature of software from the point of view of the subject who expects the feature. The subject is not restricted to a 'human' ZDMP user (eg an operator or developer) and can be any entity with a behaviour, eg the component being described, another component, etc. Feature descriptions answer representative questions similarly to user stories, to capture in a simple table who wants what, when, where, and how the subject benefits from the feature. This enables the understanding of the context of the feature, as explained in the template below. Features include acceptance criteria – a checklist that determines when the feature is considered completed. The acceptance criteria are expressed from the point of view of the user listed in the Who-part and provides a detailed description of the criteria by which user stories should be evaluated and validated. Features have a unique ID (eg T55F001). Deadlines list the project month when the feature should be completed (eg M18, M24 and M30).

The functional specification of each component and zApp includes a subsection with the corresponding UML sequence diagrams, used to depict the interaction between the subcomponents and external components to the component under definition.

2.1 Responsibilities for the WP5-8 Components

zApps are containerised applications with restful backend interfaces and normally a user interface. The zApps are initially designed to solve the use-case as described in Industry Scenarios and Use Cases D2.3. However future zApps can be designed with the platform to provide new features using the T6.1 Application Builder and uploaded to the platform through the T6.2 Marketplace. These zApps can call on many of the ZDMP components as services.

Services are used through a RESTful API only and may only have User Interface for configuration purposes only. All services and zApps need to be designed to be (typically) small, short lived, and have well defined entry and exit points.

To apply this approach, all components implement and publish a REST interface allowing the exchange of data (primarily) with a messaging bus to be implemented within the project. ZDMP supports Event-Driven SOA features so that the different components can decide their interaction pattern and react to internal and external events. Following this approach, the components of ZDMP can behave either as services or as event producers and consumers.

Figure 3 shows a the ZDMP high-level architecture grouped into IIRA tiers with an additional Developer tier.

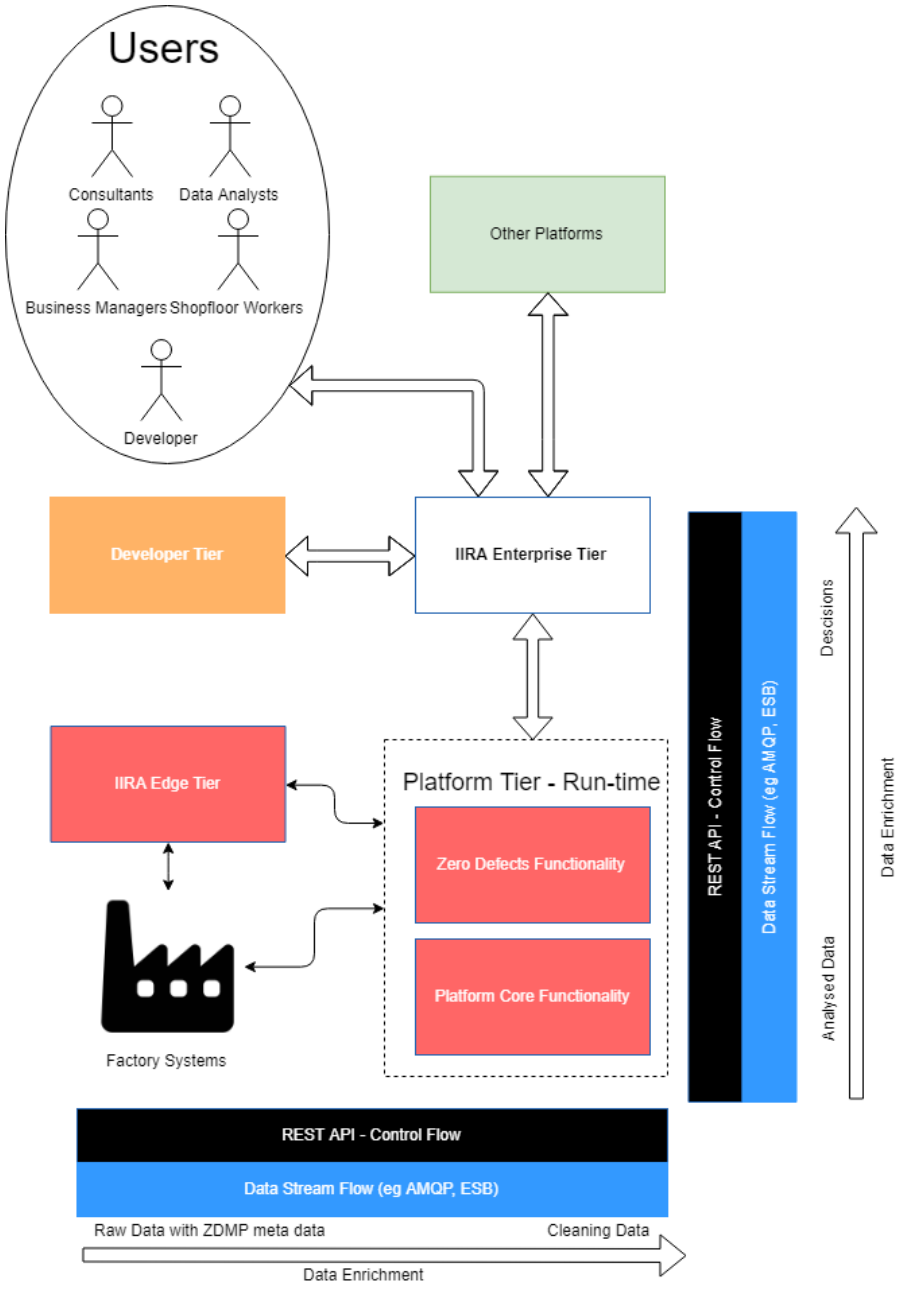


Figure 3: ZDMP High-Level Architecture

ZDMP is based on a federated architecture, which consists of several components split into the following architectural building blocks:

- **Developer Tier (Design-time):** These components aide in the production of containerised applications for zero defect manufacturing - zApps
- **Enterprise Tier (Use-time):** These components assist the run-time
- **Platform Tier (Run-time):** This is where zApps are installed. This component consists of run-time services to provide a base level functionality for ZDMP
- **Edge Tier (Run-time):** This is composed of the Distributed Computing component which allows certain zApps to be run at various locations of the system for performance gains. Many components can be run on the Edge Tier but for components where this is crucial to many of the use-cases they have been added to this section

All ZDMP components have been classified into these tiers. The table below shows the decomposition of ZDMP components together with the task ID they belong. A detailed description for each of the components depicted in Figure 4 can be found in Sections 3, 4, 5, and 6.

Task ID: Component Name	Lead Company
Developer Tier (Design-time)	
T5.3: Data Harmonisation Designer	ICE
T5.4: Orchestration Designer	ICE
T5.6: AI-Analytics Designer	SIVECO
T6.1: Applications Builder	ASC
T6.2: Security Designer	UOS-ITI
T7.1/T7.2/T7.3/T7.4: Process Quality and Optimisation Designer	PROF
Enterprise Tier (Use-time)	
T5.2: Secure Installation	IKER
T5.2 Secure Communication	ITI
T5.2 Secure Authentication/Authorisation	IKER
T6.2: Marketplace	SIVECO
T6.2: Storage	SIVECO
T6.3: Human Collaboration	SIVECO
T6.4: Portal	ICE
T6.4: Application Run-time	ICE
T6.5: Inter-platform Interoperability	ICE
Platform Tier (Run-time)	
T5.3: Data Harmonisation Run-time	ICE
T5.4: Orchestration Run-time	ICE
T5.4: Monitoring and Alerting	ASC
T5.5: Autonomous Computing	ASC
T5.6: AI-Analytics Run-time	SIVECO
T6.4: Service and Message Bus	SAG
T7.1/T7.2/T7.3: Process Optimisation Run-time	UPV
T7.4: Process Assurance Run-time	PROF
T8.2/T8.4: Product Assurance Run-time	ITI
Edge Tier (Run-time)	
T5.1: Data Acquisition	SOFT
T5.5 Distributed Computing	ASC
T7.3/T8.1: Digital Twin	CET
T8.3: Non-Destructive Inspection	VSYS

Figure 4: ZDMP Components

T6.6 General Cross-Task Integration and Improvement is not represented in this document. As its main deliverable is buffer time for integration of the components.

The global architecture and where each component fits within each tier. Each component has been given an icon as shown in Figure 5.

Figure 5: Icons for each component. The four rows represent the four tiers.

2.2 Responsibilities for zApps

The following table depicts the responsibilities for the single zApps:

ID	Name	Responsible WP / Partner
zA1.01	zAnomalyDetector	WP9 - ITI
zA1.02	zDigitalTwin	WP9 - ITI
zA1.03	zAlarm	WP9 - ITI
zA2.01	zMachineMonitor	WP9 - IKER
zA2.02	zMachineAnalytics	WP9 - IKER
zA2.03	zParameterMonitor	WP9 - IKER
zA2.04	zParameterAnalytics	WP9 - IKER
zA2.05	z3DScannerDriver	WP9 - IKER
zA2.06	z3DGenerator	WP9 - IKER
zA3.01	zXRAYMonitor	WP10 - UPV
zA3.02	zXRAYAnalytics	WP10 - UPV
zA3.03	zFeedbackMFT	WP10 - UPV
zA3.04	zArtificialIntelligenceMFT	WP10 - SIVECO (subresponsible: UPV)
zA3.05	zFeedbackAFT	WP10 - SIVECO (subresponsible: UPV)
zA3.06	zArtificialIntelligenceAFT	WP10 – UPV
zA3.07	zDriver	WP10 – UPV
zA3.08	zLineData	WP10 – UPV

zA3.09	zVisualManager	WP10 – UPV
zA3.10	zProductVersionControl	WP10 – UPV
zA3.11	zAutomaticCall	WP10 – UPV
zA3.12	zPowerManager	WP10 – UPV
zA3.13	zCycleTimeManager	WP10 – UPV
zA3.14	zAutomaticMaterialOrdering	WP10 – UPV
zA3.15	zDataArchiveControl	WP10 – UPV
zA4.01	zSteelSheetWidthMonitor	WP10, Uni (VSYS)
zA4.02	zHorizontalWeldDetection	WP10, Uni (VSYS)
zA4.03	zVerticalWeldMonitor	WP10, Uni (VSYS)
zA4.04	zShapeTubeMonitor	WP10, Uni (VSYS)
zA4.05	zWiresMonitoring	WP10 – UNI
zA4.06	zThicknessMonitor	WP10 – UNI
zA4.07	zDetectDefects	WP10, Uni (VSYS)
zA4.08	zWornOutBladeDectection	WP10 – UNI
zA4.09	zTilesCorformity	WP10, Uni (VSYS)
zA4.10	zRemoteQC	WP10 – UNI
zA4.11	zRescheduler	WP10 UPV
zA4.12	zMaterialTracker	WP10 – UNI
zA4.13	zMaterialID	WP10 – UNI

Figure 6: Partner Responsibilities for zApps

2.3 Template

This is an example for the table

ID / Function / Requirements filled	Function Metainformation
T61A9 **Get List of Stored Configurations**	**Priority:** Must **Who:** zApps Developer **When:** Design time in the app builder **Where**: Anywhere via a browser **What:** Lists existing SDK configuration files, getting their names, detail on functionality, version, etc **Why:** Browse existing configurations to model the SDK
Acceptance Criteria	Invoker got a structured list of configurations
Requirements Filled	RQ_0011, RQ_0017

Figure 7: Function Table Template

- **ID**: The identifier for each function should be built up of the task number (eg T61 for T6.1), then use a letter to identify the subcomponent (in this case A), which should be communicated to other partners working in the same task and describing functions; then append a number with three decimals, starting with 001, so that all functions have a unique ID. Examples: T61A001, T61R006, T61A088

- **ID for zApps**: The zApps have a unique identifier already, so the unique identifier for the functions of the zApps are combined of the zApp ID, a dot, and another number. For the zApp zA4.13, the functions should be named zA4.13.1, zA4.13.2, etc
- **Function Name**: A clear and descriptive name. Multiple features that are closely related may be combined (eg Create/Read/Update/Delete Stored Configuration)
- **Priority**: One of “Must”, “Should”, “Could”, which should at least represent the highest priority listed on the respective requirements. So, if a function fulfils two “Could” and one “Must” requirement, it should be “Must” itself. It might have a higher priority than the listed requirements, for example if it is necessary for another function not covered by the requirements
- **Where** is the feature used: Can it be used anywhere via a mobile zApp or web-browser (“Cloud layer”), does it have to be used on-site (“Fog layer”), in the same physical network or even close to a machine to minimise latency (“Edge layer”), or would it even be more useful if it were deployed on an IoT device (“Mist layer”)?
- **When** is it used, timing wise it might be used during *design* or *runtime*, and maybe when a specific phase or event occurs
- **Who** uses the feature, this means which role does the user have, might even be other components or zApps using APIs
- **What** is the feature?
- **Why** is the feature needed, what is the reason for using it?
- **Access Criteria:** Simple sentences describing what must happen when this function is used. If the described statement is true when this function was performed, the function works as expected. Multiple acceptance criteria can be put as a list of sentences

2.4 Workflows, Use-Case and Sequence Diagrams

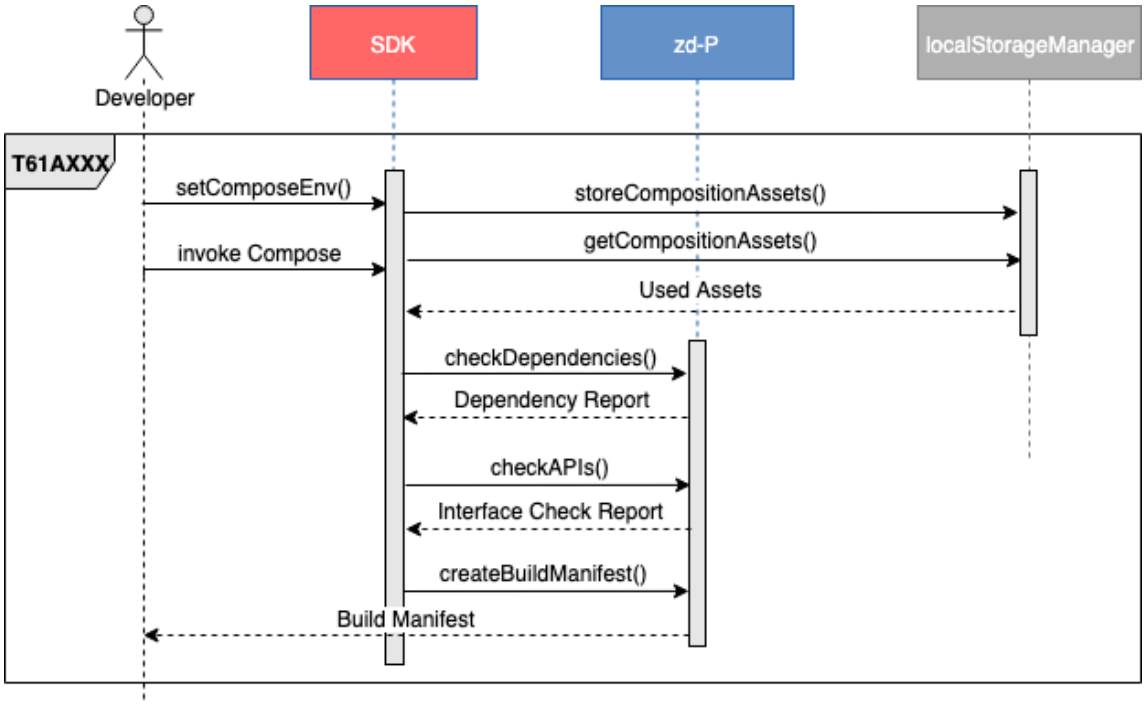


Figure 8: Example for function “Composing a zApp”

For all more complex functions, a sequence diagram or a use case diagram is designed to illustrate the workflows and interactions necessary between zApps and ZDMP components

to fulfil the function. More complex in this sense means that diagrams are not created for the sake of it but to describe functions that need more complex interactions and are therefore not self-explanatory.

Sequence Diagrams and Use-Case-Diagrams are translatable, therefore both kinds of diagrams are used, depending on what the partners found more clearly communicating the idea (see Figure 8).

3 Developer Tier: Design-time

The developer tier groups functionality used to design zApps and in one case help with designing the security of the platform. These Developer tier components are run outside the platform with help from software developers and domain specialists.

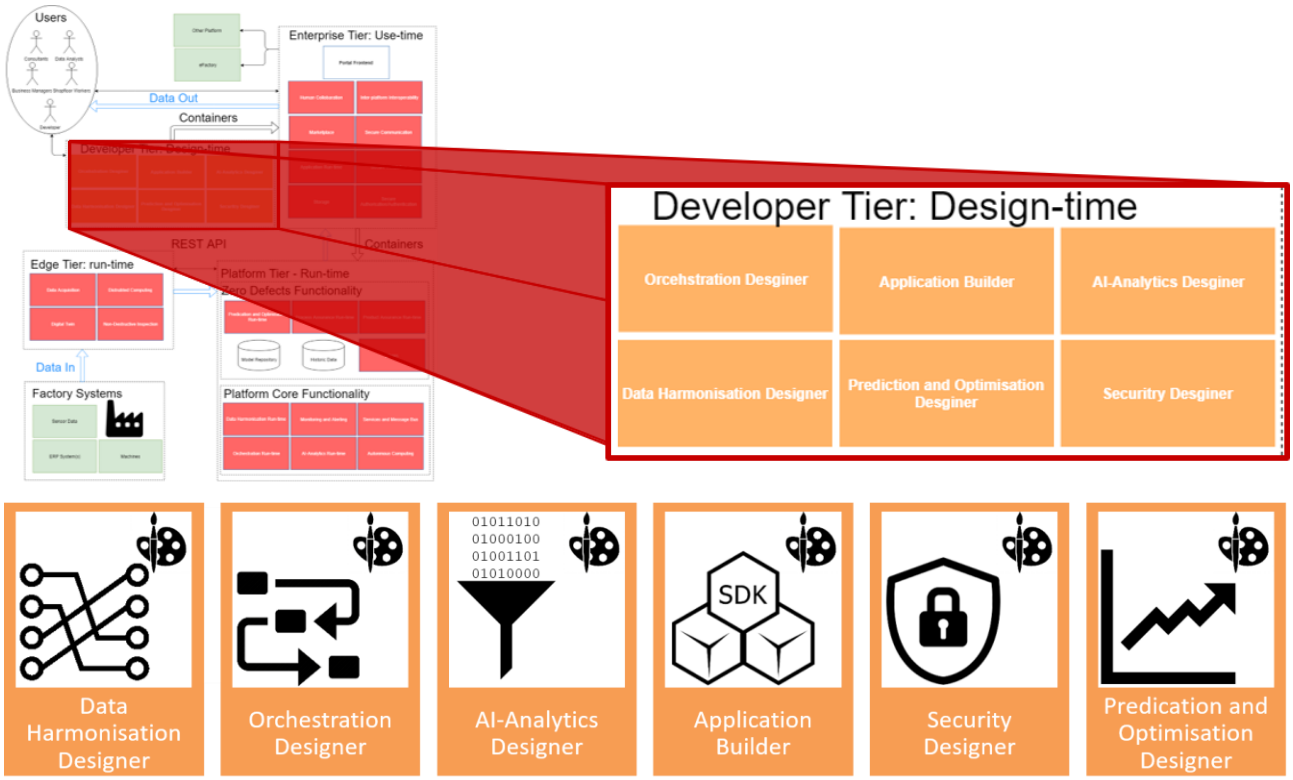
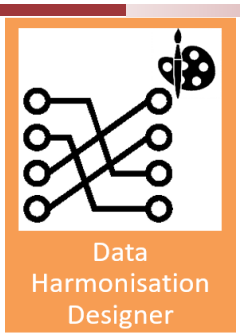


Figure 9: Developer Tier Components

3.1 Data Harmonisation Designer (T5.3)



3.1.1 Overall functional characterisation & Context

This module facilitates the transformation of different forms of data that are taken as inputs and harmonises the data into formats that can be used by other ZDMP components. This is achieved through the Data Harmonisation Designer, which is where the data is restructured from existing software systems into something that meets the needs of the ZDMP application, ie transforming data from its source format to its destination format. The formatting of the data at run-time is managed by the Data Harmonisation Run-Time, where the data from various sources into the ZDMP required format (see Section 5.1).

This designer models the definition of Manufacturing Maps; ie maps that allow the transformation and integration of data. It also provides basic functionalities for semantic homogenisation in a context of heterogeneous data. The developed applications enable a business analyst driven approach for the automatic linking of organisations' data schemas to the reference data model or the target data model needed by another component or zApp. ZDMP, the Data Harmonisation Designer and the AI-Analytics Designer will use publicly available manufacturing ontologies as reference data model (from the CREMA project) so an evolutionary data model can be supported in the form of crowdsourcing techniques. The Manufacturing Maps will be available in the ZDMP T6.2 Marketplace and Storage and exported as Transformation Services that are of a valuable use for the T5.4 Process Orchestration component.

The Data Harmonisation Designer component provides a set of functionalities that can be grouped on the following features:

- **Map Designer**: Allows the Manufacturing Maps can be generated. A Manufacturing Map file describes the rules to be executed to transform a specific syntax format A into format B which could then, for example, be used as part of a process. It will offer the user the possibility to annotate these maps with additional semantic metadata
- **Ontology Management:** Where ontological (concepts in OWL2, RDFS) and linked (RDF) datasets can be managed. It provides functionality for generic data (CRUD) management of the content stored in its semantic backend in the T6.2 Marketplace
- **Data Enrichment and Extraction:** Where the AI-Analytics can derive attributes or features from the data. This includes using statistical properties, data models and temporal data characteristics to discover internal relationships within the data

Their function can be grouped to the following features:

Subtask	Subtask description
T53A001 Connect to Database	**Priority:** Must
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time **What:** Open filesystem and access the schema of the data source (both for source and target schemas) **Why:** So that the schema can be loaded and, thus, the mappings can be performed
Acceptance Criteria	The schemas (source and/or target) are successfully retrieved
Requirements Filled	RQ_0007, RQ_0008, RQ_0116, RQ_0167, RQ_0172, RQ_0185, RQ_0662
T53A002 Read XML	**Priority:** Must
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design Time **What:** interprets eg XML schema files **Why:** So that the schema can be loaded and, thus, the mappings can be performed
Acceptance Criteria	The eg XML schema (source and/or target) is successfully interpreted and retrieved
Requirements Filled	RQ_0044, RQ_0093, RQ_0094, RQ_0389, RQ_0447, RQ_489, RQ_0551, RQ_0610, RQ_0662, RQ_0686, RQ_0722, RQ_0782
T53A003 Read CSV	**Priority:** Must
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time **What:** Interprets CSV schema files **Why:** So that the Transformation engine can read the transformation steps determined in the Map
Acceptance Criteria	The CSV schema (source and/or target) is successfully interpreted and retrieved
Requirements Filled	RQ_0044, RQ_0093, RQ_0094, RQ_0389, RQ_0447, RQ_489, RQ_0551, RQ_0610, RQ_0662, RQ_0686, RQ_0722, RQ_0782
T53A004 Read JSON	**Priority:** Must
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time **What: I**nterprets JSON schema files **Why:** So that the Transformation engine can read the transformation steps determined in the Map
Acceptance Criteria	The JSON schema (source and/or target) is successfully interpreted and retrieved
Requirements Filled	RQ_009, RQ_0010, RQ_0011, RQ0012, RQ_0040, RQ_0044, RQ_0093, RQ_0094, RQ_0159, RQ_0389, RQ_0447, RQ_489, RQ_0551, RQ_0610, RQ_0662, RQ_0686, RQ_0722, RQ_0782
T53A005 Read TXT	**Priority:** Must
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time **What:** Interprets plain text schema files **Why:** So that the Transformation engine can read the transformation steps determined in the Map
Acceptance Criteria	The plain txt schema (source and/or target) is successfully interpreted and retrieved
Requirements Filled	RQ_0044, RQ_0093, RQ_0094, RQ_0389, RQ_0447, RQ_489, RQ_0551, RQ_0610, RQ_0662, RQ_0686, RQ_0722, RQ_0782
T53A006 Read XLS	**Priority:** Must
	Who: Data Harmonisation Designer **Where:** Anywhere

	When: Design time **What:** Interprets XLS schema files **Why:** So that the Transformation engine can read the transformation steps determined in the Map
Acceptance Criteria	The XLS schema (source and/or target) is successfully interpreted and retrieved
Requirements Filled	RQ_0024, RQ_0044, RQ_0093, RQ_0094, RQ_0389, RQ_0447, RQ_489, RQ_0551, RQ_0610, RQ_0662, RQ_0686, RQ_0722, RQ_0782, RQ_0912
T53A007 **Read MySQL**	**Priority:** Must
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time **What:** Interprets MySQL schema files **Why:** So that the Transformation engine can read the transformation steps determined in the Map
Acceptance Criteria	The MySQL schema (source and/or target) is successfully interpreted and retrieved
Requirements Filled	*RQ_0023, RQ_0027, RQ_0044, RQ_0093, RQ_0094, RQ_0165, RQ_0389, RQ_0447, RQ_489, RQ_0551, RQ_0610, RQ_0662, RQ_0686, RQ_0722, RQ_0782*
T53A008 **Display UI**	**Priority:** Must
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time **What:** Show Mapping UI **Why:** So that the source schema can be displayed and, thus, the mappings can be performed
Acceptance Criteria	The Mapping UI is successfully shown
Requirements Filled	*RQ_0342, RQ_0373*
T53A009 **Load Source Schema**	**Priority:** Must
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time **What**: Loads source schema **Why:** So that the source schema can be accessed
Acceptance Criteria	The source schema is successfully loaded
Requirements Filled	*RQ_0015, RQ_0022, RQ_0030, RQ_0116, RQ_0167, RQ_0172, RQ_0185, RQ_0230, RQ_0709, RQ_0718*
T53A010 **Display Source Schema**	**Priority:** Must
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time (after T53A009) **What:** Display source schema **Why:** So that the source schema can be viewed
Acceptance Criteria	The source schema is successfully displayed
Requirements Filled	*RQ_0263, RQ_0658*
T53A011 **Analyse Source Schema**	**Priority:** Must
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time (After T53A010) **What:** Analyses source schema **Why:** So that the source schema can be manipulated **When/Where**: Design-time in the Data Harmonisation Designer
Acceptance Criteria	The source schema is successfully analysed

Requirements Filled	*RQ_0271*
T53A012 Connect to ontology	**Priority:** Must
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time **What:** Connects to domain (or ZDMP) ontology **Why:** So that alternative concepts can be suggested for the concepts present in the source schema
Acceptance Criteria	The ontology is successfully connected
Requirements Filled	*RQ_0274*
T53A013 Suggest Semantic Concepts for Source Schema	**Priority:** Must
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time **What:** Suggests alternative (or linked) concepts to the concepts present in the source schema **Why:** So that a crowdsourced domain (or ZDMP) ontology can be populated
Acceptance Criteria	Relevant concepts are suggested for a given concept
Requirements Filled	RQ_0881, RQ_0921, RQ_0923
T53A014 Display UI	**Priority:** Must
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time **What:** Show Mapping UI **Why:** So that the target schema can be displayed and, thus, the mappings can be performed
Acceptance Criteria	The Mapping UI is successfully shown
Requirements Filled	None specified
T53A015 Load Target Schema	**Priority:** Must
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time **What:** Loads target schema **Why:** So that the target schema can be accessed
Acceptance Criteria	The target schema is successfully loaded
Requirements Filled	*RQ_0167, RQ_0172, RQ_0185, RQ_0328*
T53A016 Display Target Schema	**Priority:** Must
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time (after T53A015) **What:** Display target schema **Why:** So that the target schema can be viewed
Acceptance Criteria	The target schema is successfully displayed
Requirements Filled	*RQ_0167, RQ_0172, RQ_0185, RQ_0328*
T53A017 Analyse Target Schema	**Priority:** Must
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere

	When: Design time (after T53A016) **What:** Analyses target schema **Why:** So that the target schema can be manipulated
Acceptance Criteria	The target schema is successfully analysed
Requirements Filled	*RQ_0328*
T53A018 **Connect to Ontology**	**Priority:** Must
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time **What:** Connects to domain (or ZDMP) ontology **Why:** So that alternative concepts can be suggested for the concepts present in the target schema
Acceptance Criteria	The ontology is successfully connected
Requirements Filled	*RQ_0328*
T53A019 **Suggest Semantic Concepts for Target Schema**	**Priority:** Must
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time **What:** Suggests alternative (or linked) concepts to the concepts present in the target schema **Why:** So that a crowdsourced domain (or ZDMP) ontology can be populated
Acceptance Criteria	Relevant concepts are suggested for a given concept
Requirements Filled	N/A
T53A020 **Connect to the Storage**	**Priority:** Must
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time **What:** Connect to the Storage with the credentials as directed by the T5.2 Secure Authentication/Authorisation component **Why:** So that the Maps can be read, searched, and filtered after save
Acceptance Criteria	The T6.2 Storage is accessible
Requirements Filled	*RQ_0032, RQ_0085, RQ_0187, RQ_0637*
T53A021 **Read file**	**Priority:** Must
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time **What:** Read file **Why:** So that the Maps can be loaded into the Data Harmonisation Designer component
Acceptance Criteria	The file is successfully read
Requirements Filled	*RQ_0187, RQ_0832, RQ_0835*
T53A022 **Search Map**	**Priority:** Must
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time **What:** Search map **Why:** So that the Maps can be searched into the Storage component
Acceptance Criteria	The list of maps resulted after the search matches the searching criteria specified
Requirements Filled	*RQ_0217*
T53A023	**Priority:** Must

Filtering in the Storage	**Description**
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time **What:** Apply filters **Why:** So that the Maps stored in the Data Storage can be filtered according to user specified criteria
Acceptance Criteria	The list of maps is filtered out with the specified criteria
Requirements Filled	RQ_0335, RQ_0712, RQ_0713, RQ_0714, RQ_0771, RQ_0772, RQ_0773, RQ_0774, RQ_0832, RQ_0835
T53A024 **Preview Map**	**Priority:** Should
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time **What:** Preview map **Why:** So that the Maps can be graphically previewed before loading into the Data Harmonisation Designer component
Acceptance Criteria	The map is graphically viewed by the user
Requirements Filled	None specified
T53A025 **Annotate Map**	**Priority:** Must
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time **What:** Annotate map with metadata **Why:** So that the Maps can be searched and filtered with this metadata as parameters
Acceptance Criteria	The file is successfully annotated, and the metadata is stored along with the map file
Requirements Filled	RQ_0032
T53A026 **Publish Map**	**Priority:** Must
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time **What:** The map is published to a computer readable format **Why:** So that the Maps can be wrapped and become an executable service
Acceptance Criteria	The file is successfully published and accessible
Requirements Filled	RQ_0032, RQ_0085, RQ_0637
T53A027 **Persist Map**	**Priority:** Must
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time **What:** Save the map in the Storage **Why:** So that the Maps can be re-used, retrieved, removed, searched, and filtered
Acceptance Criteria	The file is successfully persisted in the Storage
Requirements Filled	RQ_0032, RQ_0085, RQ_0637
T53A028 **Search Map**	**Priority:** Must
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere

	When: Design time **What:** Search map **Why:** So that the Maps can be searched into the Storage component
Acceptance Criteria	The list of maps resulted after the search matches the searching criteria specified
Requirements Filled	RQ_0655, RQ_0656, RQ_0657, RQ_0658
T53A029 **Delete Map**	**Priority:** Must
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time **What:** Remove map **Why:** Checks that the persisted Maps in the Storage can be removed from it
Acceptance Criteria	The selected map(s) is successfully removed from the T6.2 Storage
Requirements Filled	RQ_0032, RQ_0085, RQ_383, RQ_0637
T53A030 **Annotate Service**	**Priority:** Must
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time **What:** Annotate map for publication **Why:** So that the to-be deployed Map can be easily found in the Storage
Acceptance Criteria	The active map is successfully annotated
Requirements Filled	None specified.
T53A031 **Create Service**	**Priority:** Must
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time **What:** Create self-executing service **Why:** So that the deployed Map can be executed as a stand-alone service
Acceptance Criteria	The transformation service is successfully created from the active map, together with its annotations
Requirements Filled	None specified.
T53A032 **Deploy Service**	**Priority:** Must
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time **What:** Create Docker package **Why:** So that the deployed Map can be executed and scalable if necessary, as a stand-alone service
Acceptance Criteria	The active map, together with its annotations, is successfully packaged as Docker container
Requirements Filled	None specified
T53A033 **Connect to the Storage**	**Priority:** Must
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time **What:** Connect to Marketplace with the credentials as directed by the T5.2 Secure Authentication/Authorisation component **Why:** So that the Transformation Services (ie the deployed maps) can be published
Acceptance Criteria	The Marketplace is accessible
Requirements Filled	RQ_0032, RQ_0085, RQ_0637

T53A034 **Publish Deployed Map**	**Priority:** Must
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time (After T53A032) **What:** Publish deployed map (ie transformation service) **Why:** So that the Transformation Service can be sold to ZDMP Users and re-used within eg the Process Orchestration
Acceptance Criteria	The service is successfully published in the Marketplace
Requirements Filled	RQ_0032, RQ_0085, RQ_0637
T53A035 **Get Related Data**	**Priority:** Must
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time **What:** Obtain data with internal relationships to a given concept passed as parameter **Why:** So that the Data Harmonisation Designer can make use of the "wisdom of the crowd" for developing the maps
Acceptance Criteria	A set of relevant data with internal relationships is made available
Requirements Filled	RQ_0880, RQ_0881, RQ_0885, RQ_0921, RQ_0923, RQ_0966 RQ_0984, RQ_0985, RQ_1011, RQ_1037, RQ_1037, RQ_1040, RQ_1054, RQ_1055
T53A036 **Get Routine**	**Priority:** Must
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time **What:** Obtain the internal relationships between a set of given concepts passed as parameters **Why:** So that the Data Harmonisation Designer can make use of the "wisdom of the crowd" for developing the maps
Acceptance Criteria	A set of internal relationships is made available
Requirements Filled	RQ_0880, RQ_0881, RQ_0885, RQ_0921, RQ_0923, RQ_0966 RQ_0984, RQ_0985, RQ_1011, RQ_1037, RQ_1037, RQ_1040, RQ_1054, RQ_1055
T53A037 **Add Related Data**	**Priority:** Must
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time **What:** Add a new concept that is linked to a given concept passed as parameter **Why:** So that the Data Harmonisation Designer can provide feedback to the "wisdom of the crowd" for developing future maps
Acceptance Criteria	The ontology is updated with the provided content
Requirements Filled	RQ_0032, RQ_0085, RQ_0637, RQ_0880, RQ_0881, RQ_0885, RQ_0921, RQ_0923
T53A038 **Add Routine**	**Priority:** Must
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time **What:** Add a new link between two given concepts, passed as parameters **Why:** So that the Data Harmonisation Designer can provide feedback to the "wisdom of the crowd" for developing future maps
Acceptance Criteria	The ontology is updated with the provided content
Requirements Filled	RQ_0032, RQ_0085, RQ_0637, RQ_0880, RQ_0881, RQ_0885, RQ_0921, RQ_0923
T53A039 **Update Related Data**	**Priority:** Must
	Description

	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time (after T53A035) **What:** Update a concept that is linked to a given concept passed as parameter **Why:** So that the Data Harmonisation Designer can provide feedback to the "wisdom of the crowd" for developing future maps
Acceptance Criteria	The ontology is updated with the provided content
Requirements Filled	RQ_0032, RQ_0085, RQ_0637, RQ_0880, RQ_0881, RQ_0885, RQ_0921, RQ_0923
T53A040 **Update Routine**	**Priority:** Must
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time (after T53A036) **What:** Update a given link between two concepts, passed as parameters **Why:** So that the Data Harmonisation Designer can provide feedback to the "wisdom of the crowd" for developing future maps
Acceptance Criteria	The ontology is updated with the provided content
Requirements Filled	RQ_0032, RQ_0085, RQ_0637, RQ_0880, RQ_0881, RQ_0885, RQ_0921, RQ_0923
T53A041 **Get Concept**	**Priority:** Must
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time **What:** Get concepts from the ontology **Why:** So that the Data Harmonisation Designer can make use of the ZDMP Ontology for developing the maps
Acceptance Criteria	The ZDMP Ontology provides with the requested concepts
Requirements Filled	RQ_0966
T53A042 **Add Concept**	**Priority:** Must
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time **What:** Add concepts to the ontology **Why:** So that the Data Harmonisation Designer can make use of the ZDMP Ontology for developing the maps
Acceptance Criteria	The ZDMP Ontology is updated with new concepts
Requirements Filled	RQ_0032, RQ_0085, RQ_0637
T53A043 **Update Concept**	**Priority:** Must
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time (after T53A041) **What:** Update concepts in the ontology **Why:** So that the Data Harmonisation Designer can make use of the ZDMP Ontology for developing the maps
Acceptance Criteria	The ZDMP Ontology is updated with new metadata about existing concepts
Requirements Filled	RQ_0032, RQ_0085, RQ_0637
T53A044 **Delete Concept**	**Priority:** Must
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time (after T53A041) **What:** Remove concepts from the ontology **Why:** So that the Data Harmonisation Designer can update the ZDMP Ontology for future developing the maps

	When/Where: Design-time in the Data Harmonisation Designer
Acceptance Criteria	The ZDMP Ontology is updated with the removal of existing concepts
Requirements Filled	RQ_0032, RQ_0085, RQ_0637
T53A045 Reasoning	**Priority:** Must
	Description
	Who: Data Harmonisation Designer **Where:** Anywhere **When:** Design time **What:** Reason **Why:** So that the Data Harmonisation Designer can make use of the ZDMP Ontology for developing the maps
Acceptance Criteria	The ZDMP Ontology provides with a set of reasoned objects as per the parameters received
Requirements Filled	RQ_0047, RQ_0054, RQ_0098, RQ_0120, RQ_0140

Figure 10: Data Harmonization Designer Functions

3.1.2 Workflows

The following sub-sections describe the sequence diagrams of the Data Harmonisation Designer component.

3.1.2.1 Read Data Sources

This feature provides the capability to read a set of types of data sources that will be used when performing the mapping task. Figure 11 shows the sequence diagram of this feature.

The main steps/functionalities are as follows:

- Connect to Data Source
- Read XML
- Read CSV
- Read JSON
- Read TXT
- Read XLS
- Read MySQL Database

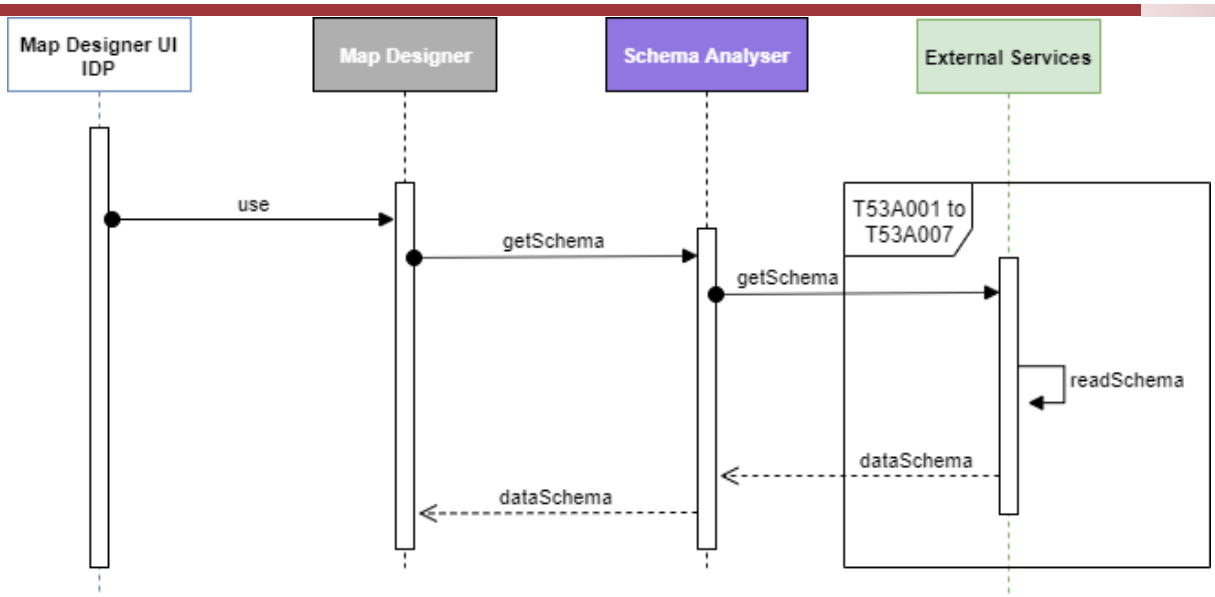


Figure 11: Read data sources sequence diagram

3.1.2.2 Read Source Schema and Read Target Schema

This feature provides the capability to read the source and the target schemas that will be used when performing the mapping task. Figure 12 shows how the Data Harmonisation component will read the source data, and Figure 13 shows how it will read the target data.

The main steps/functionalities are as follows:

- Display UI
- Load Source/Target Schema
- Display Source/Target Schema
- Analyse Source/Target Schema
- Connect to Ontology
- Suggest Semantic Concepts for Source/Target Schema

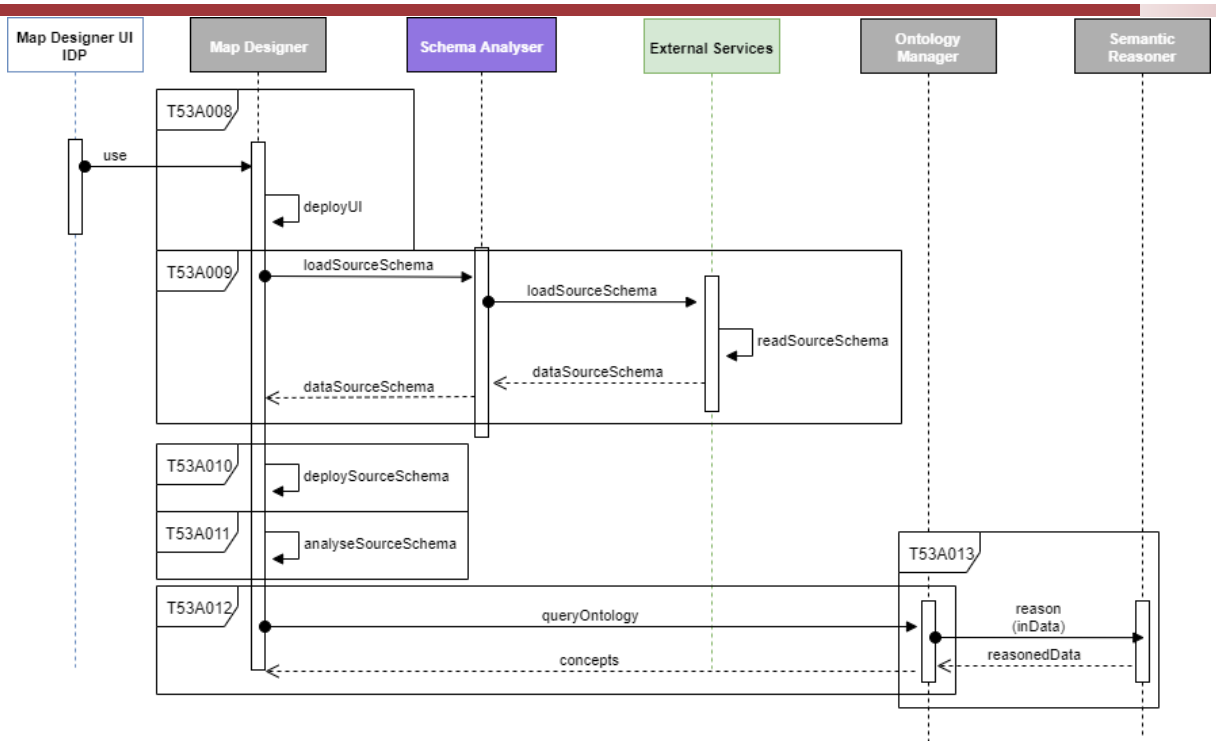


Figure 12: Read Source Schema Sequence Diagram

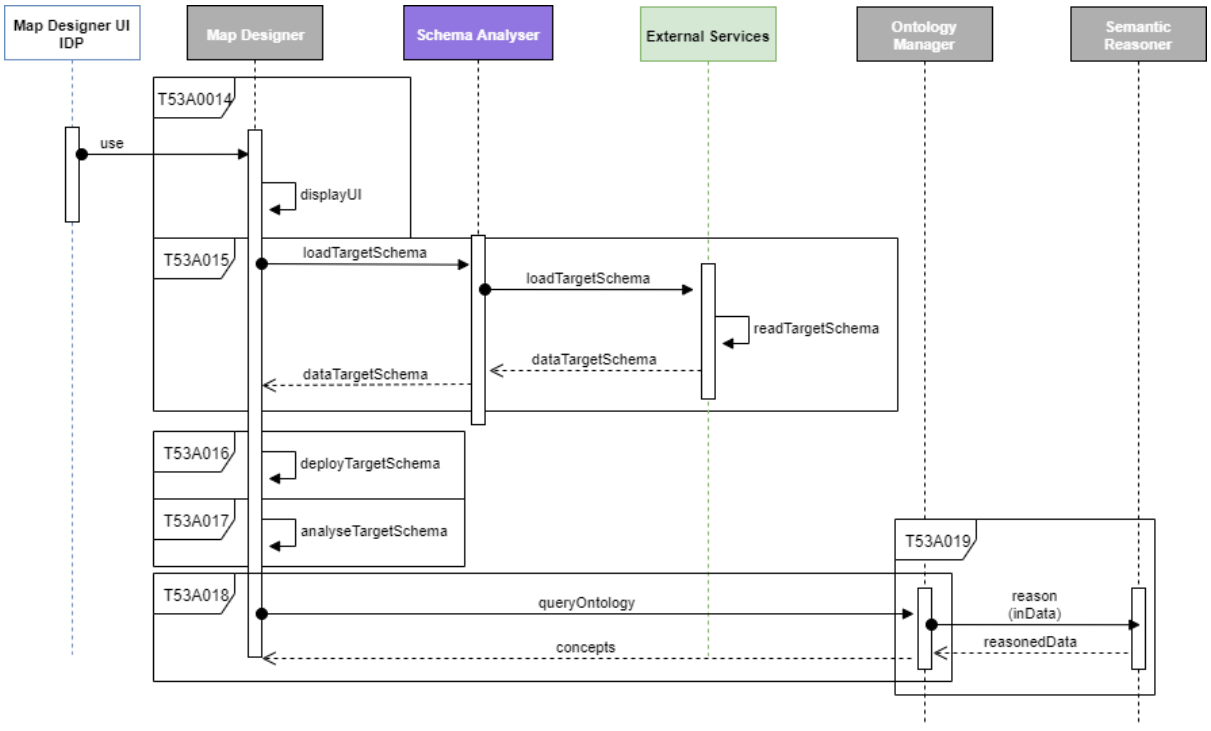


Figure 13: Read Target Schema Sequence Diagram

3.1.2.3 Manage Maps

This feature provides the capability to read, retrieve, store, and delete a Manufacturing Map from the Marketplace. Each of these capabilities have an associated sequence diagram, Figure 14 covers the reading of a map, Figure 15 covers the retrieving of a map, Figure 16 covers storing of a map, and Figure 17 covers deleting a map.

The main steps/functionalities and their subtask ID are as follows:

- Read Map
  - Connect to Storage (see function T53A020)
  - Read File (see function T53A021)
- Retrieve Map
  - Connect to Storage (see function T53A020)
  - Search Map (see function T53A022)
  - Filtering in Storage (see function T53A023)
  - Preview Map (see function T53A024)
- Store Map
  - Connect to Storage (see function T53A020)
  - Annotate Map (see function T53A025)
  - Serialise Map (see function T53A026)
  - Persist Map (see function T53A027)
- Delete Map
  - Connect to Storage (see function T53A020)
  - Search Map (see function T53A028)
  - Delete Map (see function T53A029)

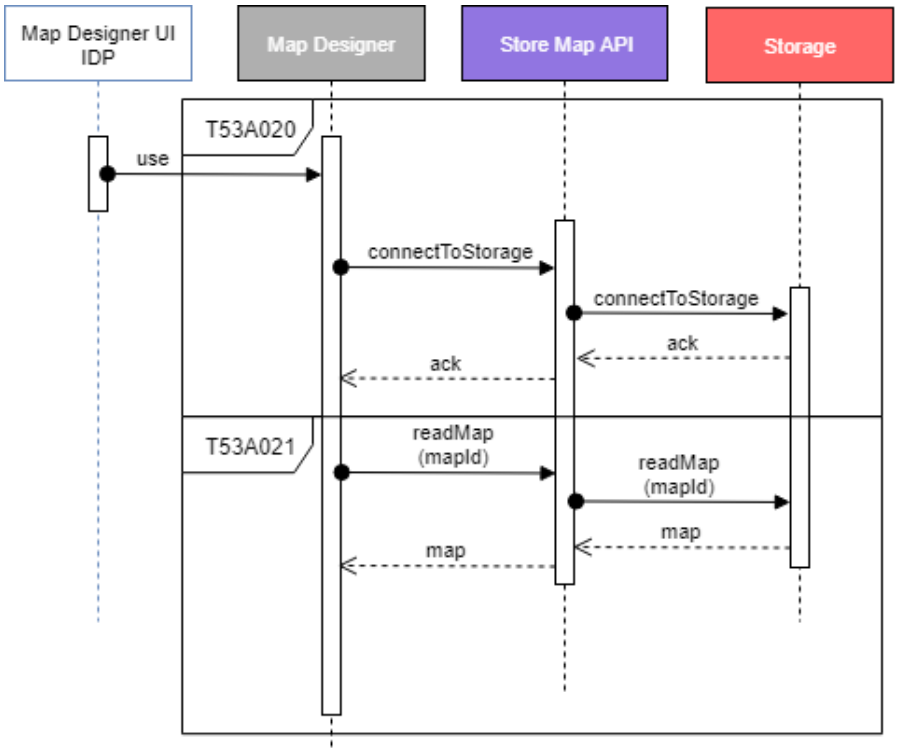


Figure 14: Read Map Sequence Diagram

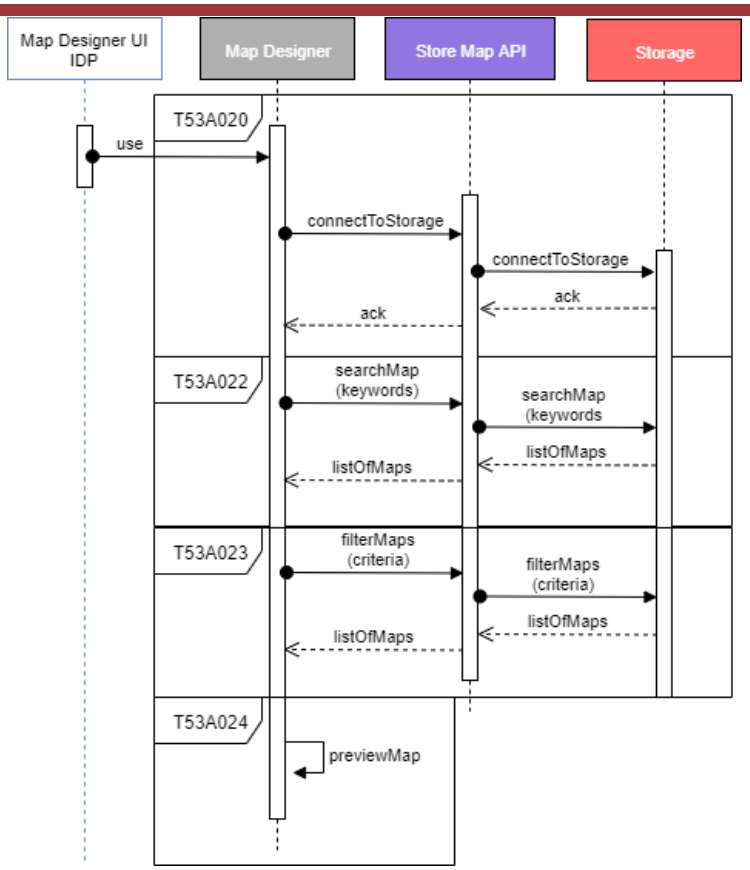


Figure 15: Retrieve Map Sequence Diagram

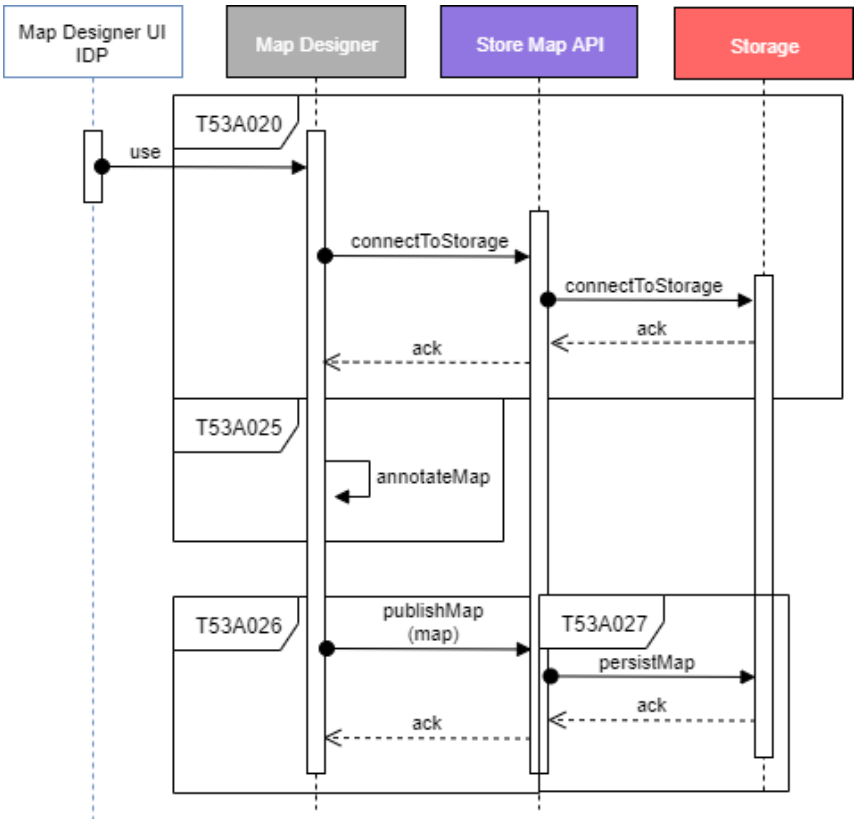


Figure 16: Store Map Sequence Diagram

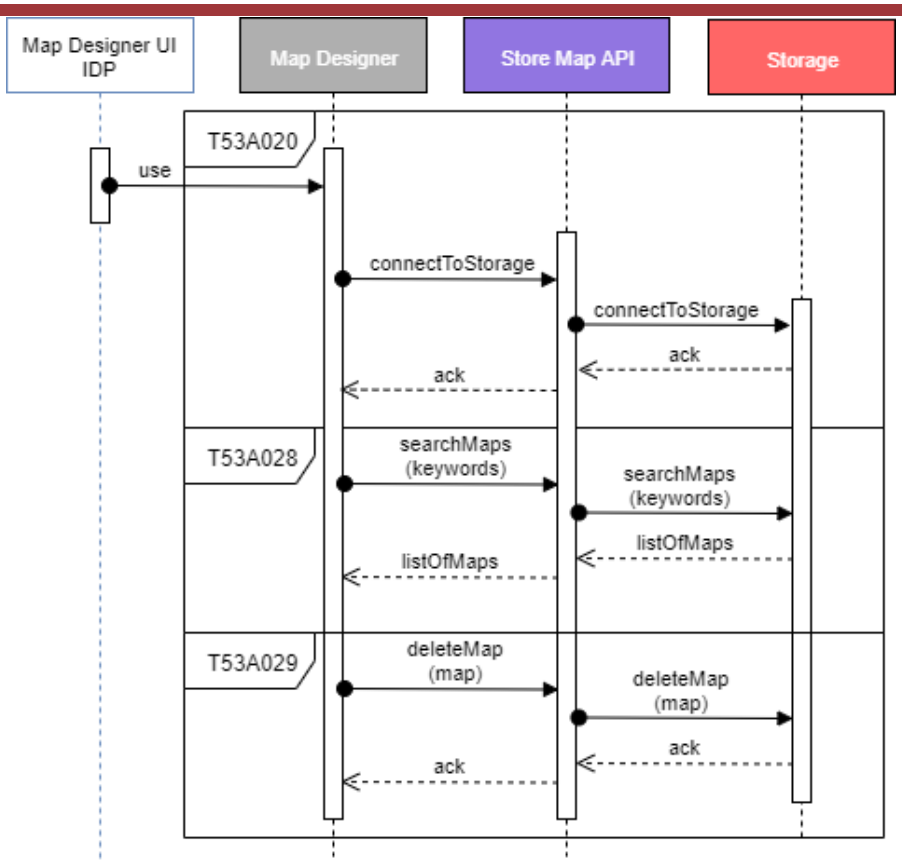


Figure 17: Delete Map Sequence Diagram

3.1.2.4 Deploy and Publish a Map

These features provide the capability to deploy and publish a map after it has been generated by the Business Analyst. The main steps/functionalities are as follows:

- Deploy Map
  - Annotate Service (see function T35A030)
  - Create Service (see function T35A031)
  - Deploy Service (see function T35A032)
- Publish Map
  - Connect to T6.2 ZDMP Marketplace (see function T35A033)
  - Publish Deployed Map (see function T35A034)

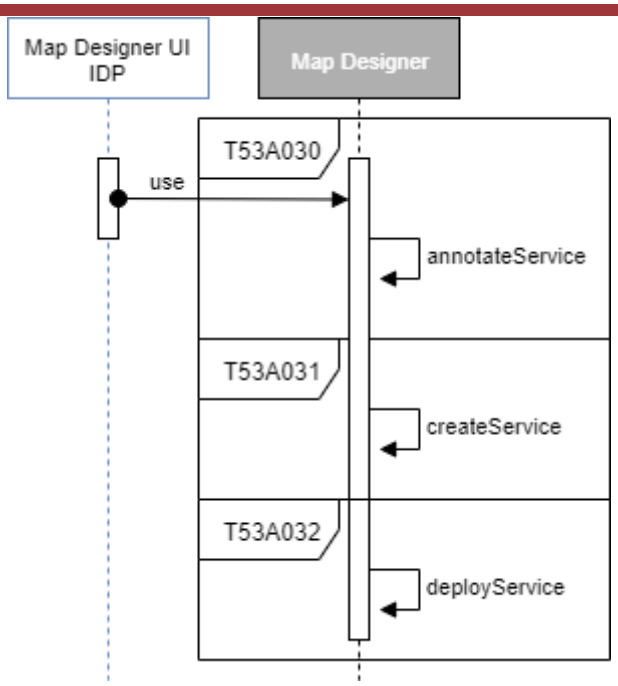


Figure 18: Deploy Map Sequence Diagram

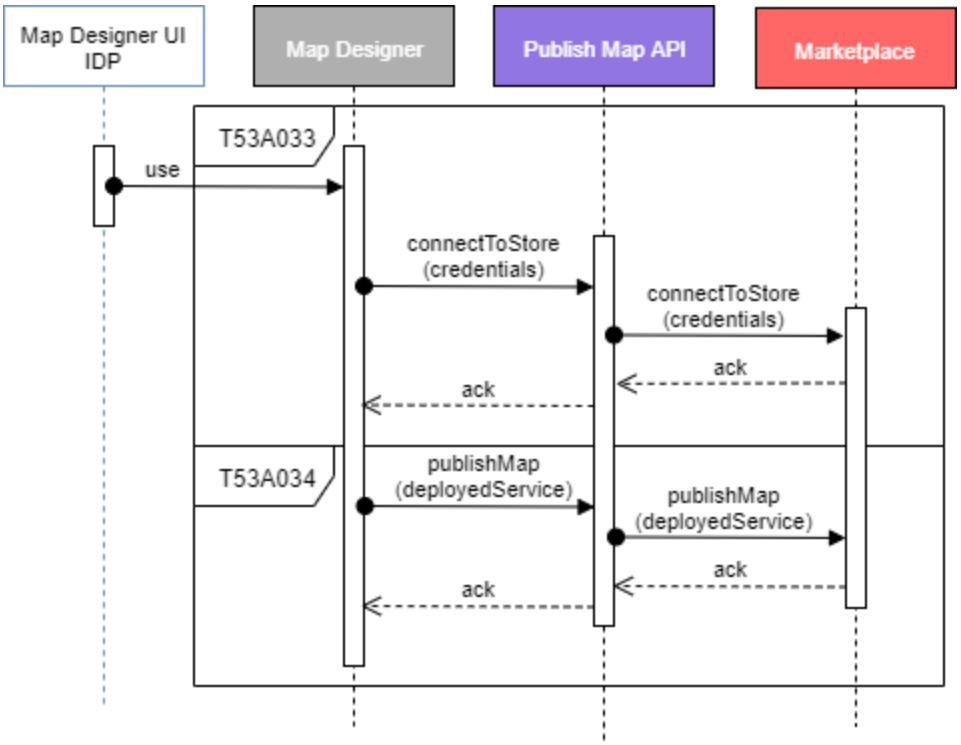


Figure 19: Publish Map Sequence Diagram

3.1.2.5 Data Enrichment & Extraction

This feature provides the capability to access the machine learning routines functionality that the Data Orchestration Designer is offering to the Business Analyst when generating their Manufacturing Maps. There are several sequence diagrams associated with this feature. Figure 20 shows getting related data or routines, Figure 21 shows adding related

data, Figure 22 shows adding new routines, Figure 23 shows updating related data, and Figure 24 shows updating routines.

The main steps / functionalities are as follows:

- Get Related Data (see function T35A035)
- Get Routine (see function T35A036)
- Add Related Data (see function T35A037)
- Add Routine (see function T35A038)
- Update Related Data (see function T35A039)
- Update Routine (see function T35A040)

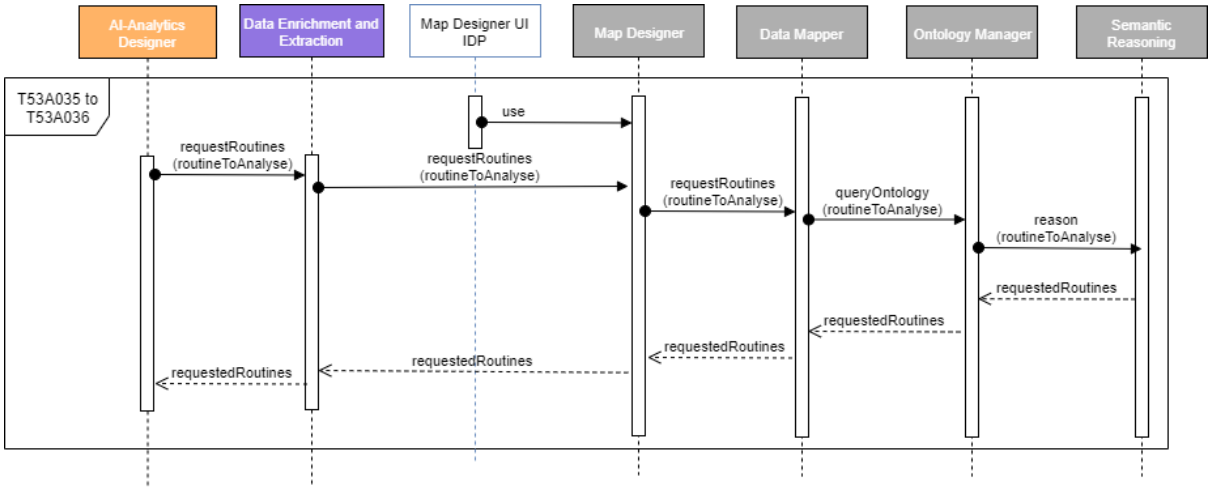


Figure 20: Get the Routines Sequence Diagram

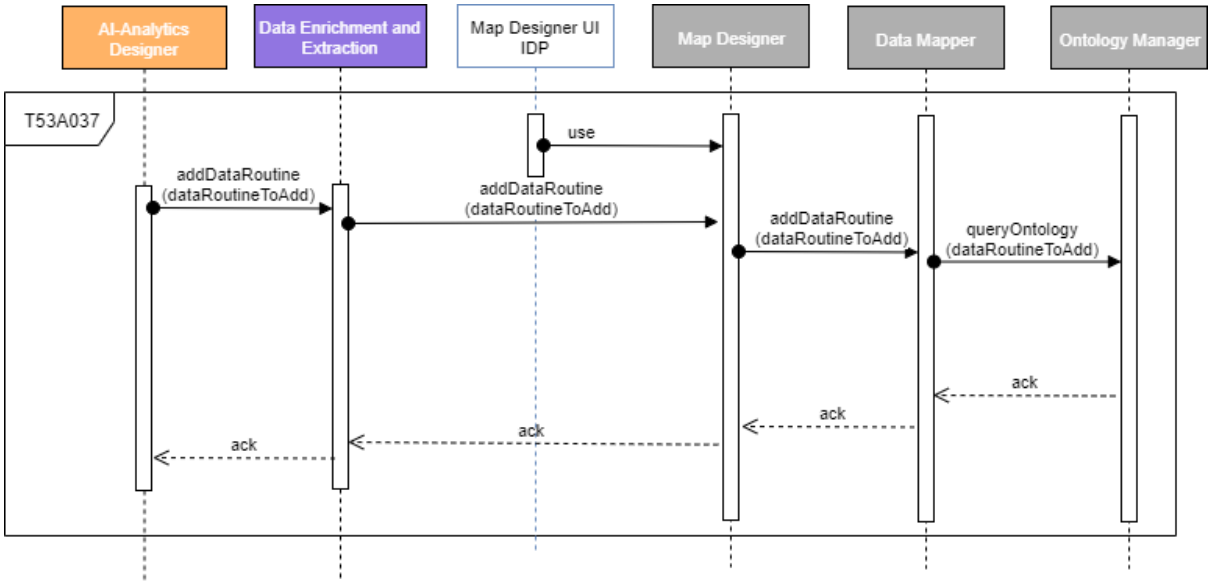


Figure 21: Add Related Data Sequence Diagram

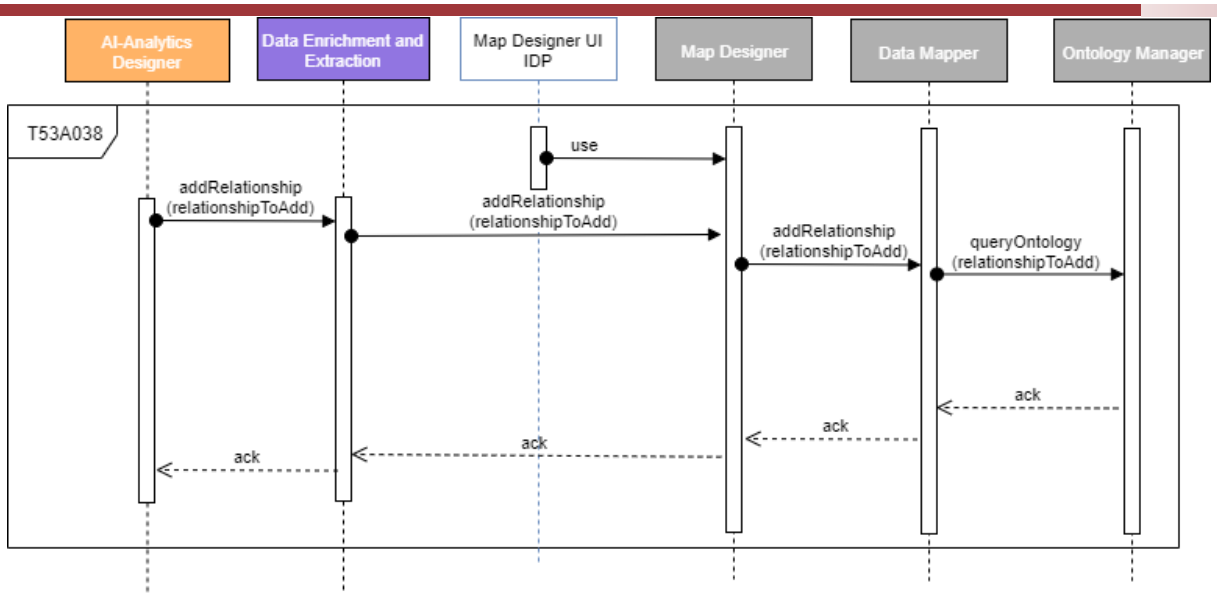


Figure 22: Add Routine Sequence Diagram

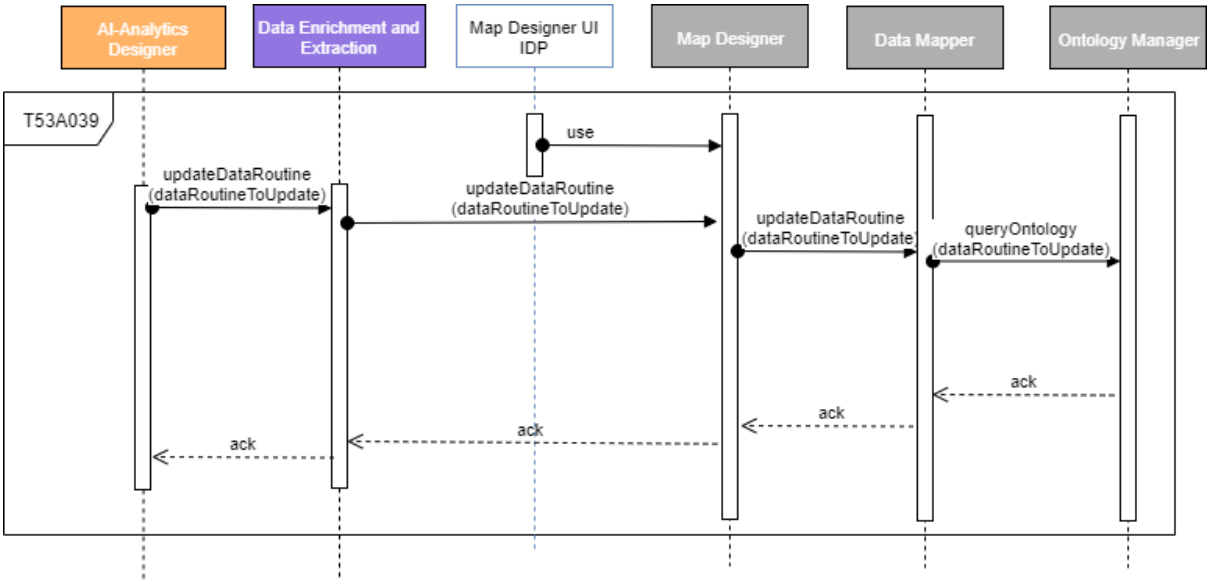


Figure 23: Update Related Data Sequence Diagram

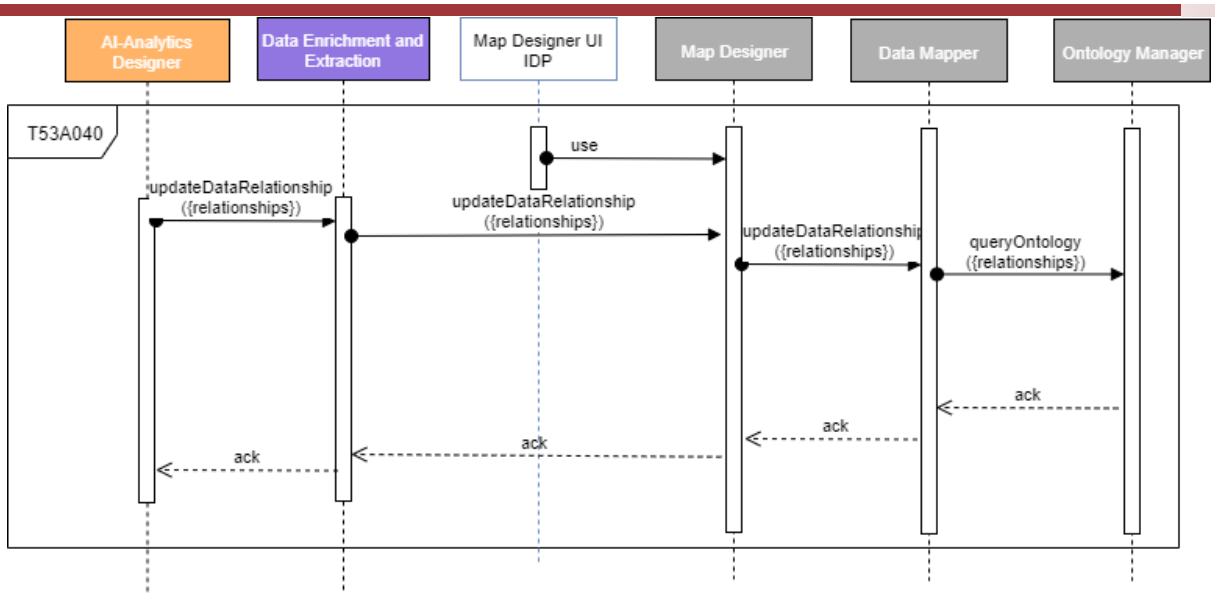


Figure 24: Update Routines Relationship Sequence Diagram

3.1.2.6 Manage Ontology

This feature provides the capability to access the domain ontology functionality that the Data Harmonisation Designer is offering to the Business Analyst when generating their Manufacturing Maps. This feature has five associated sequence diagrams. Figure 25 shows getting a concept, Figure 26 shows adding a concept, Figure 27 shows updating a concept, Figure 28 shows deleting a concept and Figure 29 shows the reasoning sequencing diagram.

The main steps/functionalities are as follows:

- Get Concept (see function T35A041)
- Add Concept (see function T35A042)
- Update Concept (see function T35A043)
- Delete Concept (see function T35A044)
- Reasoning (see function T35A045)

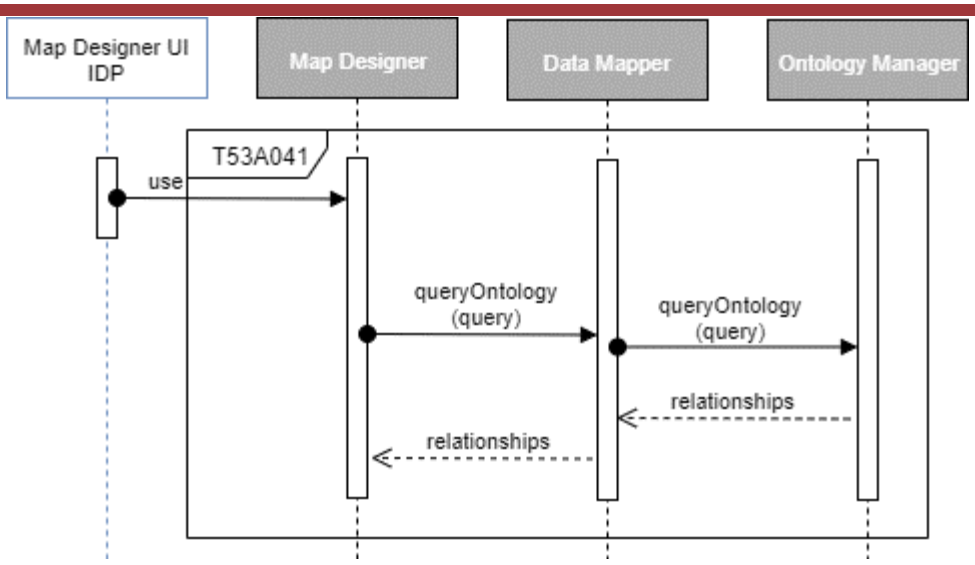


Figure 25: Get Concept Sequence Diagram

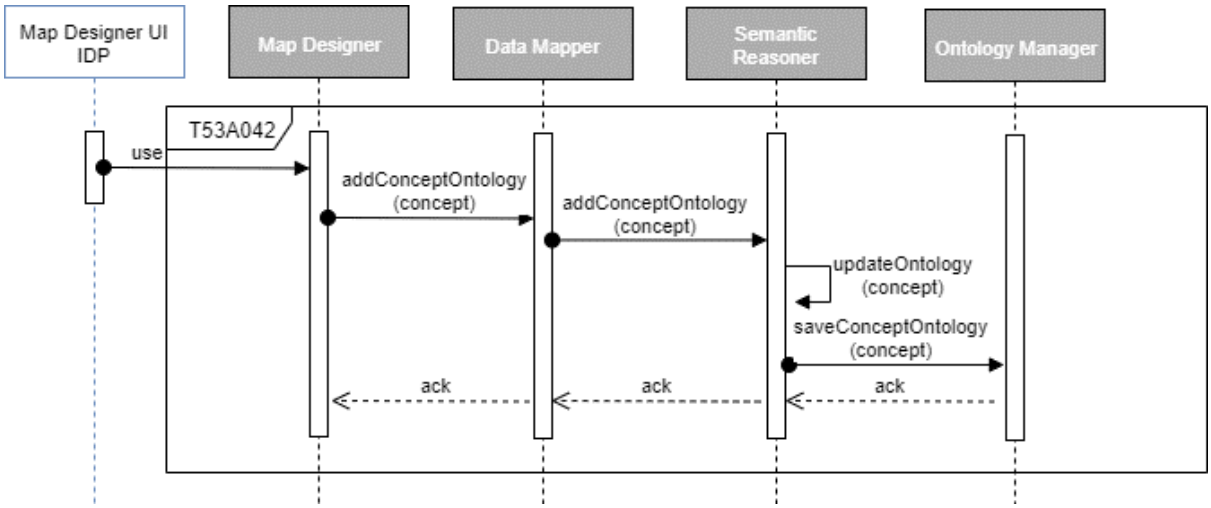


Figure 26: Add Concept Diagram

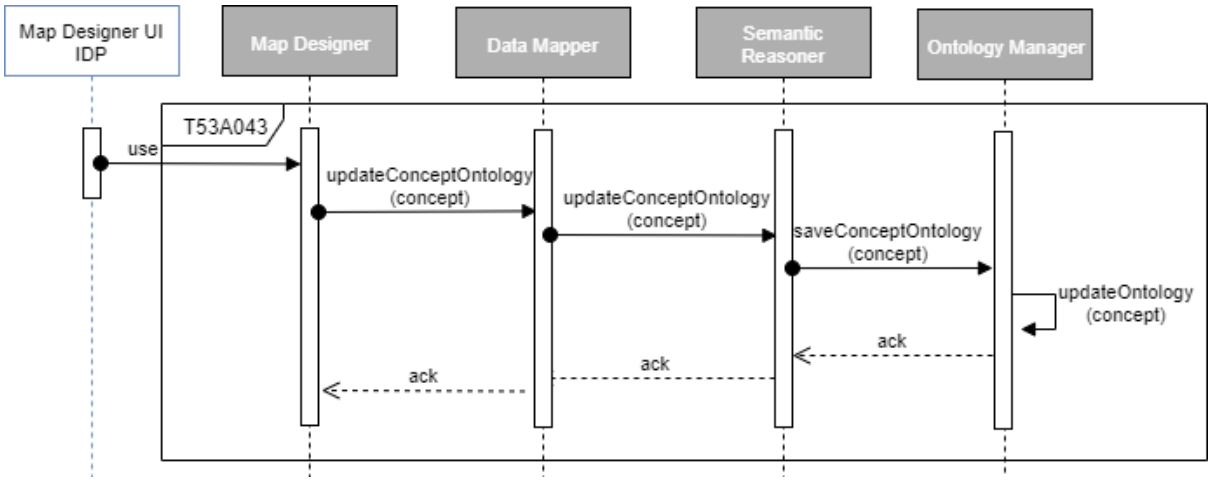


Figure 27: Update Concept Diagram

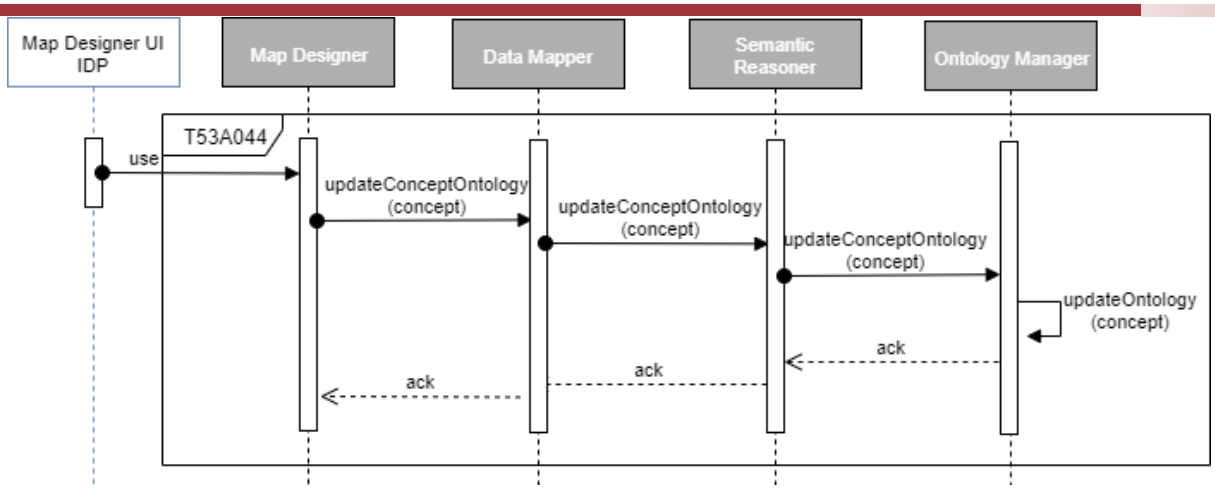


Figure 28: Delete Concept Diagram

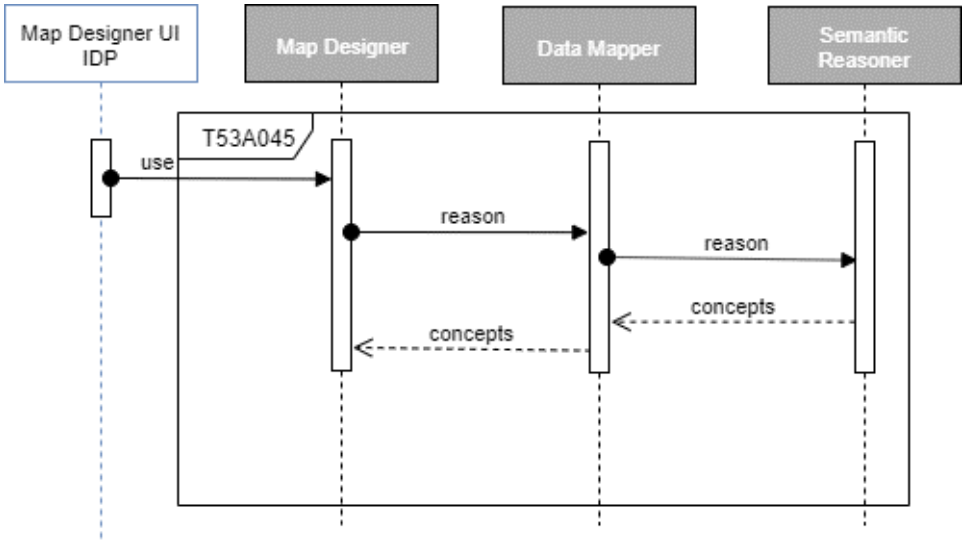


Figure 29: Reasoning Sequence Diagram

3.1.3 Additional Issues

Additional issues have appeared after the description of the architecture.

Issue	Description	Next Steps	Lead (Rationale)
Performance Issues	The following requirements with a “must“-priority were targeted at the task 5.3, but there are concerns about performance relating to the following requirements: *RQ_0036, RQ_0082, RQ_0090, RQ_0110, RQ_0118, RQ_0132, RQ_0136, RQ_0150*	Discuss with requirement providers who to solve the issue	T5.3 Data Harmonisation
Subtasks without specified fulfilled requirements	The following subtasks do not fulfil specific requirements, but are there to fulfil a more general purpose: *T53A014, T53A024, T53A030, T53A031, T53A032*	Ensure that all the tasks have requirements	T5.3 Data Harmonisation

Figure 30: Data Harmonization Designer Issues

3.2 Orchestration Designer (T5.4)

The interaction between the Orchestration Designer and other components of the ZDMP platform (Orchestration Runtime, Monitoring and Alerting, Process Engine, Marketplace, Storage, etc) will be facilitated using the T6.4 Service and Message Bus. The Monitoring and Alerting provides alerts based on a set of incoming data and rules.

3.2.1 Overall functional characterisation & Context

The Orchestration Designer is responsible for allowing users to model multiple manufacturing workflows for orchestrating the various assets available within a collaborative framework.

The tool will be a reactive, extensible, and an online workspace supporting the design or modifications of BPMN-like models and be callable by APIS for use in zApps where process design and orchestration is appropriate.

3.2.2 Functions / Features

The Orchestration Designer component provides the following set of functionalities:

- **BPMN 2.0 Modelling and Rendering Service**: This service provides the means of rendering and modelling a process in BPMN format. It connects to the Storage component for saving and retrieving a process model for designing and editing. At runtime, the process model is executed via the Process Engine
- **Toolbox elements:** The toolbox contains the list of all BPMN elements, filtered by elements attachable to the selected diagram element, if any. This includes events, gateways, and tasks
- **Process explorer:** Allows the user to create/open/browse all stored diagrams and shows a preview of each one
- **Properties panel:** The properties panel appears when users selects an existing element in the current diagram. Depending on the type of elements, a distinct set of properties appears, so user can set their values
- **Status panel:** This panel lists errors in the current diagram such as required values not set or missing information. The User can click on each item to get more information and automatically select the element which has the problem
- **Marketplace explorer:** Allows the user to use an existing asset/service from the marketplace. This is visually represented by a service task and can be added to a diagram as part of the flow. The metadata for each asset/service is used by the designer to ask for required attributes (mappings and other data required by the service call) by using the properties panel
- **Instance explorer:** Allows the user to interact with the orchestration engine, to deploy process models and get the list of running instances. It also allows the user all instance related management, like start/stop/suspend/resume instances

The functions can be grouped to the following features which have to be realised:

Subtask	Subtask description
T54A001 **Process Explorer**	**Priority:** Must
	Who: Developer **What:** Open Process Explorer to browse available processes **Why:** To provide already created processes and services to the orchestration designer

	When: During designing of a process **Where:** In the process design and the Developer Tier.
Acceptance Criteria	User received a list of the process models stored on ZDMP-Store
Requirements Filled	Cross App Functionality – see Additional Issues
T54A002 **Search/Filter process models**	**Priority:** Must
	Who: Designer **What:** Search/filter process models **Why:** So the user can find a process to be added to the orchestrator **When:** During design of a process **Where:** In the process design and the Developer Tier
Acceptance Criteria	User can search existing process models
Requirements Filled	Cross App Functionality – see Additional Issues
T54A003 **Create/Open existing process**	**Priority:** Must
	Who: Designer **What:** Create a new process/Open existing process **Why:** To access previously created processes or create new ones **When:** To start designing of a process **Where:** In the process design and the Developer Tier.
Acceptance Criteria	User can create a new process model and can open an existing one
Requirements Filled	Cross App Functionality – see Additional Issues
T54A004 **Save process model**	**Priority:** Must
	Who: Designer **What:** Save process model changes to local storage or platform storage **Why:** Automatically save changes made by the user to the current process model to limit the loss of work **When:** During process design **Where:** In process designer and the Developer Tier
Acceptance Criteria	Process model is automatically saved while user does changes. User can reload the designer and his changes are there
Requirements Filled	Cross App Functionality – see Additional Issues
T54A005 **Diagram – connect figures**	**Priority:** Must
	Who: Designer **What:** Drag and drop elements and connect them together **Why:** Allow the user to design the BPMN process model **When:** During process design **Where:** In process designer and the Developer Tier
Acceptance Criteria	User can drag and drop elements from the toolbox and connect them together to create the process model results in a valid BPMN model
Requirements Filled	Cross App Functionality – see Additional Issues
T54A006 **Update element common properties**	**Priority:** Must
	Who: Designer **What:** Update common element properties by using properties panel **Why:** To allow customisation of the connections to components or zApps **When:** During process design **Where:** In process designer and the Developer Tier
Acceptance Criteria	Attributes are stored as they are set by the user. Different attributes appear depending on which element in the diagram is selected. Common attributes such as name or position can be set using the properties panel
Requirements Filled	Cross App Functionality – see Additional Issues
T54A007 **Marketplace explorer**	**Priority:** Must
	Who: Designer **What:** Allow the user to add a service task from the marketplace to the current diagram **Why:** Use a service/asset from the marketplace as part of the process model being designed

	When: During process design **Where:** In process designer and the Developer Tier
Acceptance Criteria	User drags and drops the service task element and the marketplace explorer appears, where user can browse/select the service he wants to use
Requirements Filled	Cross App Functionality – see Additional Issues
T54A008 **Validate process**	**Priority:** Must
	Who: Designer **What:** Provides the list of errors/attributes that must be set by the user **Why:** Ensure the process model and all its elements have all required input **When:** During process design **Where:** In process designer and the Developer Tier
Acceptance Criteria	User gets the list of errors/missing required data. Clicking on each item in the list, selects the element in the diagram, highlighting the missing attributes or the wrong values.
Requirements Filled	Cross App Functionality – see Additional Issues
T54A009 **Make valid connections**	**Priority:** Must
	Who: Designer **What:** Allow only certain toolbox elements to be connected together **Why:** Following BPMN specification, certain elements cannot be connected, such as an end event and a message start. **When:** During process design **Where:** In process designer and the Developer Tier
Acceptance Criteria	User cannot create invalid connections in the process model
Requirements Fields	Cross App Functionality – see Additional Issues
T54A010 **Gateway condition designer**	**Priority:** Must
	Who: Designer **What:** Allows the user to create condition expressions that will be evaluated at runtime **Why:** Create conditions that will affect the flow in runtime **When:** During process design **Where:** In the process designer and the Developer Tier
Acceptance Criteria	User can create the expression that will affect the process flow in runtime. Expression can use mathematical functions or generic code functions (such as Substr, Regular expressions, etc)
Requirements Fields	Cross App Functionality – see Additional Issues
T54A011 **Message pub/sub designer**	**Priority:** Must
	Who: Designer **What:** Allows the user to send a message through the service bus (throw), or wait for a specific one (catch) **Why:** Provide message support as part of the BPMN specification **When:** During process design (without connection to message bus) **Where:** In process designer and the Developer Tier
Acceptance Criteria	User can set properties on the sent message or for the message to subscribe to
Requirements Fields	Cross App Functionality – see Additional Issues
T54A012 **Remove selected element**	**Priority:** Must
	Who: Orchestration Designer **What:** Remove the selected element **Why:** Standard editing function in designers **When:** During process design **Where:** In Process Designer and the Developer Tier
Acceptance Criteria	The selected element is deleted
Requirements Fields	Cross App Functionality – see Additional Issues
T54A013 **Delete process model**	**Priority:** Must
	Who: Orchestration Designer **What:** Delete process model

	Why: Allow user to delete process models
Acceptance Criteria	Process model in storage is deleted and cannot be opened anymore
Requirements Fields	Cross App Functionality – see Additional Issues
T54A014 **Deploy process model**	**Priority:** Must
	Who: User **What:** Deploy process model to process engine in the platform on to the marketplace **Why:** The engine can be executed on the process by any other privileged user **When:** During process design **Where:** In process designer and the Developer Tier
Acceptance Criteria	The process is executed in the process engine
Requirements Fields	Cross App Functionality – see Additional Issues
T54A015 **Get list of process instances**	**Priority:** Must
	Who: User **What:** Get the list of process instances **Why:** Enable a user to assess which processes instances are available **When:** During process design **Where:** In process designer and the Developer Tier
Acceptance Criteria	User can get the list of running instances
Requirements Fields	Cross App Functionality – see Additional Issues

Figure 31: Orchestration Designer Functions

3.2.3 Workflows

This section highlights the user interaction and the interaction of the component in the single function.

3.2.3.1 Process explorer

Read operations with the Storage component, for retrieving and filtering process models.

This step includes:

- Connect to Storage
- Read Process Model
- Search Process Model

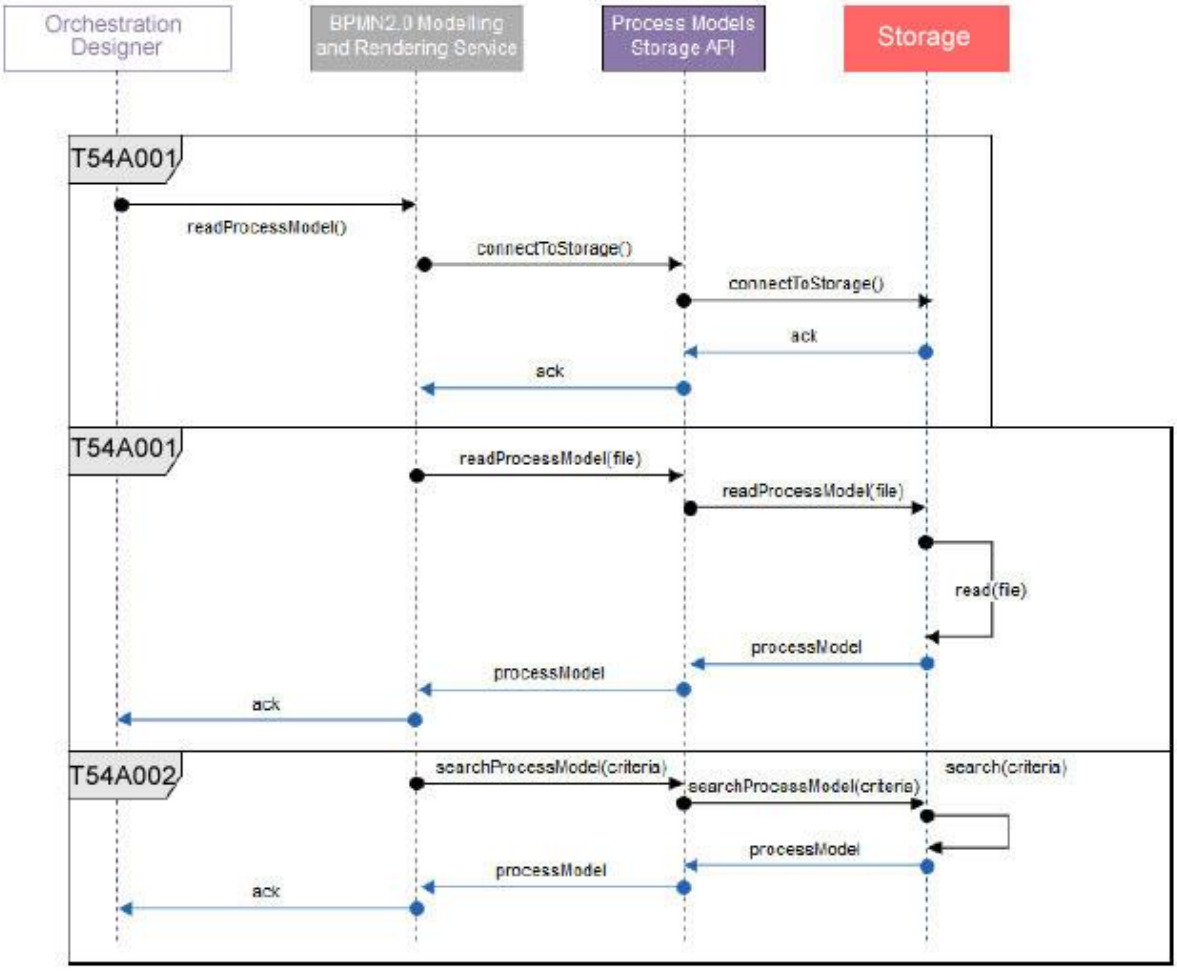


Figure 32: Read operations through Storage

3.2.3.2 Create / Save process

This feature creates the initial BPMN model ready for user interaction. The steps include:

- Connect to Storage
- Create Process Model

All changes from the user are automatically saved/persisted to the Storage by using the Process Model API.

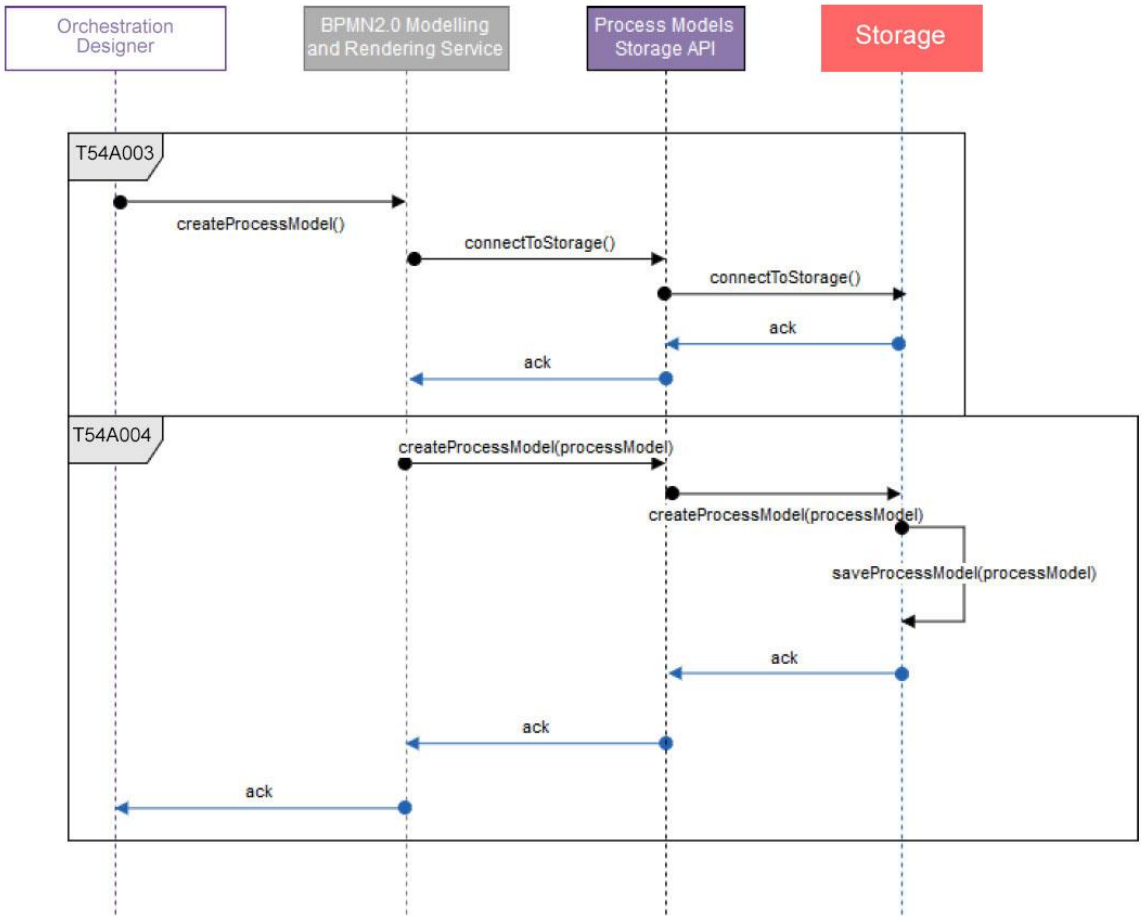


Figure 33: Create / Save process model

3.2.3.3 Deploy process

Deploy process allows the user to send the process model to the process engine, so it can be executed once the user has finished designing/drawing it.

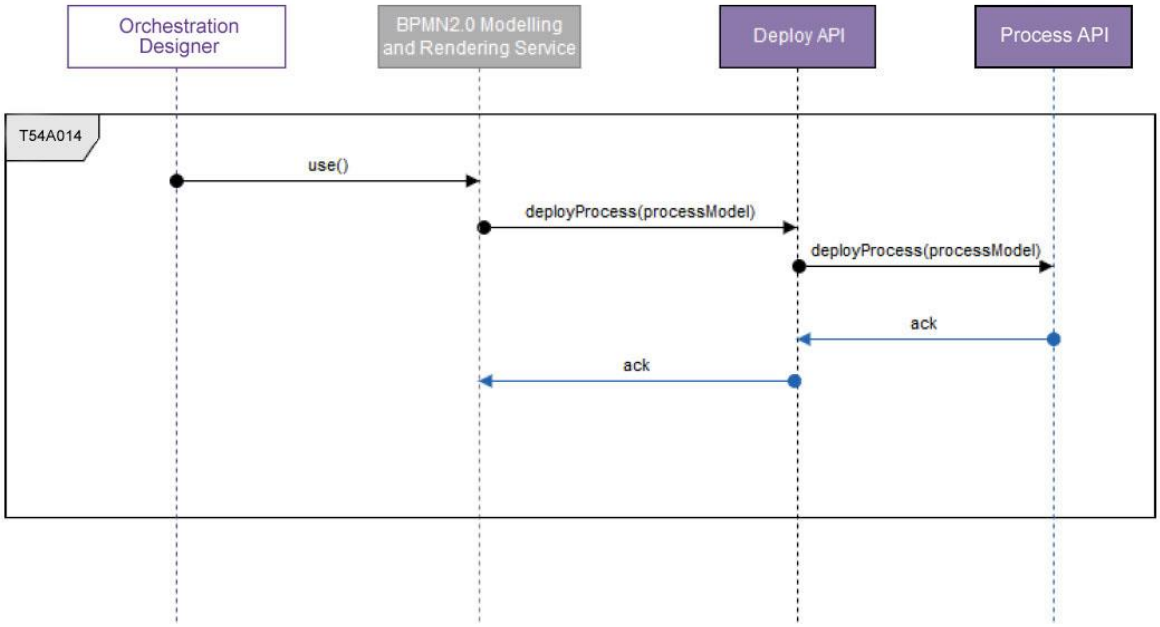


Figure 34: Deploy process model

3.2.3.4 Validate process

Orchestration Designer actively keeps validating the process model on each user interaction to provide real time messages about missing information or invalid state for the current process model.

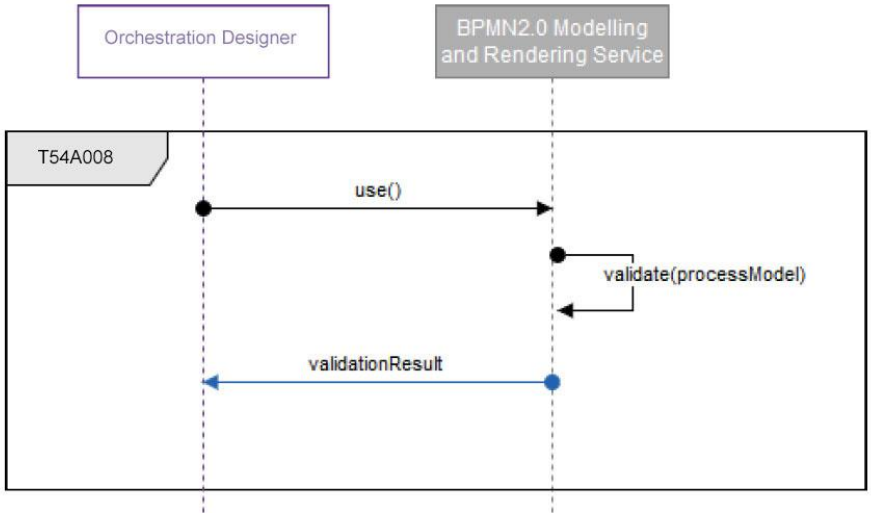


Figure 35: Validate process model

3.2.4 Additional Issues

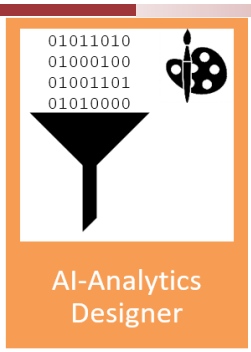
Additional issues have appeared after the description of the architecture.

Issue	Description	Next Steps	Lead (Rationale)
General component	Due to the mentioned Cross-App functionality, these functions above do not match any specific requirements as this component has very general functionality that could be used in many	See Orchestration Runtime	T5.4 Orchestration Runtime

	of the zApps. This is discussed more in the Orchestration Runtime component.		

Figure 36: Additional Issues Orchestration Designer

3.3 AI-Analytics Designer (T5.6)



3.3.1 Overall functional characterisation & Context

The AI-Analytics Designer component provides the ability to define machine learning models from historical data, in order to detect and/or predict any defects in the production process that leads to delay or inconsistency in the delivery of the further products. The machine learning models are built using analytic algorithms based on the statistical-machine-learning linear-algebra libraries. A diagram regarding the AI-Analytics Designer component and its interactions can be found in the Architecture document.

3.3.2 Functions / Features

- **Data extraction**: Historical data from the analysed processes need to be used to discover trends and patterns to build machine learning models. Therefore, the appropriate data needs to be acquired, using different formats, from diverse sources via Historic API
- **In-memory data storage**: All data extracted from historical data sources are stored in structures built in memory – as strings composed by keys and values – with exact consistency semantics and transactions, being thus prepared for algorithms training
- **Supervised algorithms development:** This function deals with machine learning models creation with a supervised learning process. The algorithm iteratively makes predictions on the training data and is corrected by this engine. Learning stops when the algorithm achieves an acceptable level of performance
- **Unsupervised algorithms development**: This feature is building machine learning models in which the learning process is not supervised. Algorithms are left to their own devices to discover and present the interesting structure in the data
- **Models validation**: This is used to validate a model internally by estimation of the model performance without having to sacrifice a validation split
- **Models storing:** This feature acts as a repository used to save binary machine learning models and to load from this repository for future use in validation process or for transform in objects for production
- **Models conversion**: Also called productionising, this function is used to prepare the machine learning models for production by transforming them into either a Plain Old Java Object (POJO) or a Model Object, Optimised (MOJO)
- **Models upload**: Using the Marketplace API, this feature uploads models, scripts, and libraries on the Marketplace and gets specific information on other items

These functions can be further decomposed into the following subtasks that have to be realised:

Subtask	Subtask description
T56A001 **Connect to data source**	**Priority:** Must
	Who: AI-Analytics Designer **Where:** Anywhere **When:** Design time **What:** Opens filesystem of the data source and access the database connections **Why:** So that the schema can be loaded and, thus, the data extraction can be performed
Acceptance Criteria	The schemas (source and/or target) are successfully retrieved

Requirements filled	RQ_0034, RQ_0035, RQ_0038, RQ_0045, RQ_0048, RQ_0051, RQ_0084, RQ_0095, RQ_0101, RQ_0103, RQ_0120, RQ_0137, RQ_0173, RQ_0174, RQ_0175, RQ_0178, RQ_0179, RQ_0180, RQ_0181, RQ_0189, RQ_0194, RQ_0201, RQ_0217, RQ_0218, RQ_0222, RQ_0305, RQ_0305, RQ_0307, RQ_0325, RQ_0329
T56A002 **Read SQL database**	**Priority:** Must
	Who: AI-Analytics Designer **Where:** Anywhere **When:** Design time **What:** Reads SQL database schemas and tables **Why:** So that the Historic API can read the records from database
Acceptance Criteria	The database records are successfully retrieved
Requirements filled	RQ_0034, RQ_0035, RQ_0038, RQ_0045, RQ_0048, RQ_0051, RQ_0084, RQ_0095, RQ_0101, RQ_0103, RQ_0120, RQ_0137, RQ_0173, RQ_0174, RQ_0175, RQ_0178, RQ_0179, RQ_0180, RQ_0181, RQ_0189, RQ_0194, RQ_0201, RQ_0217, RQ_0218, RQ_0222, RQ_0305, RQ_0305, RQ_0307, RQ_0325, RQ_0329
T56A003 **Read JSON**	**Priority:** Must
	Who: AI-Analytics Designer **Where:** Anywhere **When:** Design time **What:** Interprets JSON schema files **Why:** So that the Historic API can read the data
Acceptance Criteria	The JSON schema is successfully interpreted and retrieved
Requirements filled	RQ_0034, RQ_0035, RQ_0038, RQ_0045, RQ_0048, RQ_0051, RQ_0084, RQ_0095, RQ_0101, RQ_0103, RQ_0120, RQ_0137, RQ_0173, RQ_0174, RQ_0175, RQ_0178, RQ_0179, RQ_0180, RQ_0181, RQ_0189, RQ_0194, RQ_0201, RQ_0217, RQ_0218, RQ_0222, RQ_0305, RQ_0305, RQ_0307, RQ_0325, RQ_0329
T56A004 **Read XML**	**Priority:** Must
	Who: AI-Analytics Designer **Where:** Anywhere **When:** Design time **What:** Interprets XML schema files **Why:** So that the schema can be loaded and, thus, the data extraction can be performed
Acceptance Criteria	The XML schema is successfully interpreted and retrieved
Requirements filled	RQ_0034, RQ_0035, RQ_0038, RQ_0045, RQ_0048, RQ_0051, RQ_0084, RQ_0095, RQ_0101, RQ_0103, RQ_0120, RQ_0137, RQ_0173, RQ_0174, RQ_0175, RQ_0178, RQ_0179, RQ_0180, RQ_0181, RQ_0189, RQ_0194, RQ_0201, RQ_0217, RQ_0218, RQ_0222, RQ_0305, RQ_0305, RQ_0307, RQ_0325, RQ_0329
T56A005 **Read CSV**	**Priority:** Must
	Who: AI-Analytics Designer **Where:** Anywhere **When:** Design time **What:** Interprets CSV schema files **Why:** So that the Historic API can read the data
Acceptance Criteria	The CSV schema is successfully interpreted and retrieved
Requirements filled	RQ_0034, RQ_0035, RQ_0038, RQ_0045, RQ_0048, RQ_0051, RQ_0084, RQ_0095, RQ_0101, RQ_0103, RQ_0120, RQ_0137, RQ_0173, RQ_0174, RQ_0175, RQ_0178, RQ_0179, RQ_0180, RQ_0181, RQ_0189, RQ_0194, RQ_0201, RQ_0217, RQ_0218, RQ_0222, RQ_0305, RQ_0305, RQ_0307, RQ_0325, RQ_0329
T56A006 **Transform in K/V**	**Priority:** Must
	Who: AI-Analytics Designer **Where:** Anywhere **When:** Design time **What:** Extracts values from records and pair them with keys **Why:** Store strings as key-value pairs
Acceptance Criteria	The Key/Value strings are successfully created
Requirements filled	RQ_0034, RQ_0035, RQ_0038, RQ_0045, RQ_0048, RQ_0051, RQ_0084, RQ_0095, RQ_0101, RQ_0103, RQ_0120, RQ_0137, RQ_0173, RQ_0174, RQ_0175, RQ_0178, RQ_0179, RQ_0180, RQ_0181, RQ_0189, RQ_0194, RQ_0201, RQ_0217, RQ_0218, RQ_0222, RQ_0305, RQ_0305, RQ_0307, RQ_0325, RQ_0329
T56A007	**Priority:** Must

Display UI	**Who:** AI-Analytics Designer **Where:** Anywhere **When:** Design time **What**: Displays data retrieved and K/V strings created **Why:** So that the user can validate and save data extracted
Acceptance Criteria	The data is successfully displayed
Requirements filled	RQ_0034, RQ_0035, RQ_0038, RQ_0045, RQ_0048, RQ_0051, RQ_0084, RQ_0095, RQ_0101, RQ_0103, RQ_0120, RQ_0137, RQ_0173, RQ_0174, RQ_0175, RQ_0178, RQ_0179, RQ_0180, RQ_0181, RQ_0189, RQ_0194, RQ_0201, RQ_0217, RQ_0218, RQ_0222, RQ_0305, RQ_0305, RQ_0307, RQ_0325, RQ_0329
T56A008 **Write K/V**	**Priority:** Must
	Who: AI-Analytics Designer **Where:** Anywhere **When:** Design time **What:** Stores in memory structures with K/V strings **Why:** So that the in-memory structured can be used in models training
Acceptance Criteria	The in-memory structures successfully created
Requirements filled	RQ_0034, RQ_0035, RQ_0038, RQ_0045, RQ_0048, RQ_0051, RQ_0084, RQ_0095, RQ_0101, RQ_0103, RQ_0120, RQ_0137, RQ_0173, RQ_0174, RQ_0175, RQ_0178, RQ_0179, RQ_0180, RQ_0181, RQ_0189, RQ_0194, RQ_0201, RQ_0217, RQ_0218, RQ_0222, RQ_0305, RQ_0305, RQ_0307, RQ_0325, RQ_0329
T56A009 **Build Supervised Algorithm**	**Priority:** Must
	Who: AI-Analytics Designer **Where:** Anywhere **When:** Design time **What:** Creates supervised algorithms **Why:** So that the supervised machine learning model can be trained
Acceptance Criteria	The supervised algorithms successfully created
Requirements filled	RQ_0034, RQ_0035, RQ_0038, RQ_0045, RQ_0048, RQ_0051, RQ_0084, RQ_0095, RQ_0101, RQ_0103, RQ_0120, RQ_0137, RQ_0173, RQ_0174, RQ_0175, RQ_0178, RQ_0179, RQ_0180, RQ_0181, RQ_0189, RQ_0194, RQ_0201, RQ_0217, RQ_0218, RQ_0222, RQ_0305, RQ_0305, RQ_0307, RQ_0325, RQ_0329
T56A010 **Train Supervised Algorithm**	**Priority:** Must
	Who: AI-Analytics Designer **Where:** Anywhere **When:** Design time **What:** Runs supervised algorithms with historic data. **Why:** So that algorithm makes predictions iteratively on the training data and is corrected by this engine
Acceptance Criteria	The algorithm achieves an acceptable level of performance
Requirements filled	RQ_0034, RQ_0035, RQ_0038, RQ_0045, RQ_0048, RQ_0051, RQ_0084, RQ_0095, RQ_0101, RQ_0103, RQ_0120, RQ_0137, RQ_0173, RQ_0174, RQ_0175, RQ_0178, RQ_0179, RQ_0180, RQ_0181, RQ_0189, RQ_0194, RQ_0201, RQ_0217, RQ_0218, RQ_0222, RQ_0305, RQ_0305, RQ_0307, RQ_0325, RQ_0329
T56A011 **Validate Model**	**Priority:** Must
	Who: AI-Analytics Designer **Where:** Anywhere **When:** Design time **What:** Runs the trained machine learning model to estimate the performance. **Why:** So that a model validation at a design time is efficient than ones at a runtime
Acceptance Criteria	The performance level of the machine learning model is confirmed
Requirements filled	RQ_0034, RQ_0035, RQ_0038, RQ_0045, RQ_0048, RQ_0051, RQ_0084, RQ_0095, RQ_0101, RQ_0103, RQ_0120, RQ_0137, RQ_0173, RQ_0174, RQ_0175, RQ_0178, RQ_0179, RQ_0180, RQ_0181, RQ_0189, RQ_0194, RQ_0201, RQ_0217, RQ_0218, RQ_0222, RQ_0305, RQ_0305, RQ_0307, RQ_0325, RQ_0329
T56A012	**Priority:** Must

Build Unsupervised Algorithm	**Who:** AI-Analytics Designer **Where:** Anywhere **When:** Design time **What:** Creates unsupervised algorithms **Why:** So that the unsupervised machine learning model can be trained
Acceptance Criteria	The unsupervised algorithms successfully created
Requirements filled	RQ_0034, RQ_0035, RQ_0038, RQ_0045, RQ_0048, RQ_0051, RQ_0084, RQ_0095, RQ_0101, RQ_0103, RQ_0120, RQ_0137, RQ_0173, RQ_0174, RQ_0175, RQ_0178, RQ_0179, RQ_0180, RQ_0181, RQ_0189, RQ_0194, RQ_0201, RQ_0217, RQ_0218, RQ_0222, RQ_0305, RQ_0305, RQ_0307, RQ_0325, RQ_0329
T56A013 **Train Unsupervised Algorithm**	**Priority:** Must
	Who: AI-Analytics Designer **Where:** Anywhere **When:** Design time **What:** Runs unsupervised algorithms with historic data **Why:** So that the learning process is not supervised, thus algorithms are left to their own devices to discover patterns in the data
Acceptance Criteria	The algorithm discovers patterns successfully
Requirements filled	RQ_0034, RQ_0035, RQ_0038, RQ_0045, RQ_0048, RQ_0051, RQ_0084, RQ_0095, RQ_0101, RQ_0103, RQ_0120, RQ_0137, RQ_0173, RQ_0174, RQ_0175, RQ_0178, RQ_0179, RQ_0180, RQ_0181, RQ_0189, RQ_0194, RQ_0201, RQ_0217, RQ_0218, RQ_0222, RQ_0305, RQ_0305, RQ_0307, RQ_0325, RQ_0329
T56A014 **Display Model Validation**	**Priority:** Must
	Who: AI-Analytics Designer **Where:** Anywhere **When:** Design time **What:** Displays graphical representations of data validated by the machine learning model **Why:** So that the model performance is verified
Acceptance Criteria	The graphical representation of data validated is successfully displayed
Requirements filled	RQ_0034, RQ_0035, RQ_0038, RQ_0045, RQ_0048, RQ_0051, RQ_0084, RQ_0095, RQ_0101, RQ_0103, RQ_0120, RQ_0137, RQ_0173, RQ_0174, RQ_0175, RQ_0178, RQ_0179, RQ_0180, RQ_0181, RQ_0189, RQ_0194, RQ_0201, RQ_0217, RQ_0218, RQ_0222, RQ_0305, RQ_0305, RQ_0307, RQ_0325, RQ_0329
T56A015 **Save Model**	**Priority:** Must
	Who: AI-Analytics Designer **Where:** Anywhere **When:** Design time **What:** Saves the binary machine learning model in the repository **Why:** So that the model can be used later for validation or in production
Acceptance Criteria	The model is successfully saved.
Requirements filled	RQ_0034, RQ_0035, RQ_0038, RQ_0045, RQ_0048, RQ_0051, RQ_0084, RQ_0095, RQ_0101, RQ_0103, RQ_0120, RQ_0137, RQ_0173, RQ_0174, RQ_0175, RQ_0178, RQ_0179, RQ_0180, RQ_0181, RQ_0189, RQ_0194, RQ_0201, RQ_0217, RQ_0218, RQ_0222, RQ_0305, RQ_0305, RQ_0307, RQ_0325, RQ_0329
T56A016 **Load Model**	**Priority:** Must
	Who: AI-Analytics Designer **Where:** Anywhere **When:** Design time **What:** Loads the binary machine learning model from the repository **Why:** So that the model is used for validation or for productionising
Acceptance Criteria	The model is successfully loaded for validation / productionising
Requirements filled	RQ_0034, RQ_0035, RQ_0038, RQ_0045, RQ_0048, RQ_0051, RQ_0084, RQ_0095, RQ_0101, RQ_0103, RQ_0120, RQ_0137, RQ_0173, RQ_0174, RQ_0175, RQ_0178, RQ_0179, RQ_0180, RQ_0181, RQ_0189, RQ_0194, RQ_0201, RQ_0217, RQ_0218, RQ_0222, RQ_0305, RQ_0305, RQ_0307, RQ_0325, RQ_0329
T56A017 **Convert to POJO**	**Priority:** Must
	Who: AI-Analytics Designer **Where:** Anywhere

	When: Design time **What:** Transforms a machine learning model in a Plain Old Java Object (POJO) **Why:** So that the POJO can be used for production, at runtime
Acceptance Criteria	POJO successfully created
Requirements filled	RQ_0034, RQ_0035, RQ_0038, RQ_0045, RQ_0048, RQ_0051, RQ_0084, RQ_0095, RQ_0101, RQ_0103, RQ_0120, RQ_0137, RQ_0173, RQ_0174, RQ_0175, RQ_0178, RQ_0179, RQ_0180, RQ_0181, RQ_0189, RQ_0194, RQ_0201, RQ_0217, RQ_0218, RQ_0222, RQ_0305, RQ_0305, RQ_0307, RQ_0325, RQ_0329
T56A018 **Convert to MOJO**	**Priority:** Must
	Who: AI-Analytics Designer **Where:** Anywhere **When:** Design time **What:** Transforms a machine learning model in a Model Object, Optimised (MOJO) **Why:** So that the MOJO can be used for production, at runtime
Acceptance Criteria	POJO successfully created
Requirements filled	RQ_0034, RQ_0035, RQ_0038, RQ_0045, RQ_0048, RQ_0051, RQ_0084, RQ_0095, RQ_0101, RQ_0103, RQ_0120, RQ_0137, RQ_0173, RQ_0174, RQ_0175, RQ_0178, RQ_0179, RQ_0180, RQ_0181, RQ_0189, RQ_0194, RQ_0201, RQ_0217, RQ_0218, RQ_0222, RQ_0305, RQ_0305, RQ_0307, RQ_0325, RQ_0329
T56A019 **Deploy model**	**Priority:** Must
	Who: AI-Analytics Designer **Where:** Anywhere **When:** Design time **What:** Deploys POJO or MOJO objects using Deploy API **Why:** So that the POJO and MOJO object can be used by other components in ZDMP
Acceptance Criteria	POJO / MOJO successfully deployed.
Requirements filled	RQ_0034, RQ_0035, RQ_0038, RQ_0045, RQ_0048, RQ_0051, RQ_0084, RQ_0095, RQ_0101, RQ_0103, RQ_0120, RQ_0137, RQ_0173, RQ_0174, RQ_0175, RQ_0178, RQ_0179, RQ_0180, RQ_0181, RQ_0189, RQ_0194, RQ_0201, RQ_0217, RQ_0218, RQ_0222, RQ_0305, RQ_0305, RQ_0307, RQ_0325, RQ_0329
T56A020 **Upload model**	**Priority:** Must
	Who: AI-Analytics Designer **Where:** Anywhere **When:** Design time **What:** Uploads a machine learning model to Marketplace, using Marketplace API **Why:** So that the model can be used by other developers
Acceptance Criteria	Model successfully uploaded
Requirements filled	RQ_0034, RQ_0035, RQ_0038, RQ_0045, RQ_0048, RQ_0051, RQ_0084, RQ_0095, RQ_0101, RQ_0103, RQ_0120, RQ_0137, RQ_0173, RQ_0174, RQ_0175, RQ_0178, RQ_0179, RQ_0180, RQ_0181, RQ_0189, RQ_0194, RQ_0201, RQ_0217, RQ_0218, RQ_0222, RQ_0305, RQ_0305, RQ_0307, RQ_0325, RQ_0329
T56A021 **Upload Script**	**Priority:** Must
	Who: AI-Analytics Designer **Where:** Anywhere **When:** Design time **What:** Uploads a Python script to Marketplace, using Marketplace API **Why:** So that the script can be used by other developers
Acceptance Criteria	Script successfully uploaded
Requirements filled	RQ_0034, RQ_0035, RQ_0038, RQ_0045, RQ_0048, RQ_0051, RQ_0084, RQ_0095, RQ_0101, RQ_0103, RQ_0120, RQ_0137, RQ_0173, RQ_0174, RQ_0175, RQ_0178, RQ_0179, RQ_0180, RQ_0181, RQ_0189, RQ_0194, RQ_0201, RQ_0217, RQ_0218, RQ_0222, RQ_0305, RQ_0305, RQ_0307, RQ_0325, RQ_0329
T56A022 **Upload Library**	**Priority:** Must
	Who: AI-Analytics Designer **Where:** Anywhere **When:** Design time **What:** Uploads a library to Marketplace, using Marketplace API **Why:** So that the script can be used by other developers

Acceptance Criteria	Script successfully uploaded.
Requirements filled	RQ_0034, RQ_0035, RQ_0038, RQ_0045, RQ_0048, RQ_0051, RQ_0084, RQ_0095, RQ_0101, RQ_0103, RQ_0120, RQ_0137, RQ_0173, RQ_0174, RQ_0175, RQ_0178, RQ_0179, RQ_0180, RQ_0181, RQ_0189, RQ_0194, RQ_0201, RQ_0217, RQ_0218, RQ_0222, RQ_0305, RQ_0305, RQ_0307, RQ_0325, RQ_0329

Figure 37: AI-Analytics Designer Functions

3.3.3 Workflows

The following sub-sections describe the sequence diagrams of the AI-Analytics Designer component.

3.3.3.1 Connect to data source

The following diagram explains this function and the necessary interactions with other components.

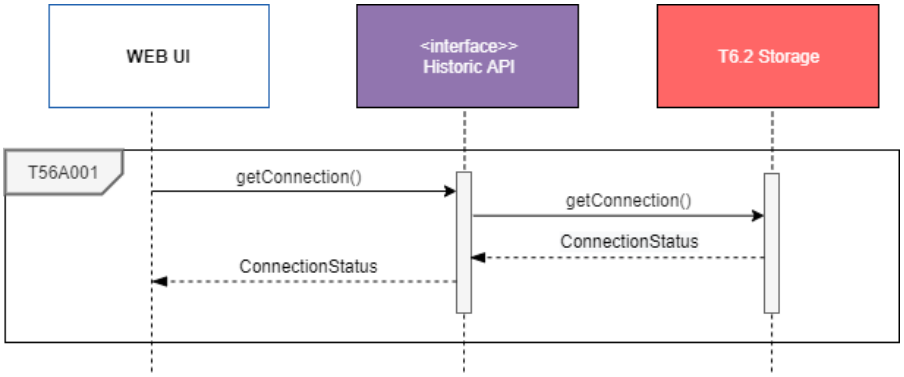


Figure 38: Connect to data source sequence diagram

3.3.3.2 Read SQL Database

The following diagram explains this function and the necessary interactions with other components.

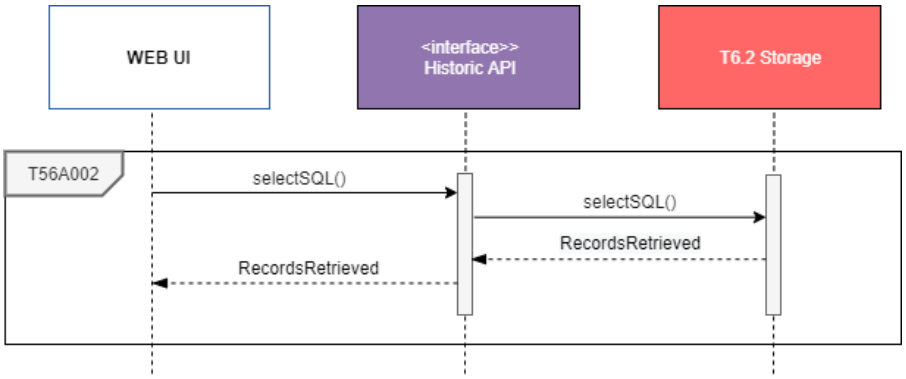


Figure 39: Read SQL Database

3.3.3.3 Read NoSQL Data

The following diagram explains this function and the necessary interactions with other components.

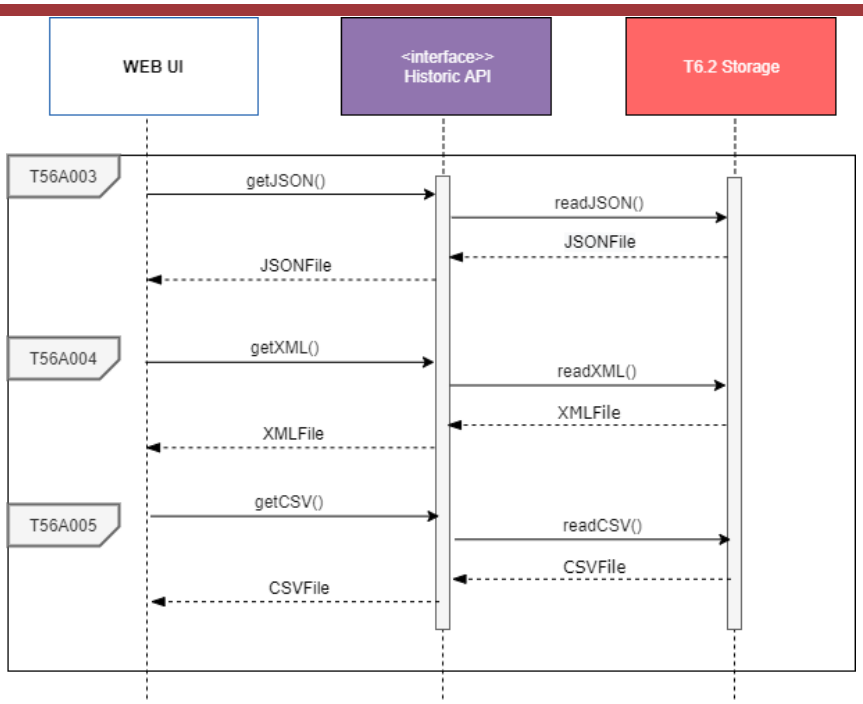


Figure 40: Read NoSQL Data sequence diagram

3.3.3.4 Transform, display, and store K/V

The following diagram explains this function and the necessary interactions with other components.

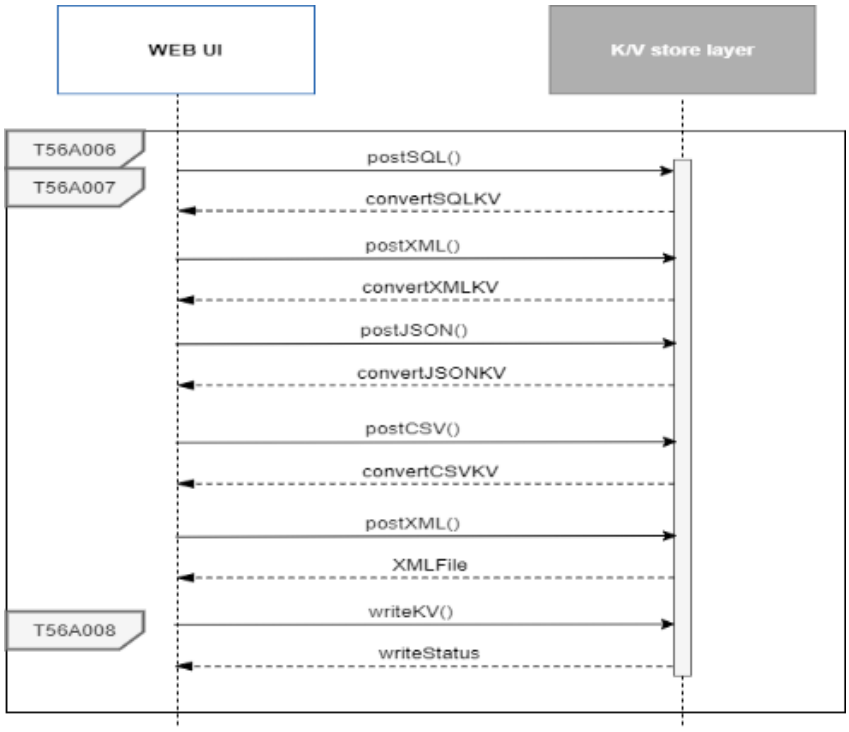


Figure 41: Transform, display and store K/V sequence diagram

3.3.3.5 Build, train and validate supervised models

The following diagram explains this function and the necessary interactions with other components.

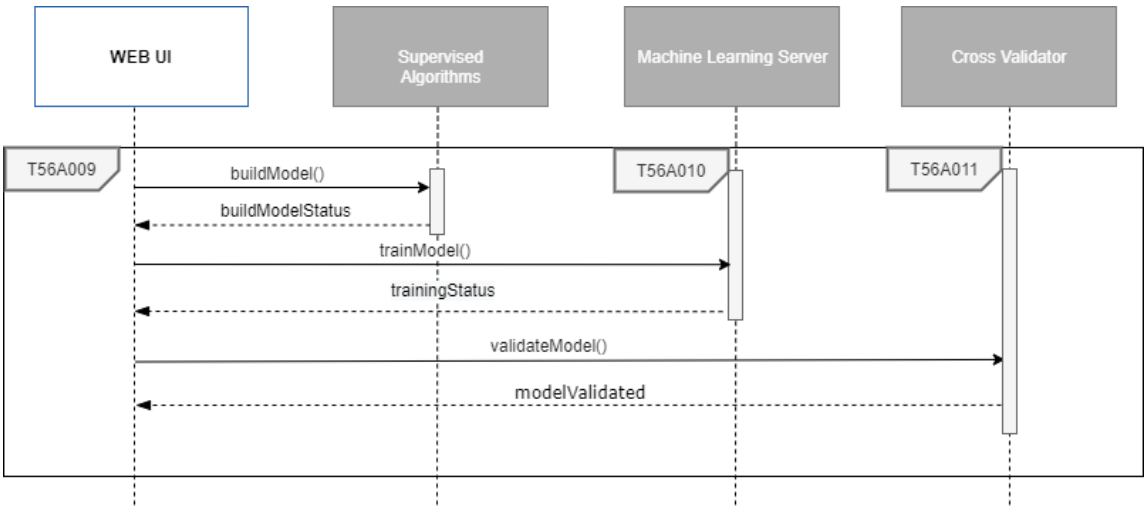


Figure 42: Build, train and validate supervised models sequence diagram

3.3.3.6 Build, train and validate unsupervised models

The following diagram explains this function and the necessary interactions with other components.

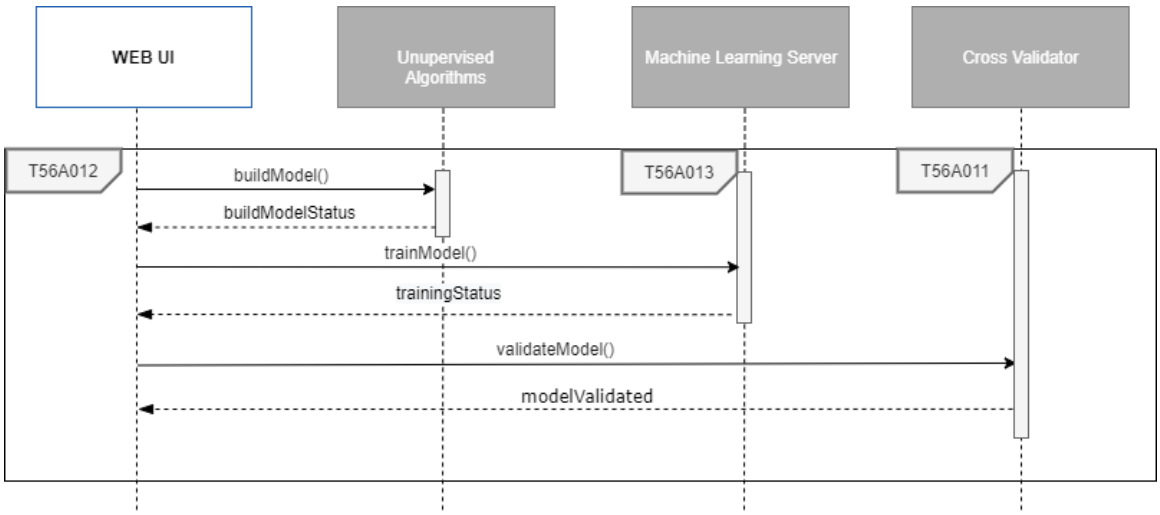


Figure 43: Build, train and validate unsupervised models sequence diagram

3.3.3.7 Save and load models

The following diagram explains this function and the necessary interactions with other components.

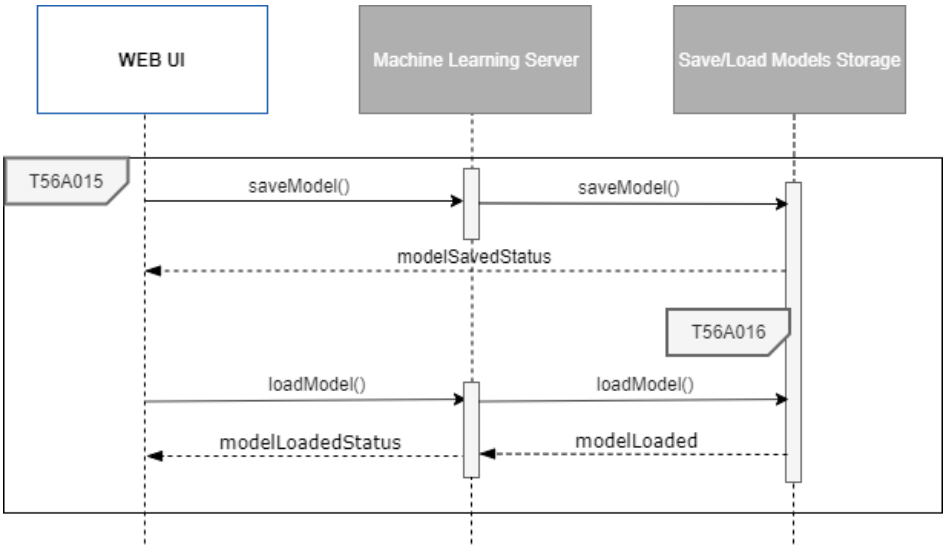


Figure 44: Save and load models sequence diagram

3.3.3.8 Productionising and deploy models

The following diagram explains this function and the necessary interactions with other components.

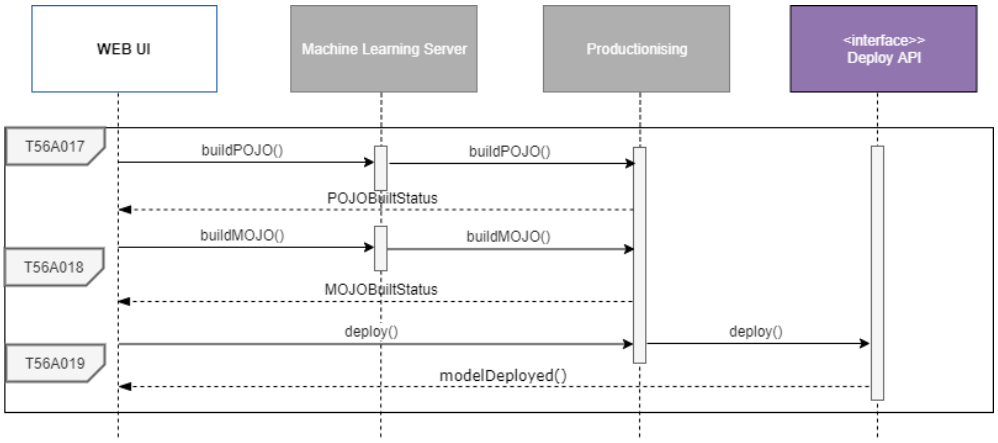


Figure 45: Productionising and deploy models sequence diagram

3.3.3.9 Upload models, scripts, and libraries

The following diagram explains this function and the necessary interactions with other components.

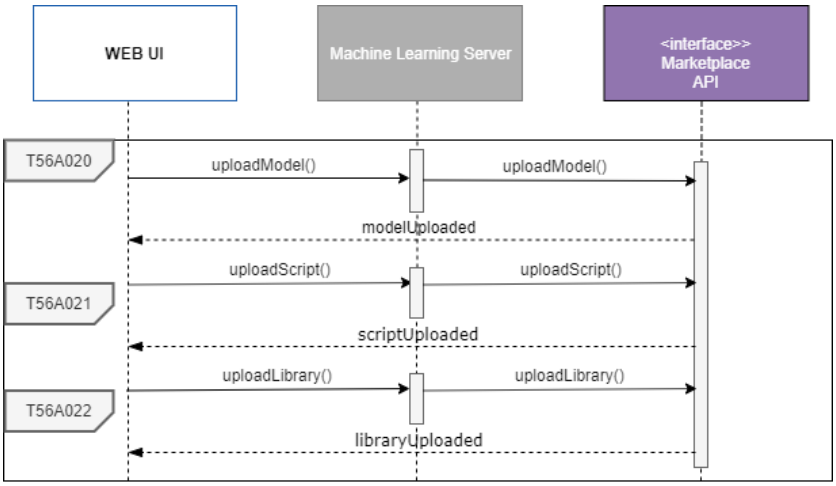


Figure 46: Upload models, scripts, and libraries sequence diagram

3.4 Applications Builder (T6.1)

3.4.1 Overall functional characterization & Context

The application builder is a HTML / CSS / JS based web frontend that enables developers to build a basic infrastructure of zApps using drag and drop components.

These drag and drop components are already linked with APIs provided by other technical components and include calls and bindings to ZDMP services. This enables to accomplish certain basic functionalities that every app needed including notifications, error reporting, sign up forms and dashboards for data visualization, etc.

These elements and functionalities that can be added to such zApps and services are described as follows:

- **Error Reporting**: Where errors in ZDMP Assets are recognised and for troubleshooting reasons reports are forwarded to the administration area of the Secure Business Cloud. There, developers can examine the error reports to react quickly with bug fixes. Error Reports consist of information about the sender, client operating system, application, time, and the error itself (eg exception stacktrace).
- **User Authorisation and Authentication:** The user can register or login to ZDMP with the function easily. A UI is provided with a connection to the security, which manages the registration and login process. For the registration, the user enters a valid email address, first name, and last name. The security component then takes over and replies with a confirmation email to the entered email address, as double-opt in confirmation by the email user is necessary to comply with GDPR. In the case of a login, only the user credentials (username and password) are necessary. Again, the security component takes over for the verification of the credentials and responds with granted or denied access.
- **Multi Language Support:** Internationalization is a major feature every app needs to support. In this, the user can switch languages in any ZDMP app. Also, a resource editor (Translator) will be provided, which shows all strings in a default language (English) that can then be translated into any language the user speaks or be switched to any language that is already provided. After translation has been performed, the user can request to add the translation to the official application. Then it depends on the developer if the new translation is accepted or declined.
- **Application Logging:** Logging is a default debugging mechanism used during development of an app as well as at runtime for bug-fixing. With this, internal processes are logged and saved to an additional logfile. This file then can give the developer hints about possible error messages and eases troubleshooting. Information in the log can be displayed by a view in the client application.
- **Notification Template:** With this, developers can provide the functionality to raise notifications, to make users aware of relevant information, depending on the user management system of the ZDMP security component
- **Automatic Updates:** Mechanism used to maintain zApps after they are development and uploaded on Marketplace. This provides functionality of automatic updating apps after bug fixes and extension in features by the zApp developer

When the developer is finished, a fully functional zApp will be created, then uploaded to the Secure Business Cloud. After this process the new zApp can be rolled out to clients / users. This component is a starting point for zApp developers, having a reason for programming a zApp for problems that are not solvable out of the box. By creating custom functionalities, the developer will provide value to users / customers through a zApp. The zApp template created by the app builder is expected to make development easier and faster.

A zApp will be composed of following UI Elements and templates:

- **Provide UI Elements:** Various UI elements are provided in the appbuilder to be used by developers. These elements come with its default properties but are also customisable for its respective purpose. This includes charts and diagrams that can be data-bound to ZDMP services (zAssets), as well as an “About” page with information and informative links.
- **Provide UI Templates:** Appbuilder provides UI Templated which are compositions of UI elements from the UI Element Repository. Each template appears as a single UI element but is composed of several UI predefined elements. A lightweight example of a UI template is a login form, which is composed of a grid, textbox, password-textbox, and buttons.
- **Provide Behaviour Elements:** Behaviour elements provide default behaviours for activities, which are related to UI interactions. Those behaviours are easy to integrate and applicable for eg forms, downloads, registration, notifications, etc.
- **Provide Behaviour Templates:** Behaviour template consists of lined up behaviours that are processed synchronously. Lined up behaviours ease the use of handling events and supports developers with additional default events.
- **Provide Holistic Templates:** Holistic templates provided by appbuilder unite UI templates and behaviour templates. Those templates include logic to manage user interactions automatically, eg a zApp template that already includes error management, contact forms with default mail templates, etc

3.4.2 Functions / Features

There function can be grouped to the following features which have to be implemented:

Subtask	Subtask description
T61A001 **Get List of Stored Configurations**	**Priority:** Must
	Who: zApps Developer **When**: Design time while developing zApps and service, runtime in ZDMP assets **Where**: Anywhere **What:** Lists existing SDK configuration files, getting their names, detail on functionality, version, etc. **Why:** Browse existing configurations to model the SDK
Acceptance Criteria	Invoker got a structured list of configurations
Requirements filled	RQ_0111, RQ_0194, RQ_0247, RQ_0319
T61A002 **Get zApps APIs**	**Priority:** Must
	Who: zApps Developer **When**: Design time while developing zApps and service, runtime in ZDMP assets with a valid access token **Where**: Anywhere **What:** Retrieves the APIs used by the selected zApp **Why:** Make use of the APIs
Acceptance Criteria	Invoker got APIs definition and description

Requirements filled	RQ_0194, RQ_0247
T61A003 **Get ZDMP Asset APIs and Manifests**	**Priority:** Must
	Who: zApps Developer **When**: Design time in the app builder **Where**: Design time while developing zApps and service, runtime in ZDMP assets with a valid access token **What:** Retrieves the API and manifest files for the selected ZDMP asset **Why:** Make use of APIs and manifests
Acceptance Criteria	Invoker got API and manifest definition and description
Requirements filled	RQ_0111, RQ_0161, RQ_0219
T61A004 **Get Data Model definitions**	**Priority:** Must
	Who: zApps Developer **When**: Design time while developing zApps and service, runtime in ZDMP assets with a valid access token **Where**: Anywhere **What:** Retrieves APIs and manifest files for the selected data model / design pattern **Why:** Make use of APIs and manifests
Acceptance Criteria	Invoker got API and manifest definition and description
Requirements filled	RQ_0111, RQ_0161, RQ_0219
T61A005 **Get Configuration**	**Priority:** Must
	Who: zApps Developer **When**: Design time while developing zApps and service **Where**: In app builder **What:** Retrieves data for the selected configuration file stored in the Data Storage **Why:** Use data for configuration of the SDK, Process Designer or Studio
Acceptance Criteria	Invoker got the selected Configuration information and its description
Requirements filled	RQ_0111, RQ_0194, RQ_0247, RQ_0319
T61A006 **Retrieve Composition data**	**Priority:** Must
	Who: zApps Developer **When**: Design time while developing zApps and service **Where**: In app builder **What:** Submits a set of APIs and Manifests describing the services, and particularly a manifest that describes how the services should be composed **Why:** To provide all information needed for the zApp build
Acceptance Criteria	zApp Composer received the set of APIs and Manifests corresponding to the zApp being developed
Requirements filled	RQ_0161, RQ_0194, RQ_0219
T61A007 **Validate Dependencies**	**Priority:** Must
	Who: zApp Composer **When**: Design time while developing zApps and service **Where**: In app builder **What:** Analyses received data corresponding to the development of a zApp and check the dependencies of the various involved modules **Why:** To ensure all information needed for performing the zApp build is available
Acceptance Criteria	All dependencies of the received modules are included and accessible, whether using local ZDMP repositories (Data Storage, ZDMP-Store, Developer Engagement Hub) or remote (internet, local machine uploading)
Requirements filled	RQ_0111, RQ_0161, RQ_0194, RQ_0219, RQ_0247
T61A008 **Structure the Build Manifest**	**Priority:** Must
	Who: zApp Composer **When**: Design time while writing manifest for zApps and service **Where**: In app builder **What:** Ensure that the Build manifest complies to a set of process steps that are

	meaningful **Why:** To ensure the process steps needed for performing the build make sense and are listed in an appropriate way
Acceptance Criteria	zApp Composer returns success on the analysis of the zApp Build Manifest
Requirements filled	RQ_0111, RQ_0161, RQ_0219
T61A009 **Create Error Report**	**Priority:** Must
	Who: Application Builder, Asset **When**: Run time in zApps when a bug occurs in live app **Where**: In zApps **What:** Uncaught exceptions are intercepted and afterwards transformed in a readable format **Why:** To increase readability of error reports
Acceptance Criteria	The result of the creation should be a valid report model, which can be parsed into a valid JSON format
Requirements filled	RQ_0161, RQ_0179, RQ_0194, RQ_0219
T61A010 **Send Error Report Automatically**	**Priority:** Must
	Who: Application Builder, Secure Business Cloud **When**: Run time in zApps when a bug occurs in live app **Where**: In zApps **What:** Errors are submitted to the Secure Business Cloud Backend to inform the developer about such errors **Why:** To increase the quality of ZDMP-Apps in the Secure Business Cloud
Acceptance Criteria	An error report is sent to the Secure Business Cloud backend and can be viewed there
Requirements filled	RQ_0161, RQ_0179, RQ_0194, RQ_0219
T61A011 **Send Error Report Manually**	**Priority:** Must
	Who: Application Builder, Secure Business Cloud **When**: Run time in zApps when a bug occurs in live app **Where**: In zApps **What:** Errors are submitted to the Secure Business Cloud Backend containing the email address of the user as a contact person in order to inform the developer about such errors **Why:** To increase the quality of ZDMP-Apps in the Secure Business Cloud
Acceptance Criteria	An error report is sent to the Secure Business Cloud backend and can be viewed there including the contact information (email)
Requirements filled	RQ_0161, RQ_0179, RQ_0194, RQ_0219
T61A012 **Forward Error Report**	**Priority:** Must
	Who: Application Builder, Secure Business Cloud **When**: Design time in the app builder, In zApps **Where**: Anywhere **What:** Errors are forwarded to the Secure Business Cloud Backend containing the email address of the user as a contact person in order to inform the developer about such errors **Why:** To increase the quality of ZDMP-Apps in the Secure Business Cloud
Acceptance Criteria	An error report is sent to the Secure Business Cloud backend and can be viewed there including the contact information (email).
Requirements filled	RQ_0161, RQ_0194, RQ_0219
T61A013 **Provide a Register form for new Users**	**Priority:** Must
	Who: Application Builder **When**: Design time in the app builder, runtime in live zApps **Where**: In app builder and zApps **What:** Provides a form to register a new user in ZDMP **Why:** To enable a common behaviour of creating new user accounts
Acceptance Criteria	A user must be created and able to login afterwards

Requirements filled	RQ_0161, RQ_0179, RQ_0201, RQ_0222, RQ_0219, RQ_0350, RQ_0462, RQ_0474, RQ_0492
T61A014 Provide User Login Form	**Priority:** Must
	Who: Application Builder **When**: Design time in the app builder, runtime in live zApps **Where**: In app builder and zApps **What**: Shows a popup of a User Login Form if an action needs to be authorised, and the user is not logged in currently **Why**: To give a common login mask including a behaviour template for "authorised only" actions
Acceptance Criteria	Only authorised users shall be able to act for specific actions with limited access
Requirements filled	RQ_0161, RQ_0179, RQ_0201, RQ_0222, RQ_0219, RQ_0350, RQ_0462, RQ_0474, RQ_0492
T61A015 Provide a button to reset a password	**Priority:** Must
	Who: Application Builder **When**: Design time in the app builder, runtime in live zApps **Where**: In app builder and zApps **What**: Provides a button which forwards to a form to reset the password **Why**: A user is not forced to create a new user account. They can just reset the password if it is forgotten
Acceptance Criteria	A new password is sent via mail to the user
Requirements filled	RQ_0161, RQ_0179, RQ_0201, RQ_0222, RQ_0219, RQ_0350, RQ_0462, RQ_0474, RQ_0492
T61A016 Provide a form to reset a password	**Priority:** Must
	Who: Application Builder **When**: Design time in the app builder, runtime in live zApps **Where**: In app builder and zApps **What**: Provides a form to fill in only the email address of the user, which is already registered **Why**: To be able to send a password reset mail to the right user
Acceptance Criteria	The password reset mail is sent to the user
Requirements filled	RQ_0161, RQ_0179, RQ_0201, RQ_0219, RQ_0350, RQ_0462, RQ_0474, RQ_0492
T61A017 Translate text	**Priority:** Should
	Who: Application Builder **When**: Design time in the app builder, runtime in live zApps **Where**: In app builder and zApps **What**: Provides an option to translate the current language in any other language **Why**: To support the community and decrease the language barrier
Acceptance Criteria	Translation can be saved and viewed afterwards in the ZDMP Asset
Requirements filled	RQ_0091, RQ_0179, RQ_0350, RQ_0462, RQ_0474, RQ_0492
T61A018 Switch language	**Priority:** Should
	Who: Application Builder **When**: Design time in the app builder, runtime in live zApps **Where**: In app builder and zApps **What**: The user can choose between several (if provided) languages and select the intended language to use for this ZDMP Asset **Why**: To provide any language for ZDMP Assets
Acceptance Criteria	The text in the ZDMP Asset is changed to the provided translation
Requirements filled	RQ_0161, RQ_0179, RQ_0219, RQ_0350, RQ_0462, RQ_0474, RQ_0492
T61A019	**Priority:** Must

Provide UI Elements	**Who:** Application Builder **When**: Design time in the app builder **Where**: In app builder **What:** Developers can take UI elements from the UI repository to use them in their ZDMP Assets, which are under development **Why:** To ease and accelerate the development of ZDMP Assets
Acceptance Criteria	UI elements are available in the ZDMP-Studio
Requirements filled	RQ_0161, RQ_0179, RQ_0201, RQ_0222, RQ_0219, RQ_0350, RQ_0462, RQ_0474, RQ_0492
T61A020 **Show Notification**	Priority: Should
	Who: Application Builder **When**: Runtime in zApps and services **Where**: In zApps and services **What:** Notification will be shown on a display, in case of a triggered event. **Why:** To make a user aware of noteworthy information
Acceptance Criteria	A notification is shown when it is expected after a triggered event
Requirements filled	RQ_0161, RQ_0179, RQ_0180, RQ_0194, RQ_0219
T61A021 **Dismiss Notification**	Priority: Should
	Who: Application Builder **When**: Runtime in zApps and services **Where**: In zApps and services **What:** Developers can add a dismiss functionality to make notifications disappear **Why:** To remove a notification that has already been read or ignored
Acceptance Criteria	The notification disappears after dismissing it
Requirements filled	RQ_0161, RQ_0179, RQ_0180, RQ_0194, RQ_0219
T61A022 **Remind Later**	Priority: Should
	Who: Application Builder **When**: Design time in the app builder, runtime in zApps **Where**: In app builder and zApps **What:** Developers can set a timer to appear and a notification after the timer is elapsed **Why:** To remind users of valuable information if they have ignored the information before
Acceptance Criteria	The notifications appear again as soon as a defined timer is elapsed
Requirements filled	RQ_0161, RQ_0179, RQ_0180, RQ_0194, RQ_0219
T61A023 **Manifest reading**	**Priority:** Must
	Who: ZDMP-OS Drivers **When**: Design time in the app builder **Where**: In app builder **What**: Returns a manifest file from an installed driver **Why**: So that ZDMP-OS enablers can query drivers on their usage
Acceptance Criteria	Existing files with drivers must be installed previously by the ZDMP-OS Marketplace Manifest interface retrieves a json file associated to a given driver required in a request
Requirements filled	RQ_0161, RQ_0219

Figure 47: Applications Builder Features

3.4.3 Workflows

This section is highlighting the user interaction and the interaction of the component in Application Builder UI.

3.4.3.1 Invoke Plugin API

The SDK includes the possibility of working with different modules such as application builders, composers, and can be extended to reuse any generic type of component, simply exposing its API to the SDK clients. These are plug-in modules that can be invoked by the SDK. Hence, these plugins also need to be connected to the SDK and requested to be accessed by their APIs, as shown in the following figure.

The main steps / functionalities are:

- Invoking the SDK for calling the desired functionality
- The SDK determines if there is any configuration needed for the execution
- The SDK completes the component configuration and invokes it to retrieve its API list
- The SDK returns to the caller the API of the plugged component

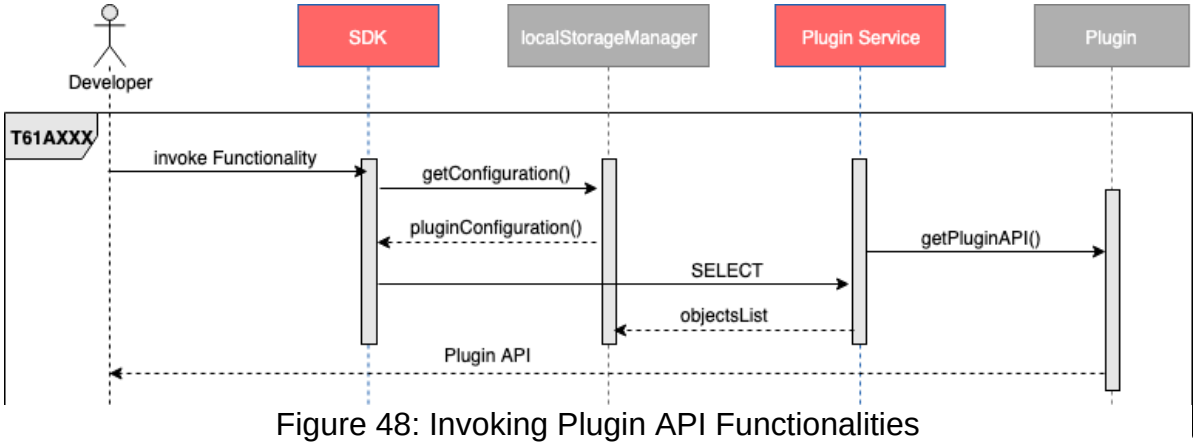


Figure 48: Invoking Plugin API Functionalities

3.4.3.2 Invoke the Service Composition Services

The SDK will expose the API for the developer (or Studio) to compose a ZDMP Application. The supporting applications are available at the ZDMP Platform and the service is invoked through the SDK, as can be seen in the following figure.

The main steps / functionalities are, when invoking the SDK for Composing the Application:

- The SDK retrieves the zApp Configuration
- The SDK invokes the Dependency checker, to see if all needed sources/libraries are available in the ZDMP Repository
- The SDK invokes the API checker, to see if all Interfaces are being compliant
- The SDK invokes an application to define the build manifest with the outcomes of the previous calls
- The SDK returns the Build Manifest to the caller. This is the needed input for making the zApp build in the future

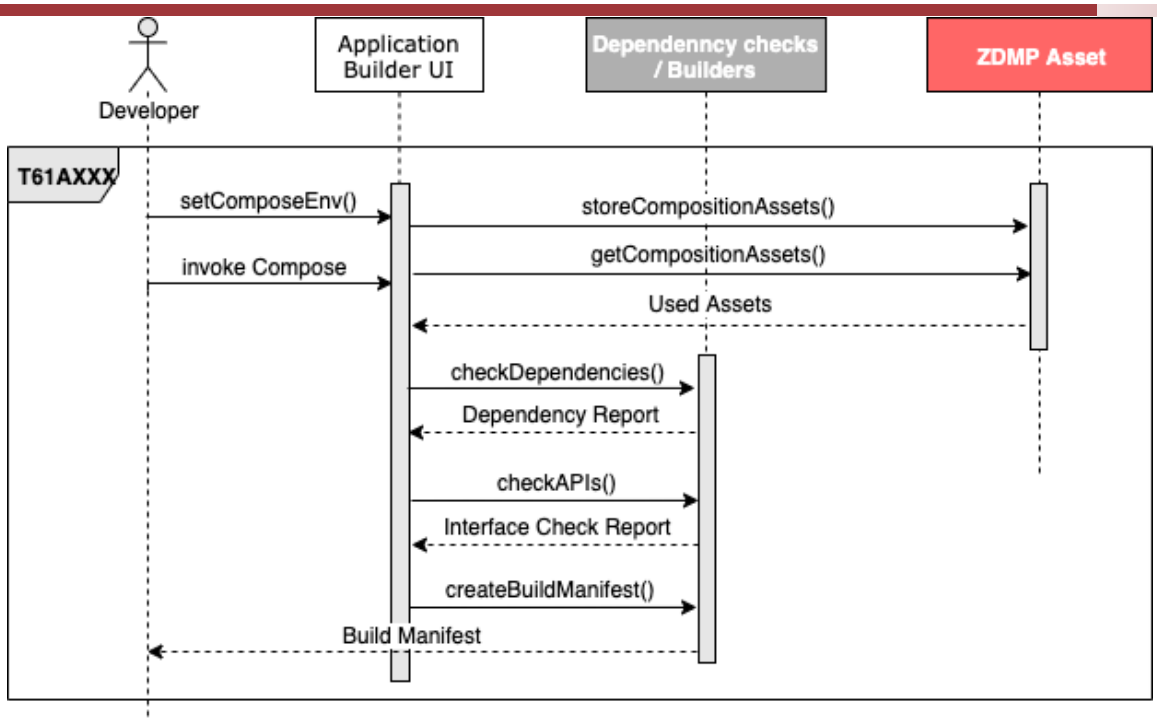


Figure 49: Composing a zApp

3.4.3.3 Invoke the Developer Engagement Hub APIs

The SDK includes the possibility of working with the APIs of the Developer Engagement Hub. No interactions are foreseen here as the SDK will simply expose the Hub's APIs, as seen in the following figure.

The main steps / functionalities are:

- Invoking the SDK for getting the appropriate Engagement Hub API for the current project

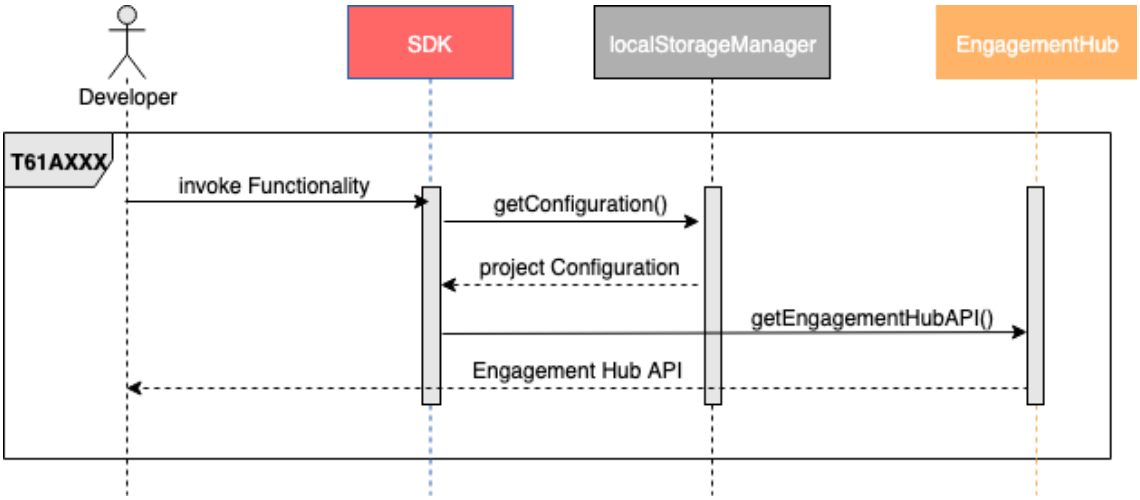


Figure 50: Invoking the Engagement Hub APIs

3.4.3.4 Reset Password

This feature enables users to reset their password in case of a forgotten password. The user will then be provided with a new password via mail by the ZDMP Security component.

The main steps/functionalities are:

- Receive the command to reset the password
- Forward command to the security component, to initialise the password reset process
- Provide a confirmation to the zAsset

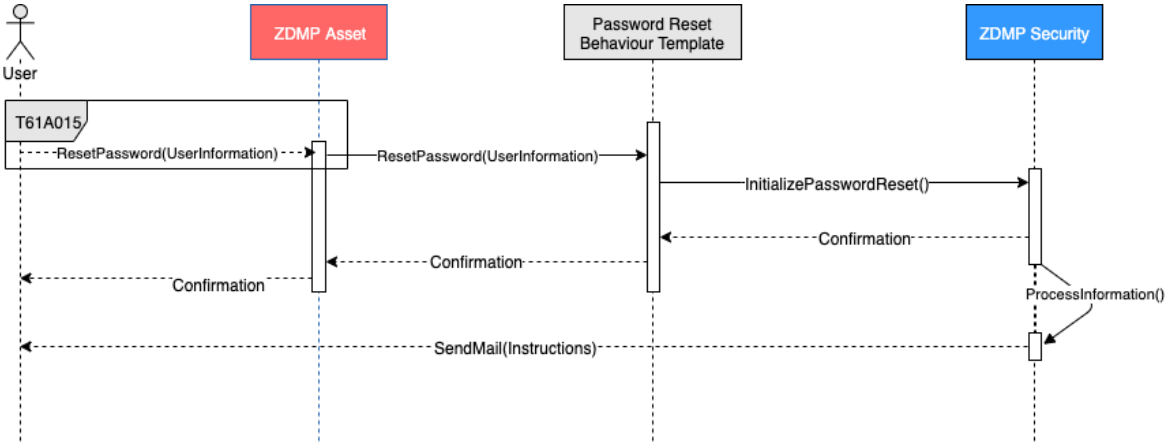


Figure 51: Reset Password Sequence Diagram

3.4.3.5 Add new Language

This feature enables users to add a new language to a zAsset. For this, a tool is provided that shows all strings from a zAsset and the default language (English) that can be translated. After the user has translated the existing strings, they can save it locally and send it to the developer to include the translation in the default version of the zAsset.

The main steps/functionalities are:

- Translate the current strings into a new language
- Save the translation locally and send it to the developer for a permanent language option
- The developer must review the new translation and must make sure that there are no obvious and knowing violations
- The developer must release a latest version of a ZDMP Asset with updated languages

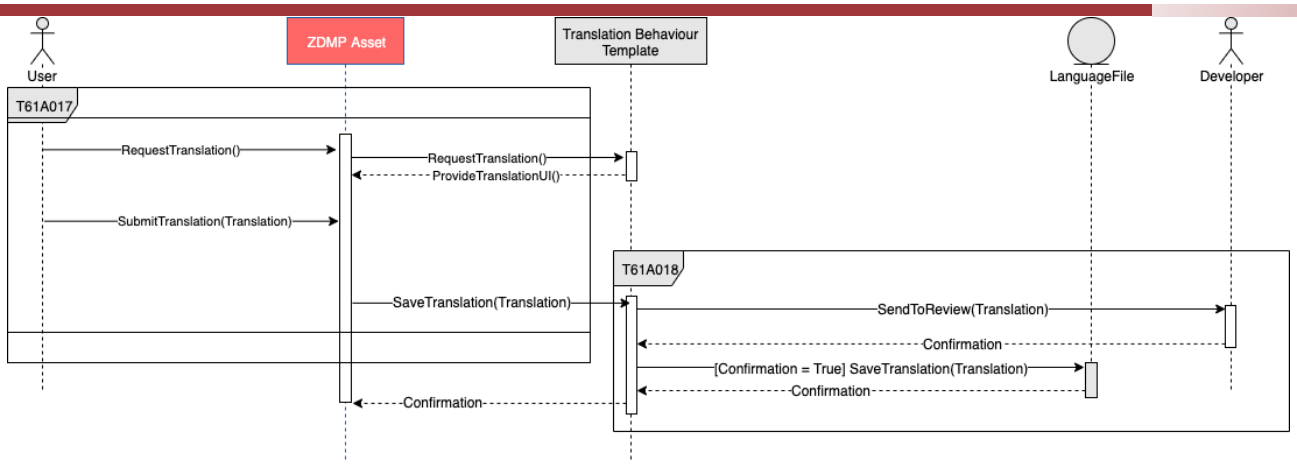


Figure 52: Add new Language Sequence Diagram

3.4.3.6 Switch Language

This feature enables users to switch the language of a zAsset at runtime. The default language is always English, but users can add new languages that also can be used by others.

The main steps/functionalities are

- Choose the language to be used in the zAsset
- Translate text into the targeted language

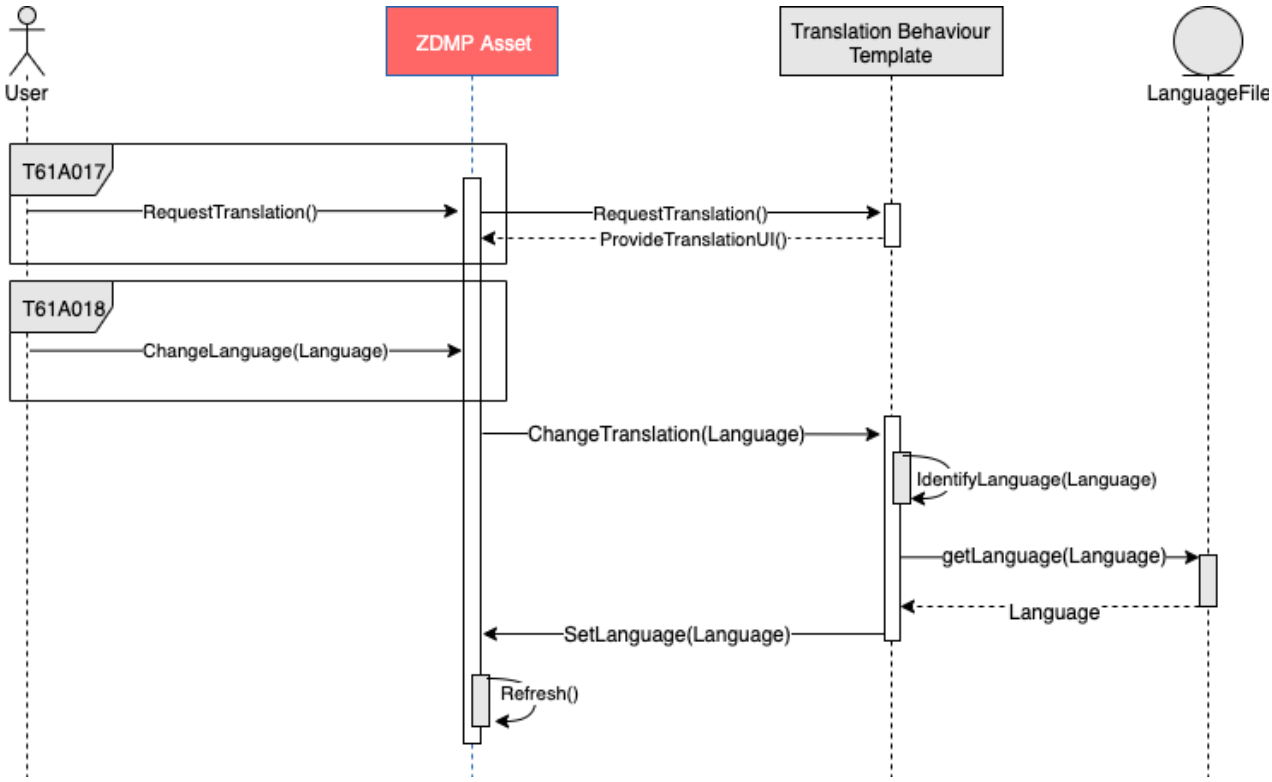


Figure 53: Switch Language Sequence Diagram

3.4.3.7 Manage Notification

This feature enables users to get notifications of important messages or pending interactions. The developer can easily integrate this behaviour via the Notification Template.

The main steps/functionalities are:

- Show the notification on the screen
- Remove the notification from the screen
- Set a reminder for a later appearance

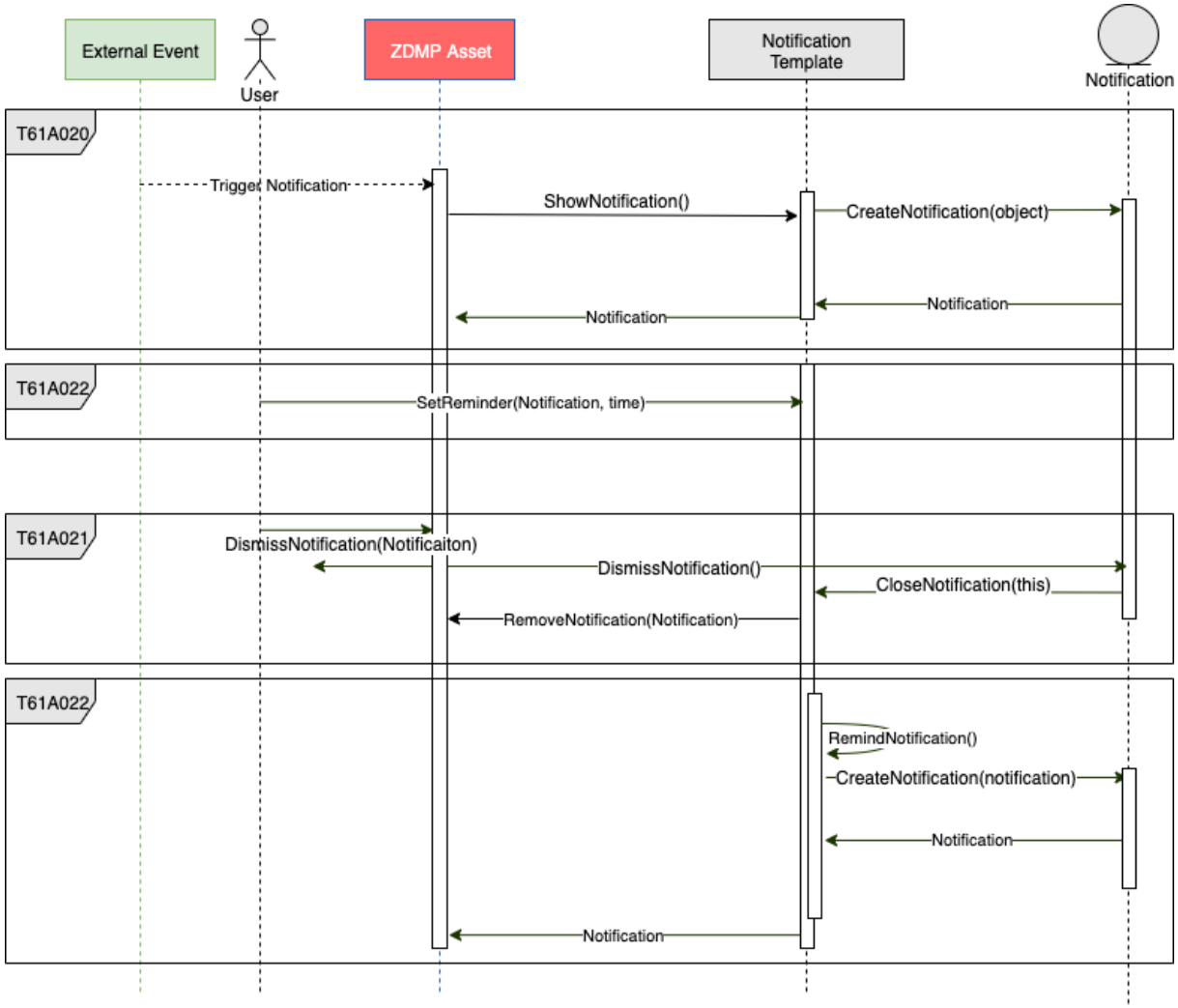


Figure 54: Manage Notification

3.5 SDK API Management (T6.1)

3.5.1 Overall Functional Characterization

The API Management is a JS based backend component that enables the Application Builder to interact with ZDMP assets. In practice, the drag and drop development environment on the web frontend is internally built by interconnected APIs and bindings to ZDMP services. These APIs are provided by other technical components that respond to individual requirements of the platform. This enables to make features functional by calling

external services as data acquisition, AI and analytics, storage, human collaboration, and marketplace.

3.5.2 Functions / Features

This function can be grouped to the following features which have to be implemented:

Subtask	Subtask description
T61B001 Get zApps	**Priority:** Must
	Who: zApps Developer **When**: Design time in the app builder **Where**: In app builder **What:** Lists existing zApps (stored on the ZDMP-Store) getting their names, detail on functionality, version, etc **Why:** Browse existing zApps to reuse them or to understand how to interact with them
Acceptance Criteria	Invoker got a structured list of the zApps stored on ZDMP-Store
Requirements filled	RQ_0258, RQ_0331, RQ_0340, RQ_0370, RQ_0001
T61B002 Get ZDMP Assets	**Priority:** Must
	Who: zApps Developer **When**: In app builder **Where**: Design time in the app builder **What:** Lists existing ZDMP assets (components, services) getting their names, detail on functionality, version, etc **Why:** Browse existing services to reuse them or to understand how to interact with them
Acceptance Criteria	Invoker got a structured list of the ZDMP components and services
Requirements filled	RQ_0258, RQ_0331, RQ_0340, RQ_0370, RQ_0001
T61B003 Get Data Analytics Services	**Priority:** Must
	Who: zApps Developer **When**: Design time in the app builder, run time in zApps and service **Where**: Anywhere **What:** Lists existing Data Analytics services, getting their names, detail on functionality, version, etc **Why:** Browse existing services to understand how to use them
Acceptance Criteria	Invoker got a structured list of the Data Analytics services
Requirements filled	RQ_0161, RQ_0174, RQ_0175, RQ_0194, RQ_0219
T61B004 Get Monitoring Services	**Priority:** Must
	Who: zApps Developer **When**: Design time in the app builder, run time in zApps and service **Where**: Anywhere **What:** Lists existing Monitoring services, getting their names, detail on functionality, version, etc **Why:** Browse existing services to understand how to use them
Acceptance Criteria	Invoker got a structured list of the Monitoring services
Requirements filled	RQ_0161, RQ_0174, RQ_0175, RQ_0194, RQ_0219
T61B005 Get Alerting Services	**Priority:** Must
	Who: zApps Developer **When**: Design time in the app builder, run time in zApps and service **Where**: Anywhere **What:** Lists existing Alerting services, getting their names, detail on functionality, version, etc **Why:** Browse existing services to understand how to use them
Acceptance Criteria	Invoker got a structured list of the Alerting services
Requirements filled	RQ_0161, RQ_0174, RQ_0175, RQ_0194, RQ_0219

T61B006 **Get Data Models**	**Priority:** Must
	Who: zApps Developer **When**: Design time in the app builder, run time in zApps and service **Where**: Anywhere **What:** Lists existing Data Models and stored patterns, getting their names, detail on functionality, version, etc **Why:** Browse existing models and patterns to understand how to use them
Acceptance Criteria	Invoker got a structured list of the stored Data Models and patterns
Requirements filled	RQ_0161, RQ_0219
T61B007 **Get Data Analytics service API**	**Priority:** Must
	Who: zApps Developer **When**: Design time in the app builder, run time in zApps and service **Where**: Anywhere **What:** Retrieves API and manifest files for the selected Data Analytics service **Why:** Make use of APIs and manifests
Acceptance Criteria	Invoker got API and manifest definition and description
Requirements filled	RQ_0161, RQ_0174, RQ_0175, RQ_0194, RQ_0219
T61B008 **Get Monitoring service API**	**Priority:** Must
	Who: zApps Developer **When**: Design time in the app builder, run time in zApps and service **Where**: Anywhere **What:** Retrieves API and manifest files for the selected Monitoring service **Why:** Make use of APIs and manifests
Acceptance Criteria	Invoker got API and manifest definition and description
Requirements filled	RQ_0161, RQ_0174, RQ_0175, RQ_0194, RQ_0219
T61B009 **Get Alerting service API**	**Priority:** Must
	Who: zApps Developer **When**: Design time in the app builder, run time in zApps and service **Where**: Anywhere **What:** Retrieves API and manifest files for the selected Alerting service **Why:** Make use of APIs and manifests
Acceptance Criteria	Invoker got API and manifest definition and description
Requirements filled	RQ_0161, RQ_0194, RQ_0219
T61B010 **Get Data Model definitions**	**Priority:** Must
	Who: zApps Developer **When**: Design time in the app builder, run time in zApps and service **Where**: Anywhere **What:** Retrieves APIs and manifest files for the selected data model / design pattern **Why:** Make use of APIs and manifests
Acceptance Criteria	Invoker got API and manifest definition and description
Requirements filled	RQ_0161, RQ_0219
T61B011 **Retrieve Composition data**	**Priority:** Must
	Who: zApps Developer **When**: Design time in the app builder, run time in zApps and service **Where**: Anywhere **What:** Submits a set of APIs and Manifests describing the services, and particularly a manifest that describes how the services should be composed **Why:** To provide all information needed for the zApp build
Acceptance Criteria	zApp Composer received the set of APIs and Manifests corresponding to the zApp being developed
Requirements filled	RQ_0161, RQ_0219, RQ_0247
T61B012 **Validate APIs**	**Priority:** Must
	Who: zApp Composer

	When: Design time in the app builder **Where**: In app builder **What:** Analyses interfaces and API specifications and ensures that the invocation data complies to that specification **Why:** To ensure that the defined APIs are being respected
Acceptance Criteria	All invocations of internal and external interfaces comply with the APIs specifications
Requirements filled	RQ_0247
T61B013 Store Configuration	**Priority:** Must
	Who: zApps Developer **When**: Design time in the app builder **Where**: Anywhere **What:** Stores the zApp project configuration file to the Data Storage **Why:** Store the new zApp configuration to the Data Storage, as well as the configuration of the SDK and Studio that were used for building it
Acceptance Criteria	Invoker got response about storing the configurations in the Data Storage
Requirements filled	RQ_0111, RQ_0161, RQ_0194, RQ_0219, RQ_0319
T61B016 Post User Credentials	**Priority:** Must
	Who: Application Builder and Security **When**: Design time in the app builder, runtime in zApps and services **Where**: In app builder and zApps **What**: The Application Builder makes a call to the security component with the entered user credentials in order to authorise the user and gets the user permissions **Why**: To provide the security component with valid user information in order to get a response
Acceptance Criteria	The Application Builder should get a response from the security component to process the response
Requirements filled	RQ_0161, RQ_0219
T61B017 Process User Credentials	**Priority:** Must
	Who: Application Builder **When**: Design time in the app builder, runtime in zApps and services **Where**: In app builder and zApps **What**: Receives the response from the Security component and evaluates it **Why**: To continue the workflow with permitted access
Acceptance Criteria	The response of the security should contain one of the following: Successful user login including authorised permission to act Successful user login including unauthorised permission to act Invalid User Credentials
Requirements filled	RQ_0161, RQ_0219
T61B018 Call Marketplace	**Priority:** Must
	Who: SDK **When**: Design time in the app builder, runtime in zApps and services **Where**: In app builder and zApps **What:** Call the ZDMP-Platform to execute a process **Why:** So that a process is effectively executed
Acceptance Criteria	The invoke call is successfully relayed to the ZDMP-Platform
Requirements filled	RQ_0258, RQ_0331, RQ_0340, RQ_0370, RQ_0001, RQ_0225
T61B019 Receive data from Platform	**Priority:** Must
	Who: SDK **When**: Design time in the app builder, runtime in zApps and services **Where**: Anywhere **What:** Data processed is sent back **Why:** So that a process can relay the data received to the calling zApp/ZDMP-OS Asset

Acceptance Criteria	The processed data is received by the Process Instance
Requirements filled	RQ_0161, RQ_0219, RQ_0258, RQ_0331, RQ_0340, RQ_0370, RQ_0001
T61B020 Call External Service Provision	**Priority:** Must
	Who: SDK **When**: Design time in the app builder, runtime in zApps and services **Where**: Anywhere **What:** Call the external service provision **Why:** So that a process is effectively helped by the execution of 3rd party services
Acceptance Criteria	The invoke call is successfully relayed to the ESP
Requirements filled	RQ_0161, RQ_0174, RQ_0175, RQ_0194, RQ_0219
T61B021 Receive data from External Service Provision	**Priority:** Must
	Who: SDK **When**: Design time in the app builder, runtime in zApps and services **Where**: Anywhere **What:** Data processed is sent back **Why:** So that a process can relay the data received to the calling z-App/ZDMP-OS Asset
Acceptance Criteria	The processed data is received by the Process Instance
Requirements filled	RQ_0161, RQ_0219
T61B022 Send usage data	**Priority:** Must
	Who: SDK **When**: Design time in the app builder, runtime in zApps and services **Where**: Anywhere **What:** Send usage data **Why:** So that the Platform can inform the System Dashboard (T6.4 / T6.5)
Acceptance Criteria	The usage data is successfully relayed to the Platform
Requirements filled	NONE
T61B023 RESTful API to messaging using HTTP	**Priority:** Must
	Who: ZDMP-OS Assets **When**: Design time in the app builder, runtime in zApps and services to request services from other APIs and services **Where**: Anywhere **What:** Provide interface for sending messages using HTTP. **Why:** To enable APPs to use messaging component to send messages by using HTTP through standardised and understandable interface.
Acceptance Criteria	zApps can send and receive message through the exposed RESTful interface
Requirements filled	RQ_0161, RQ_0194, RQ_0219

Figure 55: SDK-/API Management Features

3.5.3 Workflows

This section is highlighting the user interaction and the interaction of the component in API management.

3.5.3.1 Browse zApps and APIs

The SDK allows a developer (via API), or any other client service (eg Studio) to retrieve the set of zApps from the ZDMP Marketplace, as can be seen in the following figure.

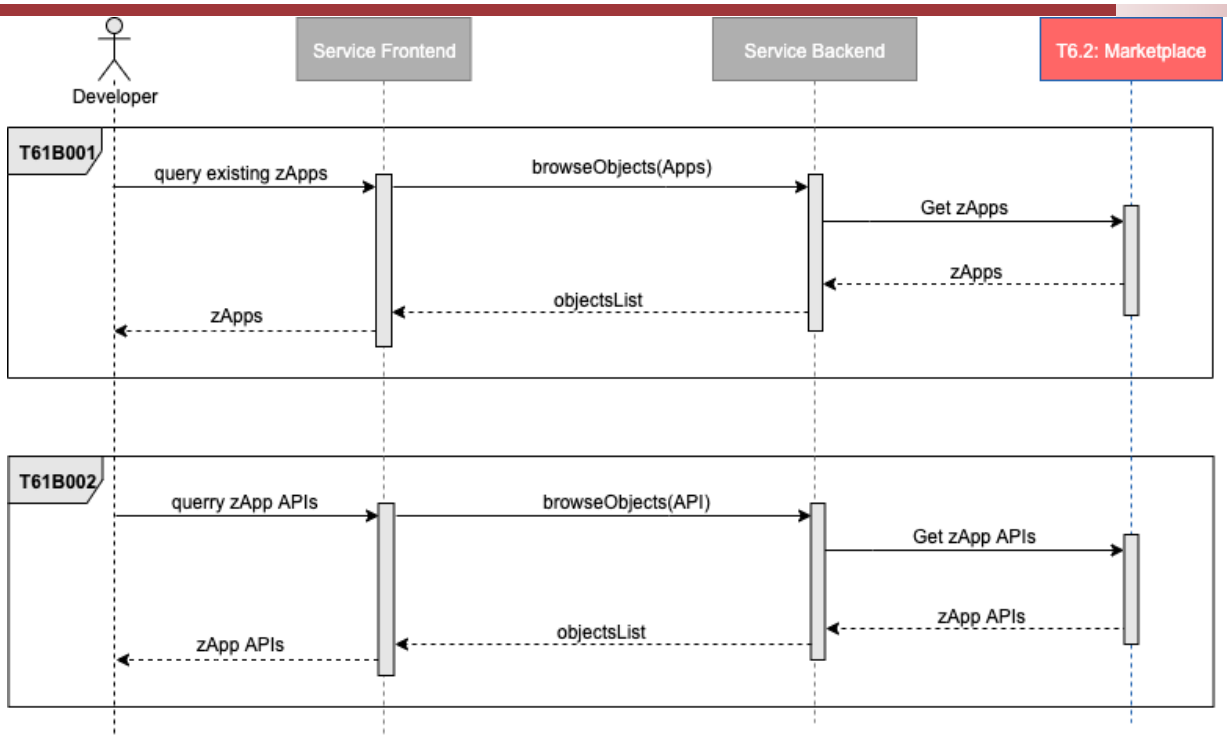


Figure 56: Retrieve zApps from Marketplace

3.5.3.2 Retrieve Assets from Data Storage

As with to the zApps retrieval, the multiple assets that are available on ZDMP can be retrieved from the Data Storage, using the flow depicted on the following figure.

The main steps / functionalities are:

- Query the existing zAssets (services for zApps) and select a set of zAssets
- Apply a desired filter and select the desired zAsset

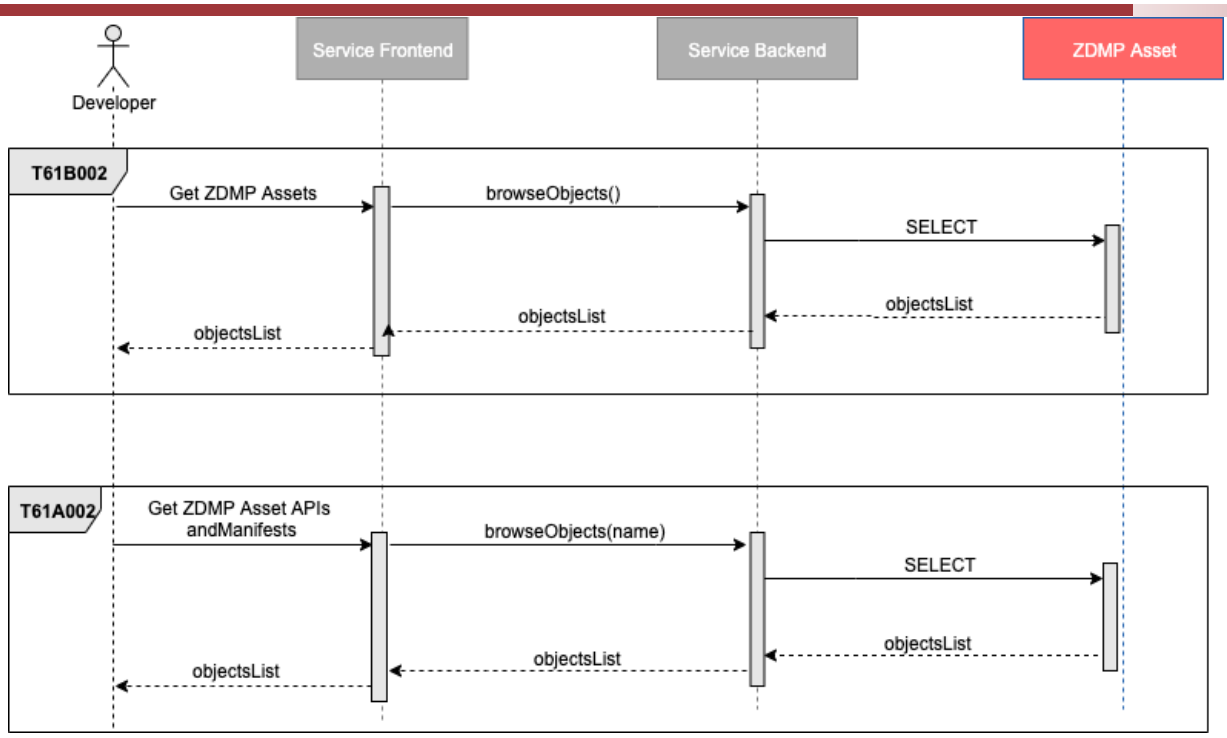


Figure 57: Retrieve ZDMP assets from the Data Storage

3.6 SDK Development Tools (T6.1)

3.6.1 Overall Functional Characterization

Application Builder provides several tools to zApps developers to support functions like editing, compiling, debugging, syntax highlight and automation of source code. To allow subgrouping of source code files, a dependency check tool is made available. After the app is developed, it can be built to a mobile app, web app or desktop app using Builder. The Builder use Ionic framework to build apps for mobile (iOS / Android) and Electron to build a desktop app.

A service validator is used to check if the connections between services made by the developer through the graphical interface are feasible to implement with the ZDMP APIs.

3.6.2 Functions / Features

There function can be grouped to the following features which have to be implemented:

Subtask	Subtask description
T61C001 Validate APIs	**Priority:** Must
	Who: 3rd party developer via zApp Composer **When:** Design time in the app builder **Where**: Can be anywhere, where access to the platform and the necessary services are given **What:** Analyses interfaces and API specifications and ensures that the invocation data complies to that specification **Why:** To ensure that the defined APIs are being respected
Acceptance Criteria	All invocations of internal and external interfaces comply with the APIs specifications

Requirements filled	RQ_0161, RQ_0247
T61C002 Build	**Priority:** Must
	Who: 3rd party developer via zApps Developer **When**: Design time in the app builder **Where**: In app builder **What:** Submits a set of APIs and Manifests describing the new zApp, together with the modules of code for building, and the response should be whether the build was successful or not. **Why:** To validate the build of the new zApp
Acceptance Criteria	Invoker got response regarding the Build invoking after submitting a set of information
Requirements filled	RQ_0091, RQ_0161, RQ_0350, RQ_0462, RQ_0474, RQ_0492
T61C003 Deploy	**Priority:** Must
	Who: zApps Developer **When**: Design time in the app builder **Where**: In app builder **What:** Submits a set of APIs and Manifests describing the new zApp, and its code/built executable **Why:** store the new zApp in the ZDMP-Store
Acceptance Criteria	Invoker got response about deploying the new zApp in the ZDMP-Store
Requirements filled	RQ_0091, RQ_0161, RQ_0350, RQ_0462, RQ_0474, RQ_0492
T61C004 Edit	**Priority:** Must
	Who: Application Builder, Asset **When**: Design time in the app builder, In zApps **Where:** In app builder **What:** Uncaught exceptions are intercepted and afterwards transformed in a readable format **Why:** To increase readability of error reports
Acceptance Criteria	The result of the creation should be a valid report model, which can be parsed into a valid JSON format
Requirements filled	RQ_0091, RQ_0161, RQ_0350, RQ_0462, RQ_0474, RQ_0492
T61C005 Compile	**Priority:** Must
	Who: Application Builder, Secure Business Cloud **When**: Design time in the app builder **Where**: In app builder **What:** Errors are submitted to the Secure Business Cloud Backend to inform the developer about such errors **Why:** To increase the quality of ZDMP-Apps in the Secure Business Cloud
Acceptance Criteria	An error report is sent to the Secure Business Cloud backend and can be viewed there
Requirements filled	RQ_0091, RQ_0161, RQ_0350, RQ_0462, RQ_0474, RQ_0492
T61C006 Debug	**Priority:** Must
	Who: Application Builder, Secure Business Cloud **When**: Design time in the app builder **Where**: In app builder **What:** Errors are submitted to the Secure Business Cloud Backend containing the email address of the user as a contact person in order to inform the developer about such errors **Why:** To increase the quality of ZDMP-Apps in the Secure Business Cloud
Acceptance Criteria	An error report is sent to the Secure Business Cloud backend and can be viewed there including the contact information (email)
Requirements filled	RQ_0091, RQ_0161, RQ_0350, RQ_0462, RQ_0474, RQ_0492
T61C007 Syntax Highlighting	**Priority:** Must
	Who: Application Builder, Secure Business Cloud **When**: Design time in the app builder

	Where: In app builder **What:** Errors are forwarded to the Secure Business Cloud Backend containing the email address of the user as a contact person in order to inform the developer about such errors **Why:** To increase the quality of ZDMP-Apps in the Secure Business Cloud
Acceptance Criteria	An error report is sent to the Secure Business Cloud backend and can be viewed there including the contact information (email)
Requirements filled	RQ_0091, RQ_0161, RQ_0350, RQ_0462, RQ_0474, RQ_0492
T61C008 Automation	**Priority:** Must
	Who: Application Builder **When**: Design time in the app builder **Where**: Design time in the app builder **What:** Provides a form to register a new user in ZDMP **Why:** To enable a common behaviour of creating new user accounts
Acceptance Criteria	A user must be created and able to login afterwards
Requirements filled	RQ_0091, RQ_0161, RQ_0258, Q_0318, RQ_0331, RQ_0340, RQ_0370, RQ_0001
T61C009 Dependency checks	**Priority:** Must
	Who: Application Builder **When**: Design time in the app builder **Where**: In app builder **What**: Shows a popup of a User Login Form if an action needs to be authorised, and the user is not logged in currently **Why**: To give a common login mask including a behaviour template for "authorised only" actions
Acceptance Criteria	Only authorised users shall be able to act for specific actions with limited access
Requirements filled	RQ_0111, RQ_0161, RQ_0247, RQ_0350, RQ_0462, RQ_0474, RQ_0492
T61C010 Parsers	**Priority:** Must
	Who: Application Builder and Security **When**: Design time in the app builder **Where**: In app builder **What**: The Application Builder makes a call to the security component with the entered user credentials in order to authorise the user and gets the user permissions **Why**: To provide the security component with valid user information in order to get a response
Acceptance Criteria	The Application Builder should get a response from the security component to process the response
Requirements filled	RQ_0091, RQ_0161
T61C011 Service Validators	**Priority:** Must
	Who: Application Builder **When**: Design time in the app builder **Where**: In app builder **What**: Receives the response from the Security component and evaluates it **Why**: To continue the workflow with permitted access
Acceptance Criteria	The response of the security should contain one of the following: Successful user login including authorised permission to act Successful user login including unauthorised permission to act Invalid User Credentials
Requirements filled	RQ_0161, RQ_0194, RQ_0247

Figure 58: SDK Development Tools Functions

3.6.3 Workflows

This section is highlighting the user interaction and the interaction of the component in the single functions.

3.6.3.1 Build zApp

The SDK exposes the API for the developer (or Studio) to build a ZDMP Application. The builders themselves are available at the ZDMP Platform, and the service is invoked through the SDK, as can be seen in the following figure.

The main steps / functionalities are:

- Invoking the SDK for Building the Application
- The SDK retrieves and configures the appropriate Builder on the ZDMP Platform
- The SDK invokes the Build process
- The SDK returns to the caller the building report

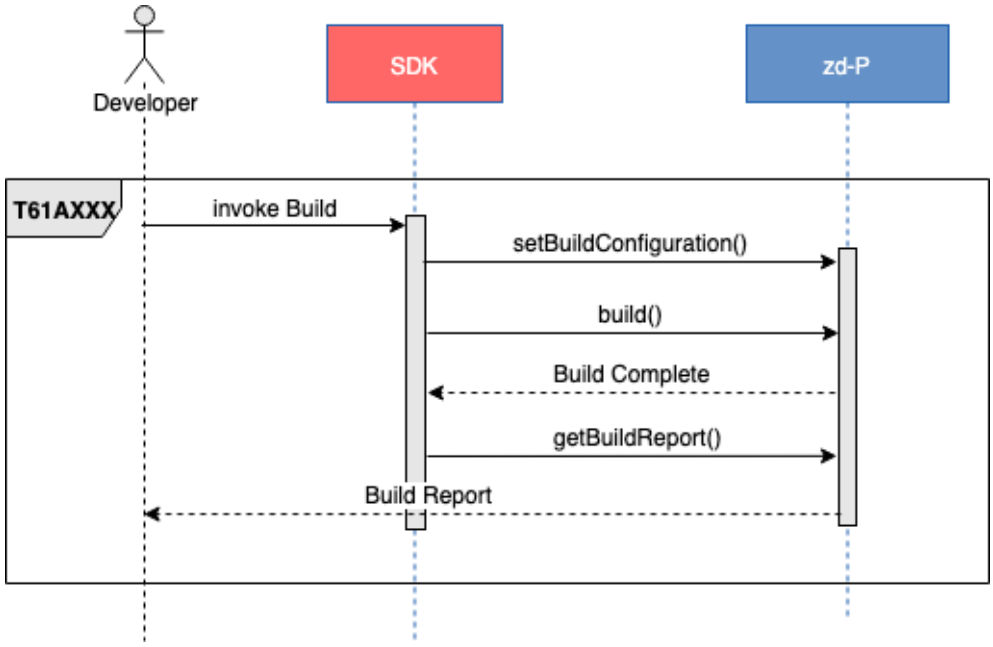


Figure 59: Building a zApp

3.6.3.2 Deploy a zApp

The SDK exposes the API for the developer (or Studio) to deploy a ZDMP Application. The creation of the deployment instance container itself is available at the ZDMP Platform, and the service is invoked through the SDK, as can be seen in the following figure.

The main steps / functionalities are:

- Invoking the SDK for Deploying the Application
- The SDK configures the deployment environment on the ZDMP Platform
- The SDK invokes the Deploy process
- The SDK returns to the caller an instance of the deployed zApp

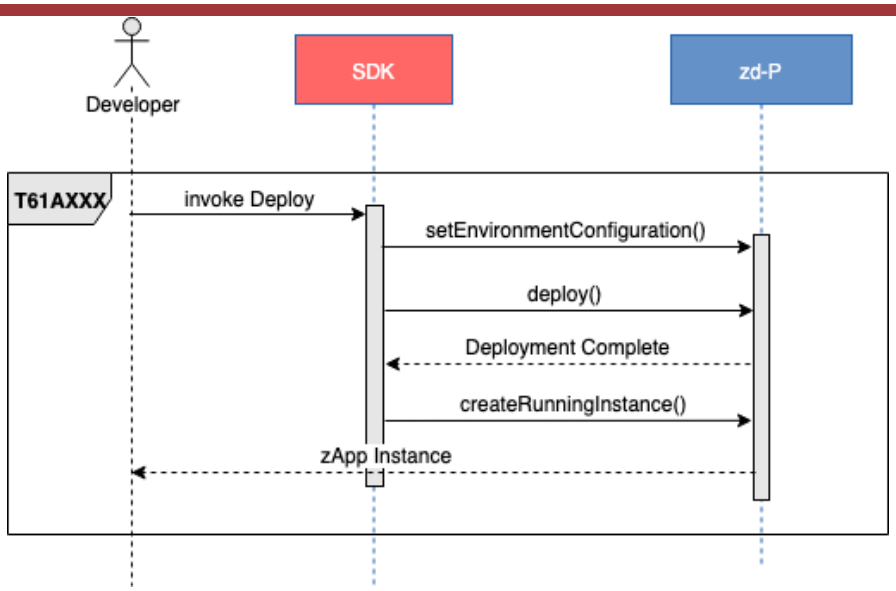


Figure 60: Deploying a zApp

3.6.3.3 Report Errors

This feature sends error reports to the developer, so that the developer can react quickly to troubleshoot the application. There, the developer gets information about the user, the used system, and the error message itself. For this feature, an internet connection is necessary, because of a direct communication with the Marketplace.

The main steps/functionalities are:

- Intercept exceptions during runtime of zAssets
- Create an error report with all retrievable information (Sender, Operating System, Error Message, and time)
- Send the error report to the Marketplace, where the developer gets insight in the administration view

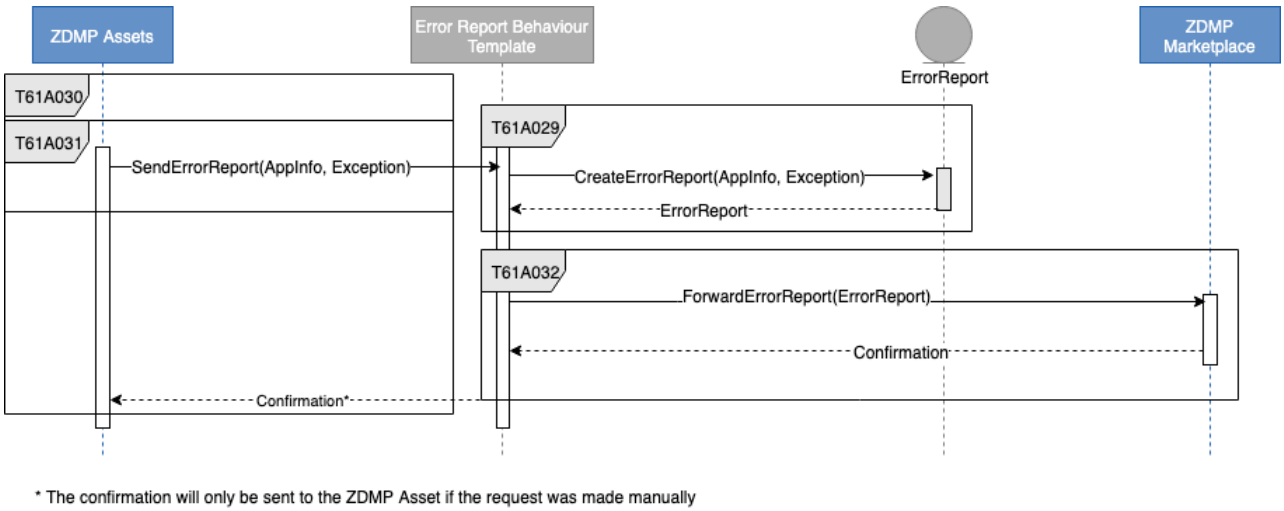


Figure 61: Report Errors Sequence Diagram

3.6.3.4 Register User

This feature enables new users to register to ZDMP, which then enables them to log in and use its functionalities. For this, the user only must enter their first name, last name, and email address. It is also required to accept ZDMP Terms of Service and Privacy Policy.

The main steps/functionalities are:

- Validate and submit user information
- Forward information to the security component, to get a confirmation about the registration process

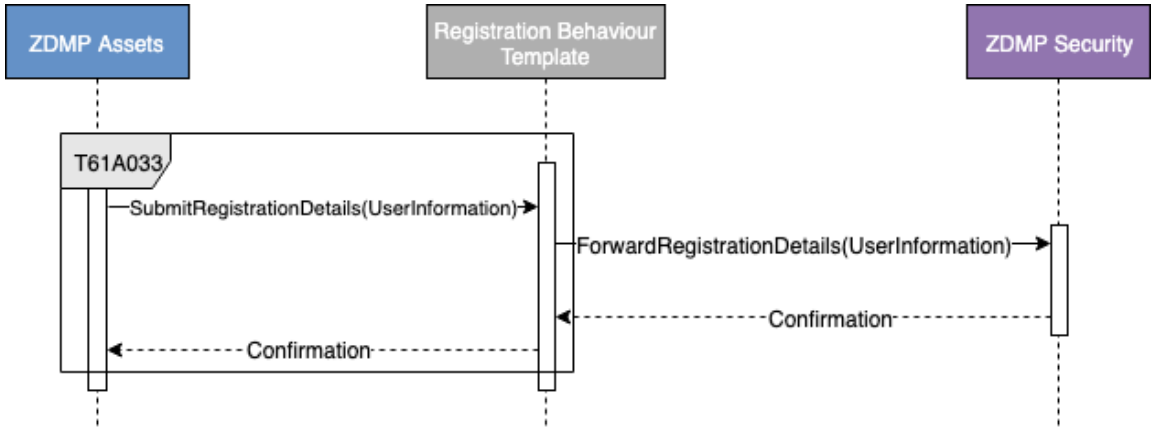


Figure 62: Register User Sequence Diagram

3.6.3.5 Login User

This feature enables new users to login into ZDMP. After that, the users can use all ZDMP functionalities within the scope of their authorisation. For the login, the user only needs to enter its credentials (username and password).

The main steps/functionalities are:

- Receive user credentials, in which the password is of course encrypted
- Forward credentials to the security component, to get session information
- Provide session information to the zAsset

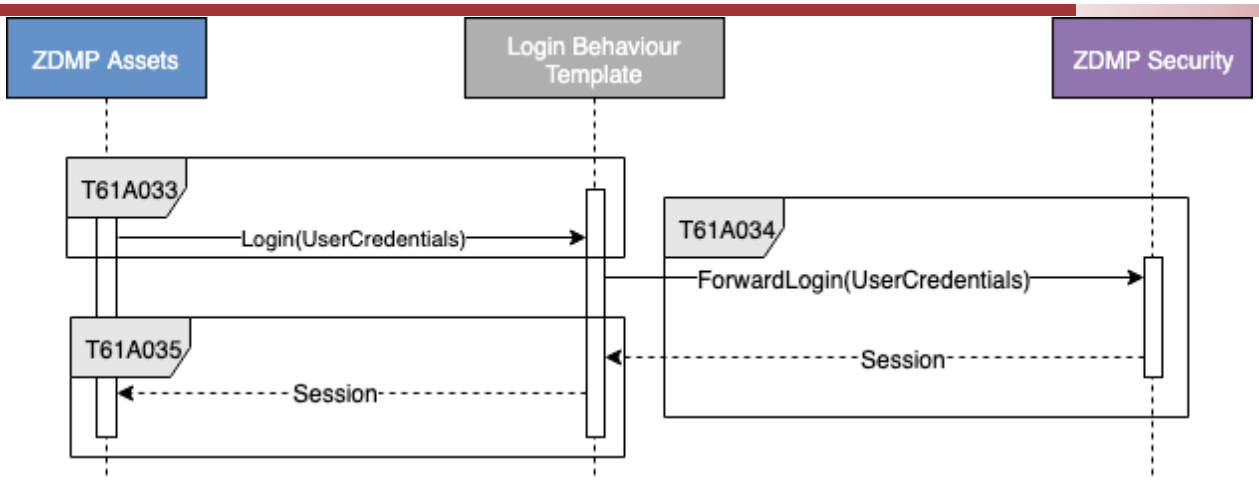


Figure 63: Login User Sequence Diagram

3.7 Security Designer (T6.2)

3.7.1 Overall functional characterization & Context

Purpose: The Security Designer (SD) enables automated and systematic identification of risks to the assets (both human and technological) contained or connected to the ZDMP platform. By assets it can mean: Computers, processes, networks, communication links, users etc. The Security Designer also allows identifying the knock-on consequences and countermeasures to mitigate these risks. The tool allows collaboration between several stakeholders to develop the system model and the associated risk catalogue.

Description: The Security Designer provides the ZDMP infrastructure and process design validation by inferring the missing assets. The primary and secondary threats are automatically generated for each asset, along with the corresponding control strategies, helping users understand what risk management measures are needed in their system. Based on the threats and controls encoded in the knowledge base, SD identifies the potential weaknesses of the model and suggests controls for addressing the nascent threats. This helps users to understand what countermeasures are required. Security modelling is an iterative process which consists of the following main steps:

- The user constructs a system model by placing assets on the modelling canvas and creates links (relations) between the assets. The user-defined entities are called “asserted assets” and “asserted relations”
- The next step is the validation which determines whether the information provided about the assets and relationships is consistent and complete. If the validation fails (ie the model gets marked as ‘invalid’) then the user updates the model and validates it again. During validation, the inferred assets/relations, threats, and security controls (the counteract the threats) are automatically generated
- The final step is the threat management, during which the user addresses individual threats by selecting controls for the assets or control strategies for the threats. The aim is to eliminate or at least mitigate all threats

The main components of SD and the association with other ZDMP components are described in detail in D4.3a: Global Architecture Specification.

3.7.2 Functions / Features

This section describes the primary features and functions of Security Designer:

- **Login and registration:** The login page is activated either by clicking on ‘Sign In’ link in the dropdown menu or by clicking on the Login button. The user must enter their username and password. If the user is new to the system, they must first register a new user account. If they have previously received an invitation email from the Security Designer Admin, they simply click the link in that email, which opens the registration page. Otherwise, the user manually clicks the Register button on the welcome page, or the Register menu option.
- **Model management:** This feature provides various functionalities such as: listing, creating, importing/exporting models and deleting models. After a successful login, the user is presented with a list of their models that were previously created. In the model details, the Domain used by the model is labelled. There is also a description

of the model, which can be edited via the Edit Details function. At the bottom of the model panel, there are several icons that reflect the status of the model.

- The Create New Model function opens a design palette and allows the user to choose which domain model to use. The import function allows the user to upload a previously saved file into a new model. The user can also import asserted facts only (eg asserted assets, relations). If restoring a previous version of a model, the user can check 'Overwrite existing model'. Attempting to import the same model without this being checked will result in an error. A user can re-import an existing model with a different name by checking New Name. Once the model was constructed, it can be exported into a file. Using the export function, the model is saved into the user's "downloads" directory. The delete action removes the model along with any associated data items (eg assets, relationships).
- **Model construction:** This is iterative process that involves: selecting and adding assets to the construction canvas, adding relationships between assets, deleting assets/relations, and renaming assets. An asset can be added to the construction canvas by selecting an icon in the Asset Palette and dragging it onto the canvas. The Asset Palette contains various assets; these fall into four main categories: Hosted Asset, Network Asset, Space, and Stakeholder. Once the assets have been put onto the modelling canvas, the user can connect pairs of assets by establishing links between them. Assets and links can be deleted. By deleting an asset all incoming and outgoing links will also get deleted. The user can rename an existing asset by editing the asset's name under the corresponding icon. By changing the name, the asset's connections will stay unaffected. All asset names must be unique.
- **Obtaining information from ZDMP components:** This information is used by security expert for constructing and refining the security model. This information may include the following: Security controls of components and zApps, logical/physical connections between components, deployment of components, authentication/authorization of users, security settings of network topology. For a detailed description see D4.3a: Global Architecture Specification.
- **Model validation:** Each newly constructed model must be validated. The validation function runs semantic reasoning that generates inferred assets and relations that are added to the model automatically and produces a list of threats and misbehaviour sets associated with the given model. This operation guarantees that the inferred assets are consistent with the asserted assets and relationships.
- **Risk calculations:** After validating the model that includes potential threats and asset misbehaviours, it is possible to calculate risk levels for them. The risk levels depend on the Trustworthiness Attributes for assets (which determine the likelihood of threats), and the Impact level settings for misbehaviours. The validated model will have default values for these which are determined from the types of Asset and the type of Trustworthiness Attribute or misbehaviour at each Asset. In most cases the defaults will be acceptable, but in most system models a few of the default settings will need to be overridden by user input, indicating which assets are likely entry points for attackers, and which asset misbehaviours would cause serious problems for operation of the system.
- **Threat management:** Threats may be resolved by selecting one or more controls within the Control sets panel. Each control set represents a control on this asset. The Threat Explorer panel displays detailed information about individual threats, including a description along with its likelihood and risk.

- **Reporting:** This function generates a detailed description on both asserted and inferred assets. The information includes the following fields: name of asset, type, trustworthiness, control sets and misbehaviours.

Details of individual functions are described in the following table.

Subtask	Subtask description
T62A001 Provide User Login Form	**Priority:** Must
	Who: T6.2 Security Designer **Where:** Anywhere **When:** Design time **What:** Present the user the login form **Why:** To enable the user to access models and to create new ones.
Acceptance Criteria	Provide access to the user.
Requirements filled	RQ_0111, RQ_0161, RQ_0247, RQ_0350, RQ_0462, RQ_0474, RQ_0492
T62A002 Provide Registration form for new users	**Priority:** Must
	Who: T6.2 Security Designer **Where:** Anywhere **When:** Design time **What:** Present the user a registration form. **Why:** Enable registration of new users
Acceptance Criteria	Successful registration, a confirmation email is sent to the user for verification.
Requirements filled	*RQ_0039, RQ_0073, RQ_0079, RQ_0081, RQ_0113, RQ_0241, RQ_0286, RQ_0423, RQ_0424, RQ_0425, RQ_0426, RQ_0427, RQ_0428, RQ_0429, RQ_0430, RQ_0431, RQ_0432, RQ_0433*
T62A003 Create New Model	**Priority:** Must
	Who: T6.2 Security Designer **Where:** Anywhere **When:** Design time **What:** Open a model construction palette **Why:** Enable to create and edit a system security model
Acceptance Criteria	The user gets access to the model design palette and can select from assets required for constructing a security model
Requirements filled	Requirements to be identified in the security modelling use case
T62A004 List models	**Priority:** Must
	Who: T6.2 Security Designer **Where:** Anywhere **When:** Design time **What:** List all models previously created by the user **Why:** For revising, updating, and deleting models
Acceptance Criteria	The selected model can be edited or removed
Requirements filled	Requirements to be identified in the security modelling use case
T62A005 Import Model	**Priority:** Must
	Who: T6.2 Security Designer **Where:** Anywhere **When:** Design time **What:** Importing model file **Why:** Reading previously stored model file
Acceptance Criteria	The model file is successfully uploaded, and the model can be edited
Requirements filled	Requirements to be identified in the security modelling use case
T62A006 Export Model	**Priority:** Must
	Who: T6.2 Security Designer **Where:** Anywhere **When:** Design time **What**: Exporting model file **Why**: Saving model file in a specific format
Acceptance Criteria	Model file saved

Requirements filled	Requirements to be identified in the security modelling use case
T62A007 Validate Security Model	**Priority:** Must
	Who: T6.2 Security Designer **Where:** Anywhere **When:** Design time **What**: All assets and connections of the security model are checked, and the missing assets inserted **Why**: Ensuring that the model is correct and consistent with the information stored in the ontology. The validation also generates a list of threats and controls for each asset
Acceptance Criteria	Producing a correct (valid) security model, generating threats and controls
Requirements filled	Requirements to be identified in the security modelling use case
T62A008 Generate Security Report	**Priority:** Must
	Who: T6.2 Security Designer **Where:** Anywhere **When:** Design time **What**: Produce a summary security report **Why**: Overview of the system, listing the untreated threats, controls, and risks for each asset
Acceptance Criteria	Full list of assets with all security attributes in the right format
Requirements filled	Requirements to be identified in the security modelling use case
T62A009 Risk calculations	**Priority:** Must
	Who: T6.2 Security Designer **Where:** Anywhere **When:** Design time **What**: Calculate the level of risk for individual threats. The risk characterises the likelihood and impact of a threat **Why**: Each security control is supposed to reduce the level of risk. By this calculation we can check how efficient the given control was
Acceptance Criteria	Correct risk calculation and ranking the threats according to the risk level
Requirements filled	Requirements to be identified in the security modelling use case
T62A010 Obtaining security related information from ZDMP components	**Priority:** Must
	Who: T6.2 Security Designer **Where:** Anywhere **When:** Design time **What**: Obtain security information from other ZDMP components **Why**: The security information will be used by the security expert for designing the model.
Acceptance Criteria	Information from ZDMP components that can be used for refining the security model
Requirements filled	RQ_0258, RQ_0331, RQ_0340, RQ_0370, RQ_0001, RQ_0161, RQ_0219, RQ_0247, RQ_0331, RQ_0340, RQ_0370, RQ_0001, RQ_0225

Figure 64: Security Modeller Functions

3.7.3 Workflows

In the following sections two workflows will be detailed. The first collects information from ZDMP components, zApps, physical and logical connections between the components, network typology, and users' security authorisation/authentication. The second workflow describes the main steps required for model construction, validation and addressing threats.

3.7.3.1 Collecting information from ZDMP components

The traditional use of the Security Designer relies only on the expert knowledge of a security analyst. In ZDMP additional information is intended to be used that can be obtained from other ZDMP components. This information is then used by the security analyst for model

construction. The sequence diagram representing the interaction of SD with ZDMP components that supply information is presented in the following figure.

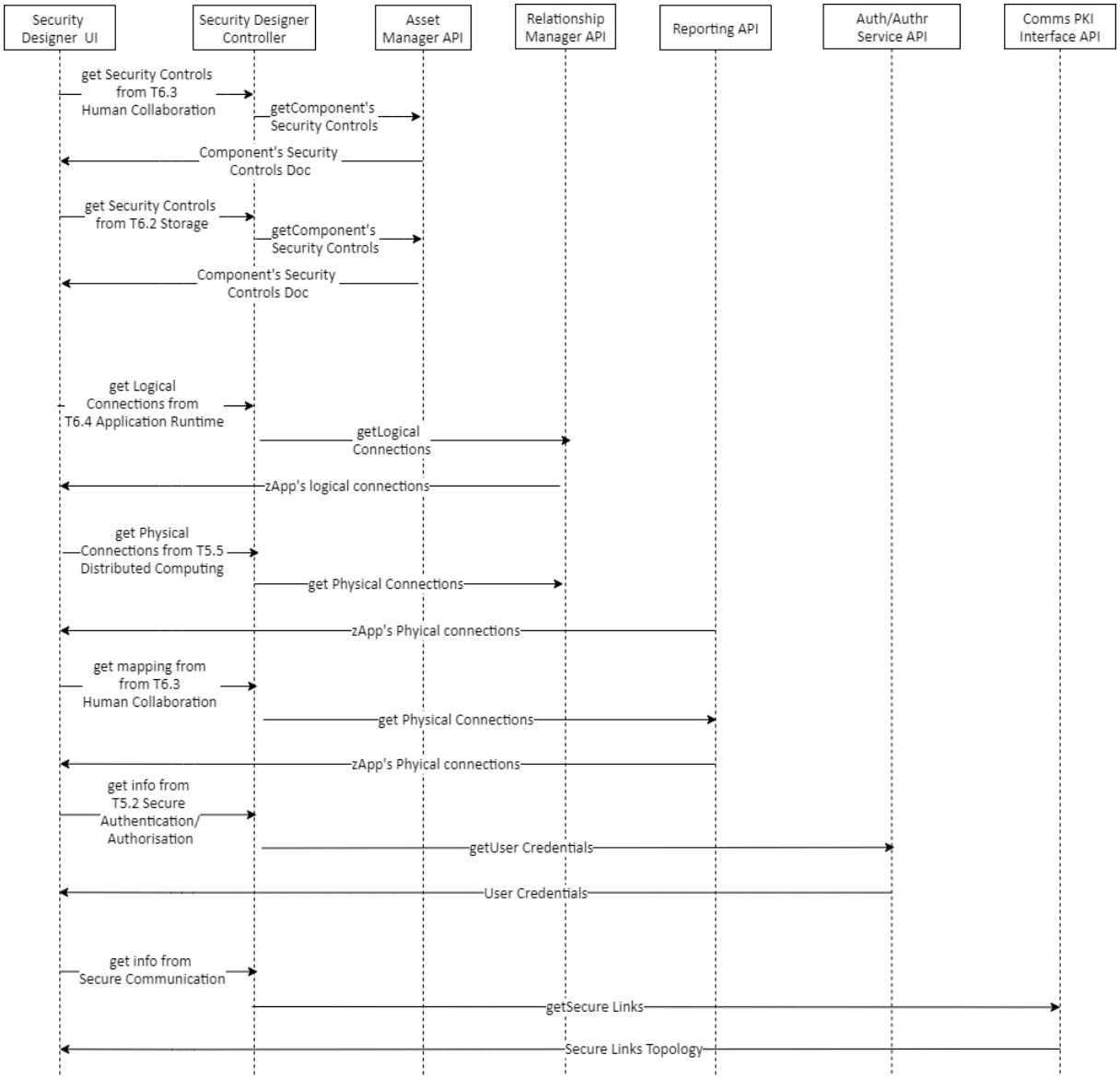


Figure 65: Collecting information from ZDMP components

3.7.3.2 Modelling process

The user interacts with the SD via the Security Designer UI running in a web browser. The user, after successful login, sends a “Create Model” request to the Security Designer Controller (represented by a REST interface). The request is forwarded to the System Model Designer (GUI for model construction). As the model is being constructed it is automatically stored in the Triple Store subcomponent. The “Validate Model” command is issued via the Security Designer UI. The command checks the model and updates it with the missing assets and relations (referred to as “inferred” entities). The validation also produces a list of threats and controls to counter the threats. This information along with the updated model is presented to the user in the Security Designer UI. In the following the user issues the “Calculate risk” command that computes the risk level and probability of occurrence for each threat. The user can then select various controls to manage the risk level. This is an iterative

process, which aims to eliminate or at least mitigate the threats. The ultimate step is the presentation of a security report that summarises the threats/controls/risks for each asset and each relation of the security model. The sequence diagram representing the “Model construction, validation and threat management” workflow is in the following figure.

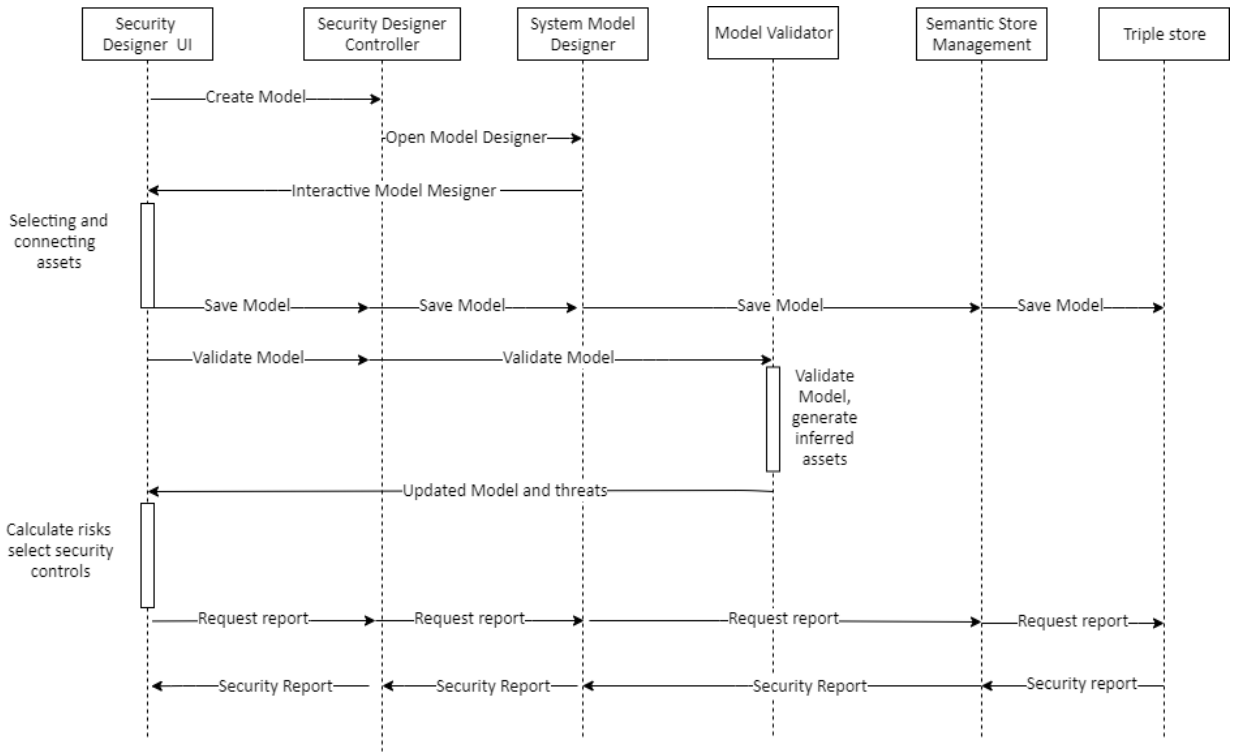


Figure 66: Model construction, validation, risk calculation and threat management

3.8 Prediction and Optimisation Designer (WP7)

3.8.1 Overall functional characterization & Context

Process Prediction and Optimisation Designer supports the development of applications that uses models to solve problems. The component provides a classification of typical prediction use cases in manufacturing processes (eg in categories such as preparation stage optimisation, process optimisation, or resource consumption optimisation), linked to prediction and optimisation models that are suited to solve the problems.

3.8.2 Functions / Features

- **Objective selection:** To configure the model project, the user selects the objective (eg minimisation or maximisation of resources). It is possible to select from a list of available pre-defined objectives according to a categorisation (eg resource efficiency maximisation, process overall efficiency optimisation, energy efficiency optimisation, start-up optimisation)
- **Performance configuration:** Different contexts require different solutions in terms of model performance (time convergence and accuracy). For the time convergence, the user defines a category of expected time performance (eg fast, medium, and slow), since time depends on the size of the problem, the algorithm used, and the available computational resources. Accuracy is the expected precision of the solution with respect to the optimal solution. In terms of usability, the user selects from a limited number of options (eg high, medium, or low)
- **Algorithm selection:** The Model Designer provides recommendations according to the project parameters in terms of available implemented models in ZDMP that can solve the problem. The user can modify this setting
- **Data source configuration:** Data sources are the fuel of models, where information is consumed to train, test and solve the business problems. Data sources are already configured in the ZDMP Platform. This feature defines the mapping between those and the inputs and outputs when training, testing, and running the model at various stages
- **Load Optimisation Model:** The Process Model Designer loads the selected model into the model runtime engine to train, test, and deploy it at the production environment
- **Train Optimisation Model:** Many models require training to customise their parametrisation according to the problem scope
- **Test Optimisation Model:** This is the stage where the algorithm is faced with a set of real data and its performance is measured to validate its appropriateness when solving the problem at hand
- **Execute Optimisation Model:** At this stage, the model is customised and has demonstrated its value addressing the business problem (eg Configuring a machine to reduce the setup time)

These functions can be further decomposed into the following subtasks that have to be realised:

Subtask	Subtask description
PE001 Create Model Project	**Priority:** Must
	Who: Process Engineer **When / Where:** During development, on the developer's environment (on-premises or in the cloud) **What:** Creates a new instance of an optimisation model **Why:** To optimise the process according to different criteria
Acceptance Criteria	A new instance of an optimisation model is available for configuration
Requirements filled	RQ-0097
PE002 Select Optimisation Objective	**Priority:** Must
	Who: Process Engineer **When / Where:** During development, on the developer's environment (on-premises or in the cloud) **What:** User selects the optimisation objective from an available set of options **Why:** To specify the type of optimisation problem to solve
Acceptance Criteria	The user can select the optimisation objective from a set of pre-defined options
Requirements filled	RQ-0096
PE003 Set Time Performance	**Priority:** Must
	Who: Process Engineer **When / Where:** During development, on the developer's environment (on-premises or in the cloud) **What:** User selects the time performance from an available set of options **Why:** To specify the type of optimisation problem to solve
Acceptance Criteria	The user can select the time performance from three categories: low, medium, high
Requirements filled	RQ-0096
PE004 Set Accuracy	**Priority:** Must
	Who: Process Engineer **When / Where:** During development, on the developer's environment (on-premises or in the cloud) **What:** User selects the accuracy from an available set of options **Why:** To specify the type of optimisation problem to solve
Acceptance Criteria	The user can select the accuracy from three categories: low, medium, high
Requirements filled	RQ-0096
PE005 Get Algorithms	**Priority:** Should
	Who: Process Engineer **When / Where:** During development, on the developer's environment (on-premises or in the cloud) **What:** User selects the optimisation algorithm from an available set of options **Why:** To specify the algorithm and model to use
Acceptance Criteria	With the optimisation, time performance, and accuracy provided, the user is prompted with a default algorithm and this default value can be updated
Requirements filled	RQ-0035, RQ-0038, RQ-0087
PE006 Set Algorithm	**Priority:** Must
	Who: Process Engineer **When / Where:** During development, on the developer's environment (on-premises or in the cloud) **What:** User confirms the selected configuration and creates the model **Why:** To build the optimisation model according to the provided configuration
Acceptance Criteria	With the optimisation, time performance, and accuracy provided, the user is prompted with a default algorithm and this default value can be updated
Requirements filled	RQ_0138

PE007 **Load Model**	**Priority:** Must
	Who: Process Engineer **When / Where:** During development, on the developer's environment (on-premises or in the cloud) **What:** User loads the created model **Why:** To load a new model instance with the provided configuration
Acceptance Criteria	If successful, the user is prompted with a confirmation message. If not, the user is prompted with an error message
Requirements filled	None. This requirement is a technical requirement that permits the indirect fulfilment of other requirements.
PE008 **Select training data source**	**Priority:** Must
	Who: Process Engineer **When / Where:** During development, on the developer's environment (on-premises or in the cloud) **What:** User selects the data sources to train the model **Why:** To configure the data sources to be used to train the model
Acceptance Criteria	The user can select the data sources that have been previously configured in the platform
Requirements filled	RQ-0044
PE009 **Train Model**	**Priority:** Must
	Who: Process Engineer **When / Where:** During development, on the developer's environment (on-premises or in the cloud) **What:** User trains the model **Why:** To train the model with the provided training sequence
Acceptance Criteria	If successful, the user is prompted with a confirmation message. If not, the user is prompted with an error message
Requirements filled	None. This requirement is a technical requirement that permits the indirect fulfilment of other requirements.
PE010 **Select test data source**	**Priority:** Must
	Who: Process Engineer **When / Where:** During development, on the developer's environment (on-premises or in the cloud) **What:** User selects the data sources to evaluate the model **Why:** To configure the data sources to be used to train the model
Acceptance Criteria	The user can select the data sources that have been previously configured in the platform
Requirements filled	RQ-0044
PE011 **Test Model**	**Priority:** Must
	Who: Process Engineer **When / Where:** During development, on the developer's environment (on-premises or in the cloud) **What:** User evaluate the model **Why:** To train the model with the provided training sequence
Acceptance Criteria	If successful, the user is prompted with a confirmation message. If not, the user is prompted with an error message
Requirements filled	None. This requirement is a technical requirement that permits the indirect fulfilment of other requirements.
PE012 **Select production data sources**	Priority: Must
	Who: Process Engineer When / Where: During configuration, on the customer's environment (on-premises or in the cloud) What: User selects the data sources to run the model Why: To configure the data sources to be used to run the model
Acceptance Criteria	The user can select the data sources that have been previously configured in the platform

Requirements filled	RQ-0044
PE013 **Execute model**	**Priority:** Must
	Who: Process Engineer **When / Where:** During development, on the customer's environment (on-premises or in the cloud) **What:** User confirms the execution of the model **Why:** To execute the optimisation model according to the provided configuration
Acceptance Criteria	If successful, the user is prompted with a confirmation message. If not, the user is prompted with an error message
Requirements filled	RQ-0044, RQ-0049, RQ-0082, RQ-0098, RQ-0913, RQ-0918
PE014 **Visualise Output Data**	**Priority:** Must
	Who: Process Engineer **When / Where:** During development, on customer's environment (on-premises or in the cloud) **What:** User visualises results of executing the model **Why:** To visualise the data inferred from the optimisation model
Acceptance Criteria	The user can visualize the optimisation model output data
Requirements filled	RQ-0049, RQ-0098, RQ_0921, RQ_0923, RQ_0949

Figure 67: Prediction and Optimization Designer Features

3.8.3 Workflows

3.8.3.1 Create Project

This workflow builds a new model in the Model Designer component based on the input provided by the user. The main steps are:

- Create a new instance of a model project
- Allow the user to configure the project with its objective, time constraint and accuracy constraint to get a list of model recommendations
- Update the model project with a user-selected algorithm

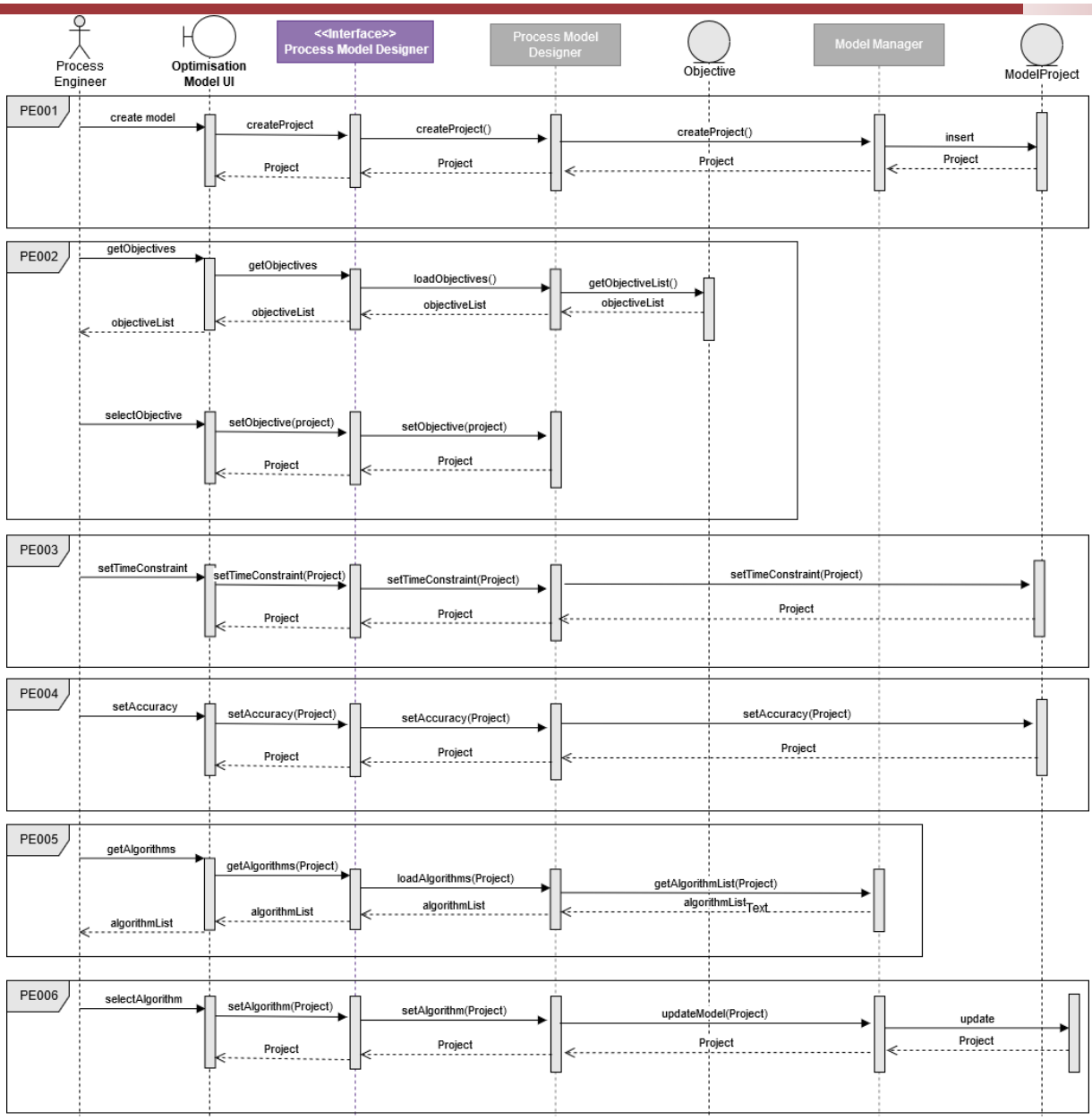


Figure 68: Create Project Sequence Diagram

3.8.3.2 Load Model

This workflow loads the selected algorithm / model that is being used to solve the problem. The main steps are:

- Find model from already existing developed models on the Models Repository
- Load the model on the Model Executor Runtime

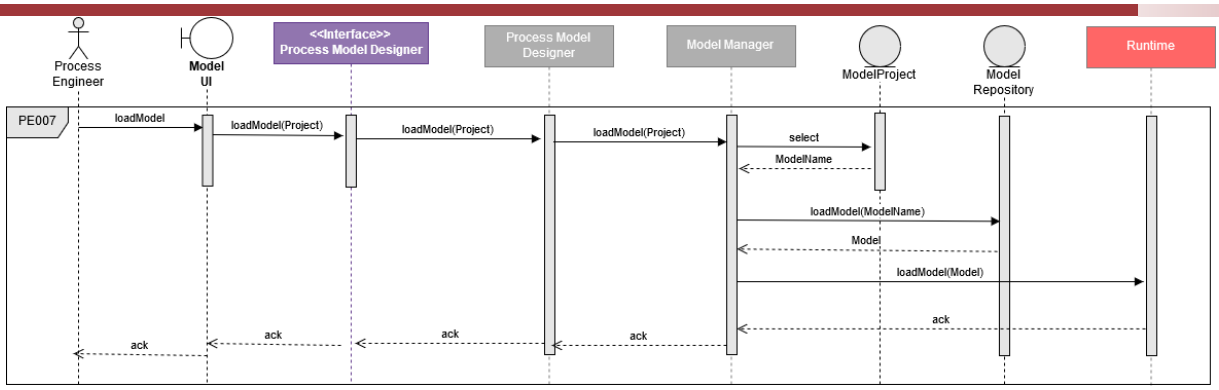


Figure 69: Load Model Sequence Diagram

3.8.3.3 Train model

This workflow trains a model instance in the Analytics and AI component based on the provided training data configured by the user. The main steps are:

- Configure the training data sources
- Train the model instance with the provided data source

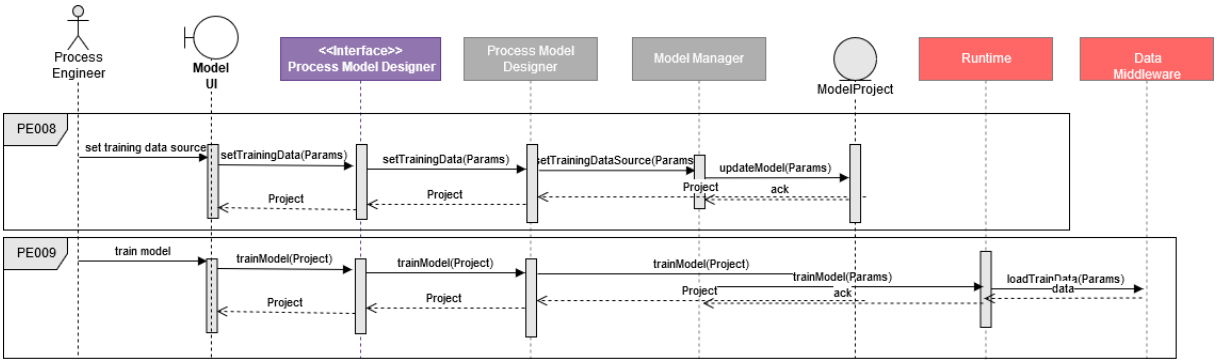


Figure 70: Train Model Sequence Diagram

3.8.3.4 Test model

This workflow trains a model instance based on the provided training data configured by the user. The main steps are:

- Configure the training data sources
- Train the model instance with the provided data source

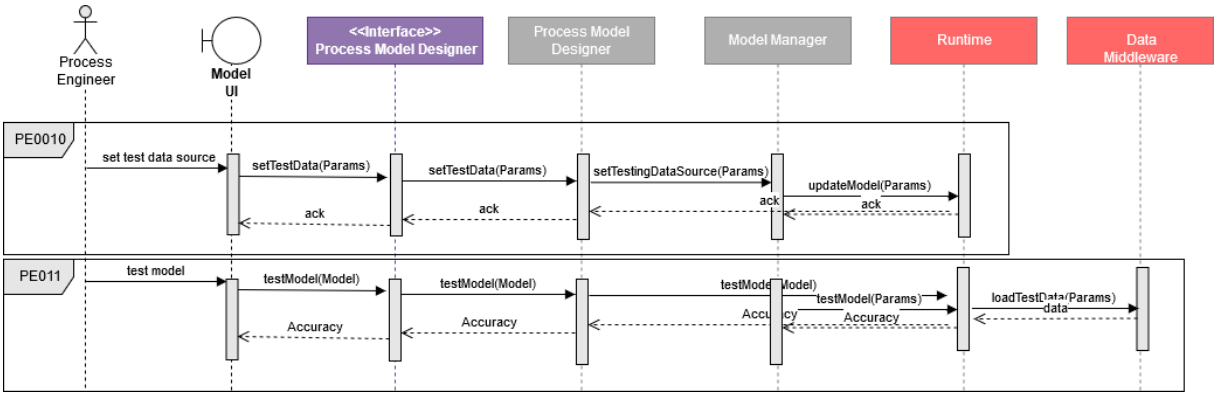


Figure 71: Test Model Sequence Diagram

3.8.3.5 Execute Model

This workflow involves using the Model Runtime to execute the model. The main steps are:

- Configure the production data sources
- Run the model against the data source

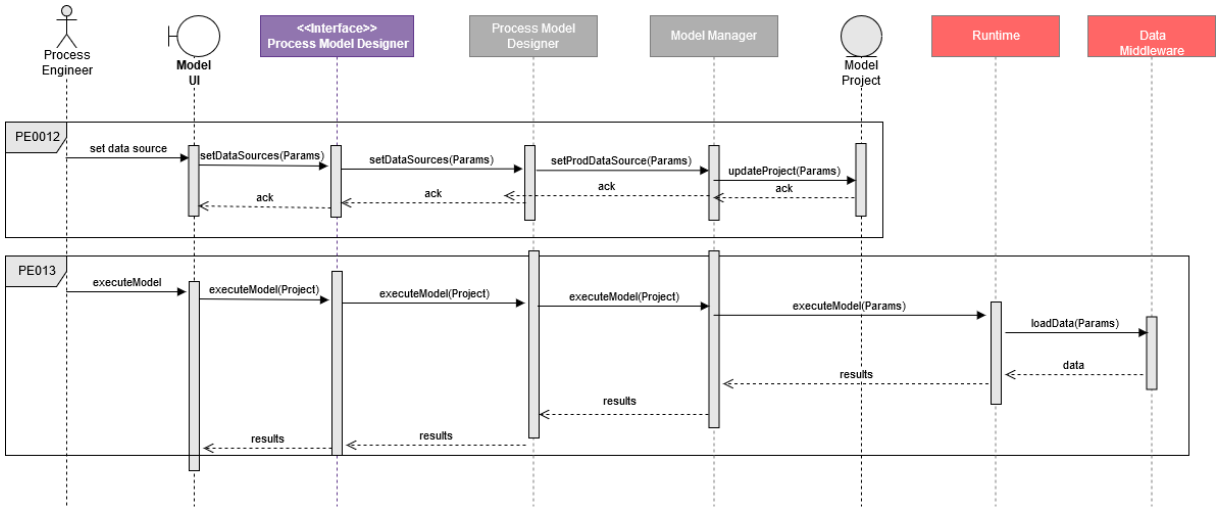


Figure 72: Execute Model Sequence Diagram

3.8.3.6 Visualize Output Data

This workflow retrieves the results from the model and represents the results in a meaningful way for the user. The main steps are:

- Subscribe to the configured data sources for output data
- Update the results
- Interpret the results and present the information to the user

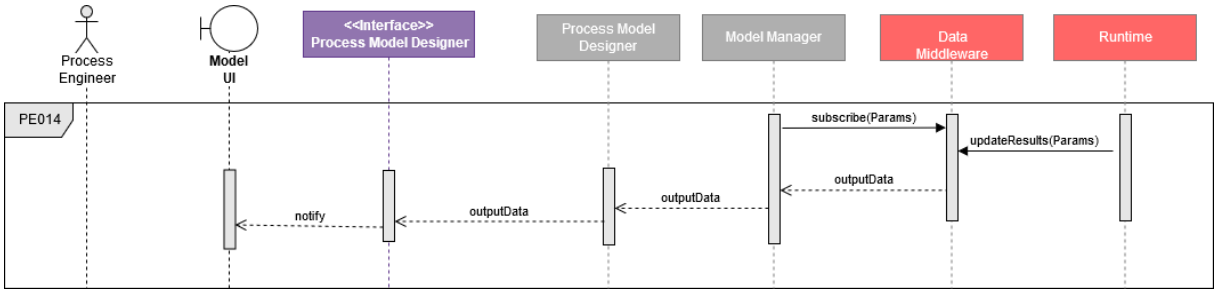


Figure 73: Visualize Output Data Sequence Diagram

3.8.4 Additional Issues

Additional issues have appeared after the description of the architecture. (If no additional issues came up, just write “None.”).

Issue	Description	Next Steps	Lead (Rationale)
Filled Requirements	The following requirements were not targeted specifically to task 7.1, but are referenced as applicable to its functionality: RQ_035, RQ_038, RQ_044, RQ_049, and RQ_082	Discuss with requirement providers who to solve the issue	T4.1 task lead
Interactions with external components	Model creation/upload, Model-loading and Model execution are referenced to be supported by other ZDMP components, but as far as competencies are not clear enough yet, they are identified generically	Discuss with technical architecture responsible and AI component responsible	
Visualiser	Model execution can produce simple and complex results. Complex results can be charts that can be executed and offered by external.	Discuss with technical architecture responsible and AI component responsible	
Input mapping	Algorithms will run on data sources offered by ZDMP IO component. Deeper discussions with IO task leader are necessary to establish interactions between the model designer, runtime, and IO.	Discuss with IO technical lead	

Figure 74: Additional Issues Prediction and Optimization Designer

4 Enterprise Tier: Use-time

Enterprise level components create the environment necessary for the ZDMP platform. This includes T5.2 Security Run-time, T6.2 Marketplace and T6.4/T6.5 Application Run-time.

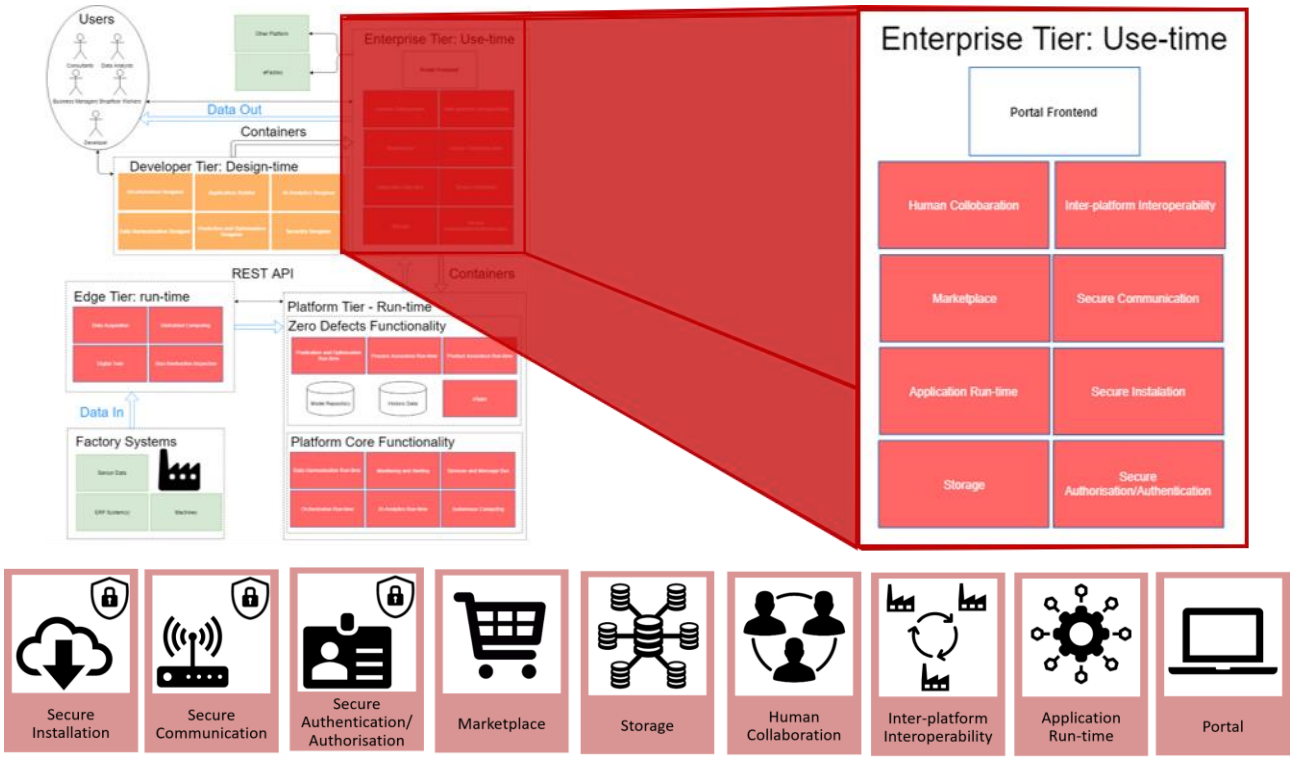


Figure 75: Enterprise Tier Components

4.1 Security Command Centre (T5.2)

4.1.1 Overall functional characterization & Context

The Security Command Centre is the main subcomponent of the security component architecture. It orchestrates the services provided by the other security modules with the aim of preventing and mitigating security incidents. This prevention is based on the monitoring of all interactions between ZDMP assets and taking the appropriate actions defined at the Security Command Centre.

4.1.2 Functions / Features

The foreseen functions and features of the Security Command Centre are the following:

- **Global security administration:** Security policies (eg password policies, access control policies, certificate policies, anti-intrusion policies, etc) can be managed and security status can be monitored in this GUI in real-time. Using this, security administrators can supervise and control the security of the ZMDP platform
- **User policies creation:** Security administrators can register and edit roles for users and zApps that permit access to ZDMP resources (eg to the storage component) in this GUI
- **Security policies creation:** Security policies can be created in this GUI, by associating roles to specific users and ZDMP assets. This process is conducted each time that a new zApp is approved for installation, so that its access permissions are well established. In addition, this component enforces the security policies, denying unauthorised access to ZDMP resources. Only administrators can edit security policies manually
  Security certificates control: In this function, the installation of new root certificates and the revocation of user certificates can be done. These actions are critical to keep the ZDMP platform secure and can be triggered either manually by a security administrator (eg caused by the exposure of a private key) or automatically (due to the expiration of a certificate).

Subtask	Subtask description
T52A001 **Access to roles and policies**	**Priority:** Must
	Who: Administrator **Where:** In the Secure Authentication/Authorisation component **When:** Runtime **What:** Show current roles and policies **Why:** Monitoring of user and ZDMP assets rights
Acceptance Criteria	The administrator was able to access roles and policies of ZDMP assets and users
Requirements filled	N/A
T52A002 **CRUD roles**	**Priority:** Must
	Who: Administrator **Where:** In the Secure Authentication/Authorisation component **When:** Runtime **What:** Perform CRUD operations on roles **Why:** Assign roles to grant or deny access rights to ZDMP protected resources
Acceptance Criteria	The administrator was able to perform all CRUD operations on a role in the Secure Authentication/Authorisation component
Requirements filled	N/A

T52A003 CRUD user	**Priority:** Must
	Who: Administrator **Where:** In the Secure Authentication/Authorisation component **When:** Runtime **What:** Perform CRUD operations on users **Why:** Provide access rights to users
Acceptance Criteria	The administrator was able to perform all CRUD operations on a user in the Secure Authentication/Authorisation component
Requirements filled	N/A
T52A004 CRUD security policies	**Priority:** Must
	Who: Administrator **Where:** In the Secure Authentication/Authorisation component **When:** Runtime **What:** Perform CRUD operations on security policies **Why:** Provide access rights to users and ZDMP assets
Acceptance Criteria	The administrator was able to perform all CRUD operations on security policies in the Secure Authentication/Authorisation component
Requirements filled	N/A

Figure 76: Security Command Centre Functions

4.1.3 Workflows

This section shows an overview of the main interaction between the Security Command Centre with other modules and components.

4.1.3.1 CRUD operations on users and ZDMP assets permissions

The Security Command Centre allows the authenticated administrator (via the Security Command Centre UI) to verify the permissions required by a zApp and create the corresponding roles, users and security policies on the Secure Authentication / Authorisation component (also via the Secure AuthN/AuthS API component).

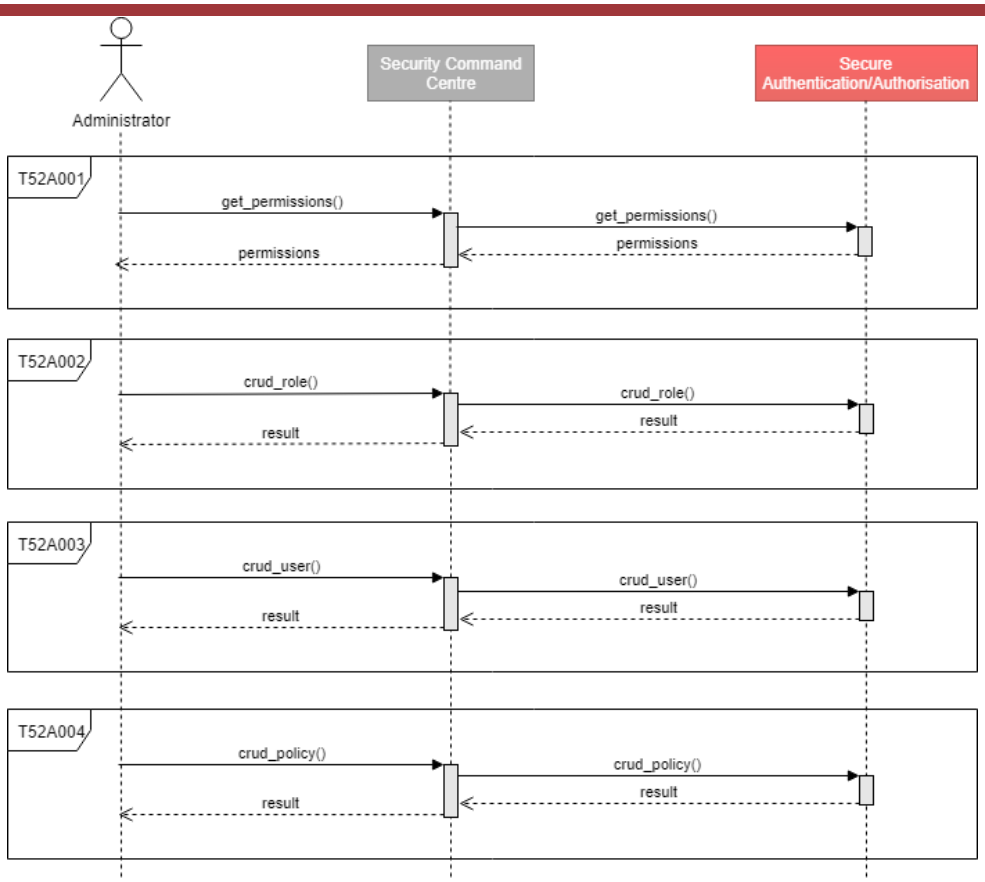


Figure 77: CRUD operations on users and ZDMP assets permissions

4.2 Installation Broker Service (T5.2)

4.2.1 Overall functional characterization & Context

The Installation Broker Service is the module that supports the Security Command Centre in the downloading and installation of a new zApp. Specifically, this module receives zApp packet downloading requests. Then, this component assists the download of the specific packet from the Marketplace and performs the corresponding signature verification. The direct download of zApps from the Marketplace (T6.2) is not allowed for security reasons. Once the signature verification is finished, this module requests the Security Command Centre the creation of the required permissions and, if this permissions creation process is successful, the Installation Broker Service forwards the zApp packet to the user through the Secure Authentication/Authorisation component acting as a proxy. Similarly, the Installation Broker Service also provides the zApp package to the Application Runtime component to control the execution of such installed zApp.

4.2.2 Functions / Features

The functions and features of the Installation Broker Service are the following:

- **ZApps download**: Users request the Installation Broker Service the download of the zApp package through the Secure Installation API. Then, the broker contacts the Marketplace and gets the corresponding zApp package
- **ZApps verification**: The Installation Broker Service conducts the security checks, such the manifest signature verification, on the downloaded zApp package
- ZApp permissions creation: When the security checks are successful, the Installation Broker Service requests the Security Command Centre to create relationships between the user, the zApp and the required permissions

Subtask	Subtask description
T52A005 Download zApp package	**Priority:** Must
	Who: ZApp user **Where:** In the Installation Broker Service module **When:** Runtime **What:** Request the download of the zApp package **Why:** Install the corresponding zApp
Acceptance Criteria	The Installation Broker Service obtained the corresponding zApp package
Requirements filled	N/A
T52A006 Verify zApp signature	**Priority:** Must
	Who: Installation Broker Service **Where:** In the Installation Broker Service module **When:** Runtime **What:** Verify all signatures included in the zApp's manifest **Why:** Check integrity of the downloaded zApp package and ensure the identity of the zApp developer
Acceptance Criteria	The signature verification process was successful
Requirements filled	N/A
T52A007	**Priority:** Must
	Who: Installation Broker Service

Create zApp permissions	**Where:** In the Secure Authentication/Authorisation component **When:** Runtime **What:** Request the creation of the different permissions required by the zApp, which are specified in the manifest **Why:** Establish permissions related to users, roles and security policies required for the proper installation and operation of the zApp
Acceptance Criteria	zApp permission creation was confirmed by the Secure Authentication/Authorisation component
Requirements filled	N/A
T52A008 Deliver zApp package	**Priority:** Must
	Who: Installation Broker Service **Where:** In the Installation Broker Service module **When:** Runtime **What:** Send the zApp package to the requesting ZDMP asset and to the Application Runtime component **Why:** Enable the installation of the zApp
Acceptance Criteria	The zApp package was received by the Secure Authentication/Authorisation component and forwarded to the corresponding ZDMP asset, and by the Application Runtime component
Requirements filled	N/A

Figure 78: Installation Broker Service Functions

4.2.3 Workflows

This section shows an overview of the main interaction between the Installation Broker Service with other modules and components.

4.2.3.1 Download and pre-installation security operations on a zApp

When an authenticated user requests to download a zApp (via API), the Installation Broker Service receives the request and then, it contacts the Marketplace to get the corresponding package. Subsequently, this module runs the signature verification process over such package and requests to the Security Command Centre the corresponding establishment of resources specified in the zApp manifest for the specific user (identified by an access token).

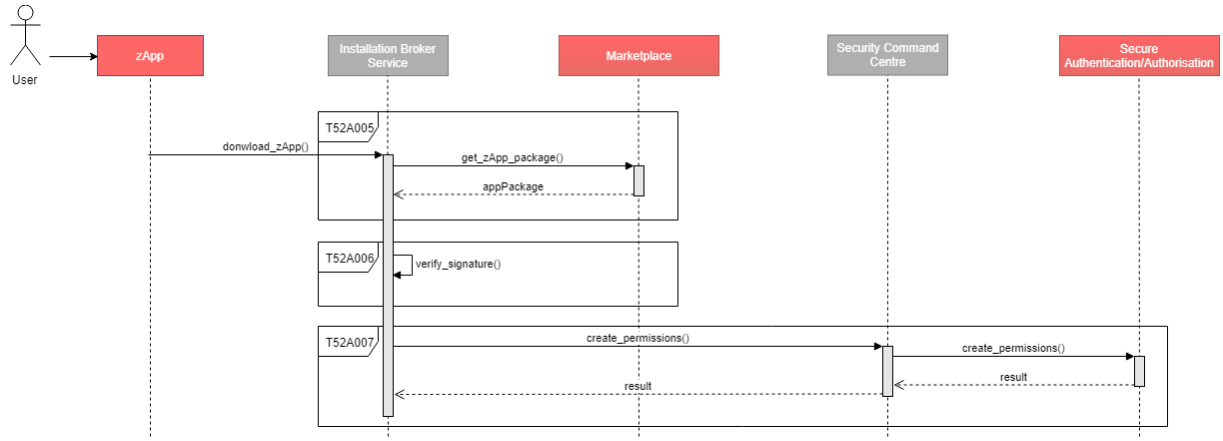


Figure 79: Download and pre-installation security operations on a zApp

4.2.3.2 Deliver the zApp package

The Installation Broker Service oversees delivering the corresponding zApp package to the Secure Authentication/Authorisation component (acting as proxy) to be forwarded to

the requester ZDMP asset. Additionally, the zApp package is also delivered to the Application Runtime component with the aim of enabling this component to control the execution of such zApp.

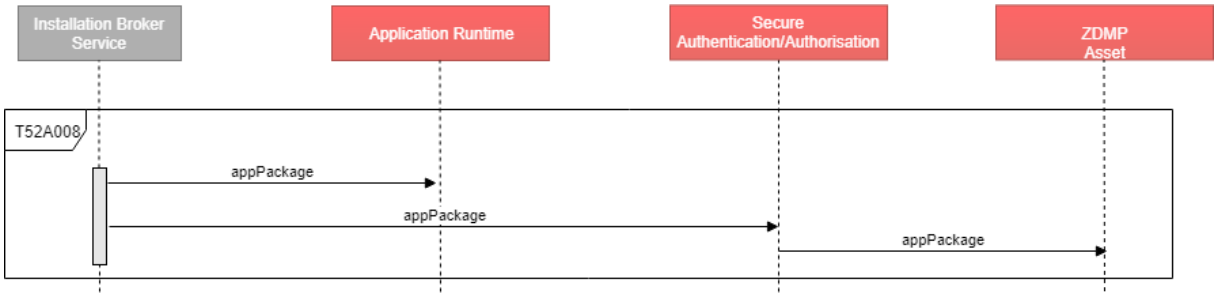


Figure 80: Deliver the corresponding zApp package

4.3 Identity Service (T5.2)

4.3.1 Overall functional characterization & Context

The Identity Service is a module that is included in the Secure Authentication / Authorisation component. It conducts the authentication of ZDMP assets (eg users, components, zApps, etc) trying to access a protected resource. The Identity Service scheme considers the current industry guidelines and standards regarding authentication and password management. Every authentication policy and subject account is managed centrally from this module. In case of a successful authentication, an associated access token is issued to the corresponding ZDMP asset.

4.3.2 Functions / Features

The functions and features of the Identity Service are the following:

- **Authentication** receives and analyses authentication requests to decide if they are legitimate or not. The defined authentication policies are applied, and the required subject information is retrieved. Once the evaluations have been conducted, a decision is made and if the authentication is valid, an access token is granted. This access token is used subsequently in the authorisation process.
- **ZDMP asset info and credentials management**: ZDMP asset information and the associated authentication credentials are stored under cryptographic protection
- **Token management**: The issued access tokens are stored which are those associated to the different authenticated ZDMP assets
- **Auth logging**: Every authentication attempt, successful or not, is registered. Authentication logs are an important source of information which could lead to the identification of security incidents, such as brute force attacks. Typical events that can be checked are:
    - Successful ZDMP asset logins
    - Failed ZDMP asset logins
    - ZDMP asset account changes
    - Password changes

Subtask	Subtask description
T52A009 **Create a new account**	**Priority:** Must
	Who: ZDMP asset **Where:** In the Identity Service module **When:** Runtime **What:** Create a new account associated with a ZDMP asset **Why:** Enable subsequent authentications and the granting of the corresponding access tokens
Acceptance Criteria	The new ZDMP asset account was successfully created
Requirements filled	RQ_0039, RQ_0073, RQ_0079, RQ_0081, RQ_0113, RQ_0241, RQ_0286, RQ_0423, RQ_0424, RQ_0425, RQ_0426, RQ_0427, RQ_0428, RQ_0429, RQ_0430, RQ_0431, RQ_0432, RQ_0433, RQ_0434, RQ_0435, RQ_0436, RQ_0437, RQ_0438, RQ_0439, RQ_0440, RQ_0441, RQ_0442, RQ_0443, RQ_0444, RQ_0469, RQ_0470, RQ_0471, RQ_0472, RQ_0473, RQ_0474, RQ_0475, RQ_0476, RQ_0477, RQ_0478, RQ_0479, RQ_0480, RQ_0481, RQ_0482, RQ_0483, RQ_0484, RQ_0485, RQ_0486, RQ_0487, RQ_0488, RQ_0526, RQ_0527, RQ_0528, RQ_0529, RQ_0530, RQ_0531, RQ_0532, RQ_0533, RQ_0534, RQ_0535, RQ_0536, RQ_0537, RQ_0538, RQ_0539, RQ_0540, RQ_0541, RQ_0542, RQ_0543, RQ_0544, RQ_0545, RQ_0546, RQ_0547, RQ_0548, RQ_0586, RQ_0587, RQ_0588, RQ_0589, RQ_0590, RQ_0591, RQ_0592, RQ_0593, RQ_0594, RQ_0595, RQ_0596, RQ_0597, RQ_0598, RQ_0599, RQ_0600, RQ_0601, RQ_0602, RQ_0603,

	RQ_0604, RQ_0605, RQ_0606, RQ_0607, RQ_0615, RQ_0619, RQ_0620, RQ_0621, RQ_0622, RQ_0668, RQ_0672, RQ_0675, RQ_0727, RQ_0734
T52A010 Authentication	**Priority:** Must
	Who: ZDMP asset **Where:** In the Identity Service module **When:** Runtime **What:** Authentication of a particular ZDMP asset **Why:** Get an ZDMP asset-associated access token to be used in subsequent authorisation operations
Acceptance Criteria	A new access token was issued and received by the corresponding ZDMP asset
Requirements filled	RQ_0039, RQ_0073, RQ_0079, RQ_0081, RQ_0113, RQ_0241, RQ_0286, RQ_0423, RQ_0424, RQ_0425, RQ_0426, RQ_0427, RQ_0428, RQ_0429, RQ_0430, RQ_0431, RQ_0432, RQ_0433, RQ_0434, RQ_0435, RQ_0436, RQ_0437, RQ_0438, RQ_0439, RQ_0440, RQ_0441, RQ_0442, RQ_0443, RQ_0444, RQ_0469, RQ_0470, RQ_0471, RQ_0472, RQ_0473, RQ_0474, RQ_0475, RQ_0476, RQ_0477, RQ_0478, RQ_0479, RQ_0480, RQ_0481, RQ_0482, RQ_0483, RQ_0484, RQ_0485, RQ_0486, RQ_0487, RQ_0488, RQ_0526, RQ_0527, RQ_0528, RQ_0529, RQ_0530, RQ_0531, RQ_0532, RQ_0533, RQ_0534, RQ_0535, RQ_0536, RQ_0537, RQ_0538, RQ_0539, RQ_0540, RQ_0541, RQ_0542, RQ_0543, RQ_0544, RQ_0545, RQ_0546, RQ_0547, RQ_0548, RQ_0586, RQ_0587, RQ_0588, RQ_0589, RQ_0590, RQ_0591, RQ_0592, RQ_0593, RQ_0594, RQ_0595, RQ_0596, RQ_0597, RQ_0598, RQ_0599, RQ_0600, RQ_0601, RQ_0602, RQ_0603, RQ_0604, RQ_0605, RQ_0606, RQ_0607, RQ_0615, RQ_0619, RQ_0620, RQ_0621, RQ_0622, RQ_0668, RQ_0672, RQ_0675, RQ_0727, RQ_0734

Figure 81: Identity Service Features

4.3.3 Workflows

This section shows an overview of the main interaction between the Identity Service with other modules and components.

4.3.3.1 Account creation and authentication

The Identity Service allows the ZDMP assets to create an account and register certain attributes associated to them. Once an account is created, the corresponding ZDMP asset is enabled to launch the authentication process by using the user and password previously registered.

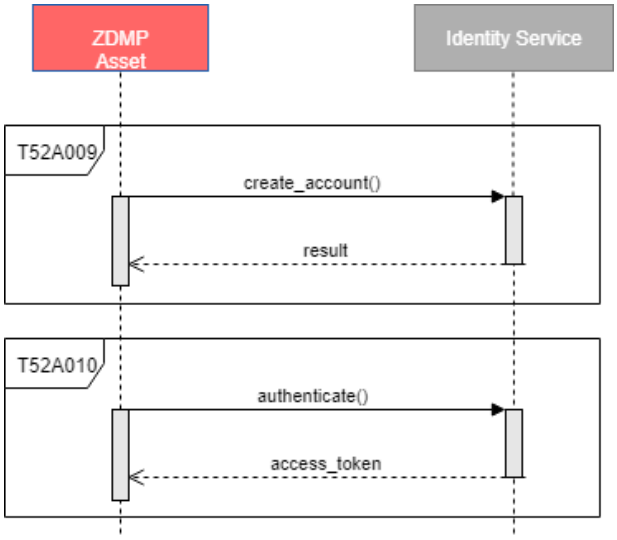


Figure 82: Account creation and authentication processes of a ZDMP asset

4.4 Authorisation Service (T5.2)

4.4.1 Overall functional characterization & Context

The Authorisation Service is another module of the Secure Authentication/Authorisation component. Authorisation to access protected resources is a complex task as it involves advanced security concepts (identity-based, role-based access control, attribute-based access control, etc). Most developers embed the authorisation logic within the application code, which makes it hard to maintain, evolve, and integrate with external services providing extra authorisation attributes. To reduce the authorisation logic and avoid these issues, the Authorisation Service takes advantage of flexible and standard-compliant authorisation schemes.

4.4.2 Functions / Features

The functions and features of the Authorisation Service are the following:

- **Policy enforcement**: Every request made from a specific ZDMP asset is intercepted and transformed into a well-formed eXtensible Access Control Markup Language (XACML) request, to be approved or rejected. Only if the request is approved, it is forwarded
- **Policy decision**: Every XACML request is assessed and answered (ie, acceptance or rejection). The decisions are primarily based on the authorisation policies administrated. Nevertheless, it is possible to include other external conditions to decide
- **Policy administration**: Authorisation policies are stored and retrieved when requested. These authorisation policies are created and edited under request from the Security Command Centre module
- **Policy information provision**: Additional attributes, coming from multiple sources, are retrieved to enable more informed decision during the authorisation process. This enables the security model, based on Role-based Access Control (RBAC), to be extended with Attribute-Based Access Control (ABAC), also referred to as RBAC-ABAC, which is the recommended access control model today.

x

Subtask	Subtask description
T52A011 Resource access authorisation	**Priority:** Must
	Who: ZDMP asset **Where:** In the Authorisation Service module **When:** Runtime **What:** Get authorised access to protected resources **Why:** Protect resources from unauthorised accesses
Acceptance Criteria	ZDMP asset was able to access a protected resource
Requirements filled	RQ_0039, RQ_0073, RQ_0079, RQ_0081, RQ_0113, RQ_0241, RQ_0286, RQ_0423, RQ_0424, RQ_0425, RQ_0426, RQ_0427, RQ_0428, RQ_0429, RQ_0430, RQ_0431, RQ_0432, RQ_0433, RQ_0434, RQ_0435, RQ_0436, RQ_0437, RQ_0438, RQ_0439, RQ_0440, RQ_0441, RQ_0442, RQ_0443, RQ_0444, RQ_0469, RQ_0470, RQ_0471, RQ_0472, RQ_0473, RQ_0474, RQ_0475, RQ_0476, RQ_0477, RQ_0478, RQ_0479, RQ_0480, RQ_0481, RQ_0482, RQ_0483, RQ_0484, RQ_0485, RQ_0486, RQ_0487, RQ_0488, RQ_0526, RQ_0527, RQ_0528, RQ_0529, RQ_0530, RQ_0531, RQ_0532, RQ_0533, RQ_0534, RQ_0535, RQ_0536, RQ_0537, RQ_0538, RQ_0539, RQ_0540, RQ_0541, RQ_0542, RQ_0543, RQ_0544, RQ_0545, RQ_0546, RQ_0547, RQ_0548, RQ_0586, RQ_0587, RQ_0588, RQ_0589, RQ_0590, RQ_0591, RQ_0592, RQ_0593, RQ_0594, RQ_0595, RQ_0596, RQ_0597, RQ_0598, RQ_0599, RQ_0600, RQ_0601, RQ_0602, RQ_0603, RQ_0604, RQ_0605, RQ_0606, RQ_0607, RQ_0615, RQ_0619, RQ_0620, RQ_0621, RQ_0622, RQ_0668, RQ_0672, RQ_0675, RQ_0727, RQ_0734

Figure 83: Authorization Service Features

4.4.3 Workflows

This section shows an overview of the main interaction between the Authorisation Service with other modules and components.

4.4.3.1 Request authorised access to protected resources

The Authorisation Service allows the ZDMP assets to request access to protected resources. To accept or deny such requests, this component bases on the corresponding access token and the stored authorisation policies.

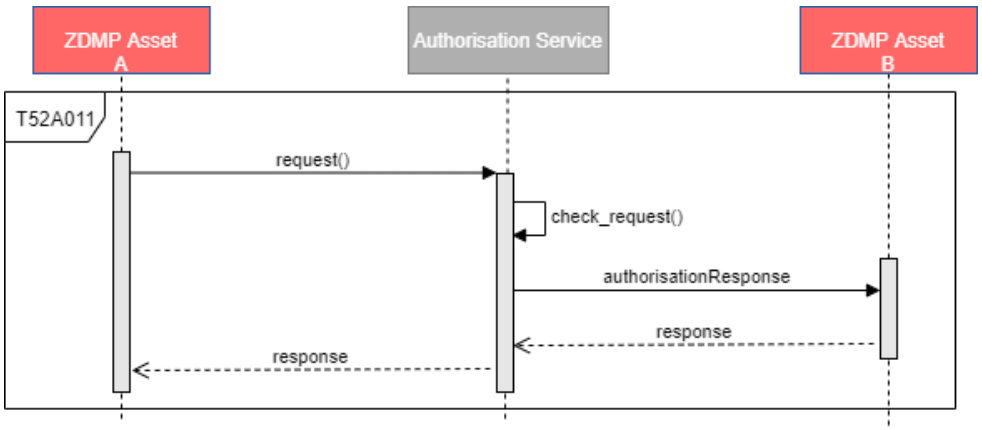


Figure 84: Request authorised access to a protected resource

4.5 Intrusion Detection Service (T5.2)

4.5.1 Overall functional characterization & Context

The Intrusion Detection Service is another the Secure Authentication/Authorisation component. This module works together with another service called Authorisation Service, by analysing all requests and generating alerts and logs if any suspicious activity is detected. Specifically, the Intrusion Detection Service conducts certain verifications over each authorisation request, such as pattern analysis in the access token or detecting cybersecurity attacks, such as Denial of Service (DoS).

4.5.2 Functions / Features

The functions and features of the Intrusion Detection Service is the following:

- **Detect suspicious activity**: The Intrusion Detection Service monitors all communications among ZDMPS assets and Authorisation Service to find out possible cases of cyberattacks
- **Logging suspicious activity**: When the Intrusion Detection Service detects any suspicious activity, this module registers it in a secure log database, which can be analysed by the administrators
- **Generate alerts**: In parallel with the logging task, this module also sends several types of alerts to notify the appropriate recipients (eg administrators or ZDMP users)

Subtask	Subtask description
T52A012 **React to a suspicious activity**	**Priority:** Must
	Who: Detection Intrusion Service **Where:** In the Detection Intrusion Service module **When:** Runtime **What:** Detect, log, and alert any suspicious activity **Why:** Protect resources from intrusions
Acceptance Criteria	Detection Intrusion Service was able to detect a suspicious activity pattern
Requirements filled	RQ_0039, RQ_0073, RQ_0079, RQ_0081, RQ_0113, RQ_0241, RQ_0286, RQ_0423, RQ_0424, RQ_0425, RQ_0426, RQ_0427, RQ_0428, RQ_0429, RQ_0430, RQ_0431, RQ_0432, RQ_0433, RQ_0434, RQ_0435, RQ_0436, RQ_0437, RQ_0438, RQ_0439, RQ_0440, RQ_0441, RQ_0442, RQ_0443, RQ_0444, RQ_0469, RQ_0470, RQ_0471, RQ_0472, RQ_0473, RQ_0474, RQ_0475, RQ_0476, RQ_0477, RQ_0478, RQ_0479, RQ_0480, RQ_0481, RQ_0482, RQ_0483, RQ_0484, RQ_0485, RQ_0486, RQ_0487, RQ_0488, RQ_0526, RQ_0527, RQ_0528, RQ_0529, RQ_0530, RQ_0531, RQ_0532, RQ_0533, RQ_0534, RQ_0535, RQ_0536, RQ_0537, RQ_0538, RQ_0539, RQ_0540, RQ_0541, RQ_0542, RQ_0543, RQ_0544, RQ_0545, RQ_0546, RQ_0547, RQ_0548, RQ_0586, RQ_0587, RQ_0588, RQ_0589, RQ_0590, RQ_0591, RQ_0592, RQ_0593, RQ_0594, RQ_0595, RQ_0596, RQ_0597, RQ_0598, RQ_0599, RQ_0600, RQ_0601, RQ_0602, RQ_0603, RQ_0604, RQ_0605, RQ_0606, RQ_0607, RQ_0615, RQ_0619, RQ_0620, RQ_0621, RQ_0622, RQ_0668, RQ_0672, RQ_0675, RQ_0727, RQ_0734

Figure 85: Intrusion Detection Service Features

4.5.3 Workflows

This section shows an overview of the main interaction between the Intrusion Detection Service with other modules and components.

4.5.3.1 React to a suspicious activity

The Intrusion Detection Service avoids intrusions to the ZDMP platform. Once an intrusion is detected, this module logs such suspicious activity in a secure database to provider

traces of this activity to the administrators. These traces are of use to administrators to find out potentially unauthorised accesses and to adapt the intrusion detection rules according to normal behaviour of ZDMP assets.

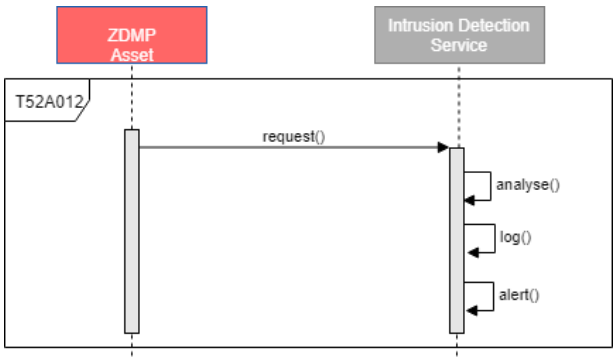


Figure 86: React to a suspicious activity

4.6 Secure Communications PKI Service (T5.2)

4.6.1 Overall functional characterization & Context

The communications Public Key Infrastructure (PKI) service is the subcomponent that provides the credentials (digital certificates and keys) and functions required to further establish secure communications between physical devices, gateways, and servers identified as ZDMP Assets. Every ZDMP communication is initiated with a handshake where security credentials are exchanged and mutually verified. This handshake provides the security baseline for the mutual authentication of the communication devices, and for agreement on the security configuration of the communications. As security credentials are critical elements, they are managed according to international recommendations (eg NIST recommendations), which establish requirements regarding maximum usage periods, minimum key lengths, etc. The PKI service, in cooperation with the Security Command Centre, integrates functions to comply with these recommendations.

4.6.2 Functions / Features

The functions and features of the Communications PKI Service are the following:

- Certificate issue: New client certificates are created. These certificates include the details that permit the identification of the subject (physical device, gateway, or server)
- Certificate revocation or renewal: Certificates are registered that are out-of-date or have been compromised (in Certificate Revocation Lists, CRLs), to guarantee that their use is discontinued. The Security Command Centre can request clients to renew their certificate to avoid revocations
- Certificate retrieval: Stored certificates are retrieved based on their identifying details (eg certificate subject)
- Root and intermediate certificate installation: Where new root and intermediate certificates are installed, so that they can be used to sign and verify other (client) certificates

Subtask	Subtask description
T52A013 **Issue new certificates**	**Priority:** Must
	Who: Physical devices, gateways, servers **Where:** In Communications PKI Service module **When**: Before establishing new connection sockets **What:** Request a new ZDMP client digital certificates **Why:** Enable secure communications
Acceptance Criteria	A physical device (or any other ZDMP asset) received its own digital certificate
Requirements filled	RQ_0025, RQ_0033, RQ_0039, RQ_0043, RQ_0079, RQ_0081, RQ_0093, RQ_0099, RQ_0100, RQ_0113, RQ_0116, RQ_0136, RQ_0198, RQ_0241, RQ_0286
T52A014 **Install root certificates**	**Priority:** Must
	Who: Security Command Centre **Where:** In Communications PKI Service module **When**: Before issuing client certificates with the Comms PKI service **What:** Deliver root certificates to ZDMP PKI **Why:** Update ZDMP chain of trust
Acceptance Criteria	The ZDMP PKI installed the new root or intermediate certificate
Requirements filled	RQ_0025, RQ_0033, RQ_0039, RQ_0043, RQ_0079, RQ_0081, RQ_0093, RQ_0099, RQ_0100, RQ_0113, RQ_0116, RQ_0136, RQ_0198, RQ_0241, RQ_0286
T52A015	**Priority:** Should

Revoke certificates	**Who:** Security Command Centre **Where:** In Communications PKI Service module **When**: After permanent expiration or detecting an invalid certificate **What:** Revoke a certificate **Why:** Prevent insecure certificates from enabling trusted communications
Acceptance Criteria	The certificate was revoked
Requirements filled	RQ_0025, RQ_0033, RQ_0039, RQ_0043, RQ_0079, RQ_0081, RQ_0093, RQ_0099, RQ_0100, RQ_0113, RQ_0116, RQ_0136, RQ_0198, RQ_0241, RQ_0286
T52A016 **Renew certificates**	**Priority:** Should
	Who: Security Command Centre **Where:** In Communications PKI Service module **When**: After expiration of a client certificate that still needs secure comms. **What:** Renew a certificate **Why:** Prevent expired certificates from enabling trusted communications
Acceptance Criteria	The certificate was renewed
Requirements filled	RQ_0025, RQ_0033, RQ_0039, RQ_0043, RQ_0079, RQ_0081, RQ_0093, RQ_0099, RQ_0100, RQ_0113, RQ_0116, RQ_0136, RQ_0198, RQ_0241, RQ_0286
T52A017 **Certificate retrieval**	**Priority:** Must
	Who: Security Command Centre or T6.2 Security Designer **Where**: In Communications PKI Service module **When**: On demand to check certificate details and secure links **What:** Retrieve certificates from IDS via Comms. PKI Service **Why:** Manage ZDMP chain of trust, check validity against credentials or expiry dates
Acceptance Criteria	The Security Command Centre received a requested certificate. If the request is coming from T6.2, the Security designer received requested certificate list.
Requirements filled	RQ_0025, RQ_0033, RQ_0039, RQ_0043, RQ_0079, RQ_0081, RQ_0093, RQ_0099, RQ_0100, RQ_0113, RQ_0116, RQ_0136, RQ_0198, RQ_0241, RQ_0286

Figure 87: Secure Communications PKI Service Features

4.6.3 Workflows

4.6.3.1 Issue new certificates

The Comms. PKI Service provides ZDMP Assets with the possibility to request a client certificate that enables establishing secure channels with other elements inside ZDMP runtime platform. Upon receiving the request, the Comms. PKI Service validates the parameters involved, such as identifying if such client already has a valid certificate. If there is the need to issue a new certificate, the PKI service creates it, replies to the client, and stores the new certificate in the corresponding IDS.

The main steps / functionalities are:

- Invoking the Comms. PKI Service through the API to request a certificate
- The Comms. PKI Service determines the necessity of a new certificate
- If the request fails, an error message is sent to the Asset
- The Comms. PKI Service creates the certificate and stores it in the IDS
- The Comms. PKI Service, through the API, sends the certificate to the client

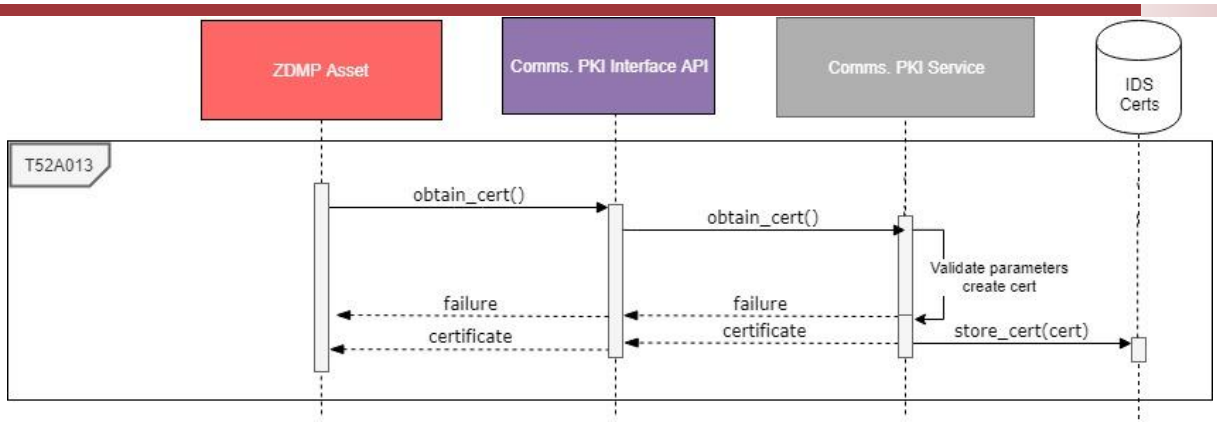


Figure 88: Issue New Certificates Sequence Diagram

4.6.3.2 Install Root certificates

The Security Command Centre can install root or intermediate certificates in the Comms. PKI Service, enabling the service to sign client certificates for ZDMP assets that are trusted for being under a root Certificate Authority (CA).

The main steps / functionalities are:

- The Security Command Centre requests the installation of a root/intermediate certificate to the Comms. PKI Service
- The Comms. PKI Service installs and stores the received certificate
- The Comms. PKI Service confirms the installation or informs with an error if fails

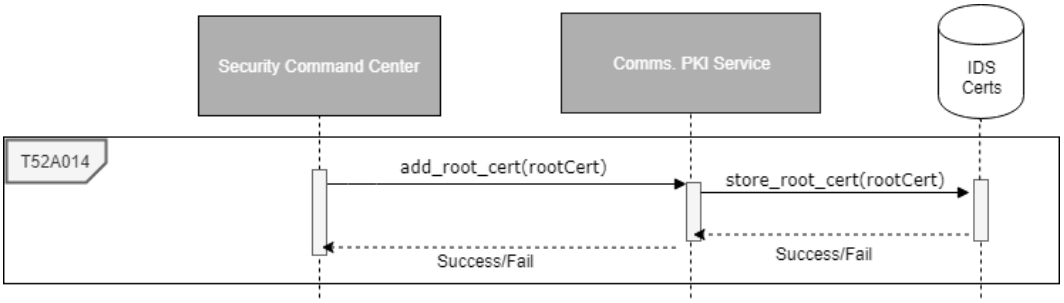


Figure 89: Install Root Certificates Sequence Diagram

4.6.3.3 Revoke certificates

The Security Command Centre can request the revocation of a client certificate to the Comms. PKI Service. Upon receiving the request, the Comms. PKI Service validates the parameters of the certificate and adds it to the Certificate Revocation List (CRL), which needs to be updated to analyse the validity of certificates exchanged in the platform communications.

The main steps / functionalities are:

- The Security Command Centre requests the revocation of a certificate to the Comms. PKI Service
- The Comms. PKI Service checks details of revoked certificate and adds it to the CRL
- The Comms. PKI Service confirms the success or failure of the revocation

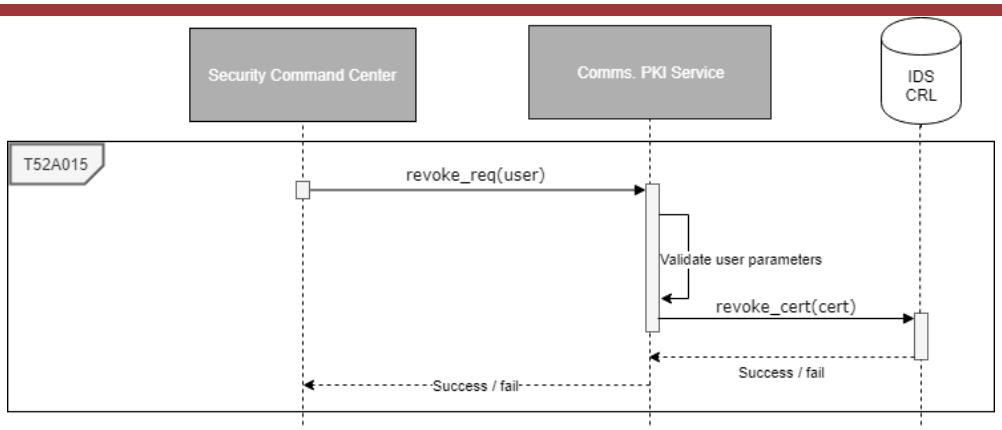


Figure 90: Revoke Certificates Sequence Diagram

4.6.3.4 Renew certificates

The Security Command Centre can request a ZDMP Asset to renew its client certificate to the Comms. PKI Service. The client then sends the request via Comms. PKI API. Upon receiving the request via the API, the Comms. PKI Service validates the parameters of the certificate, updates the expiration date of given certificate (also in the IDS) and replies to the client with the updated certificate.

The main steps / functionalities are:

- The Security Command Centre requests the renewal of a certificate to a ZDMP Asset, via the API
- The ZDMP Asset invokes the Comms. PKI Service through the API to request an updated certificate
- The Comms. PKI Service validates the details of the certificate
- If renewal fails, an error message is propagated from PKI service towards the SCC.
- The Comms. PKI Service updates the certificate and stores it in the IDS
- The Comms. PKI Service, through the API, sends the certificate to the client

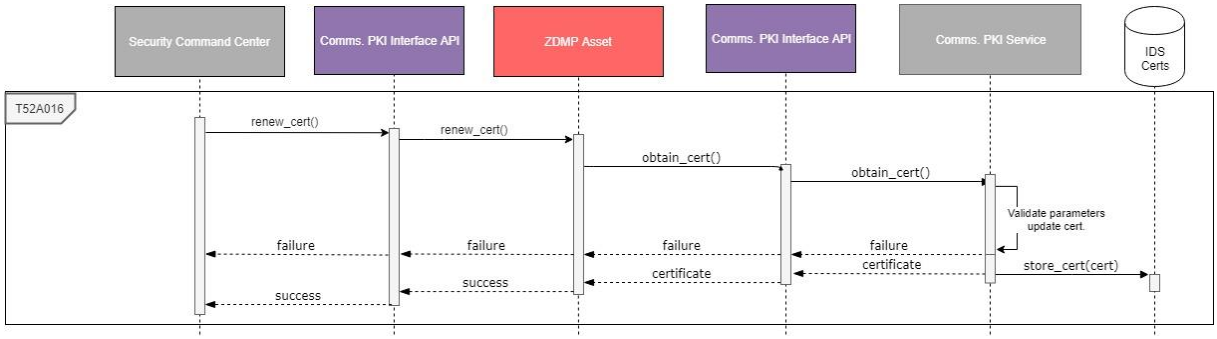


Figure 91: Renew Certificates Sequence Diagram

4.6.3.1 Retrieve certificates

The Security Command Centre can retrieve certificates in the Comms. PKI Service, enabling the SCC to check further actions needed to ensure the chain of trust in the platform, such as requesting the renewal or revocation of client certificates. It is also needed to check which secure links are stored in the certificates IDS, to reply to queries from T6.2 Security Designer.

The main steps / functionalities are:

- The Security Command Centre requests the details of a certificate to the Comms. PKI Service
- The Comms. PKI Service retrieves the certificate from the IDS
- The Comms. PKI Service sends back the requested certificate to the Security Command Centre

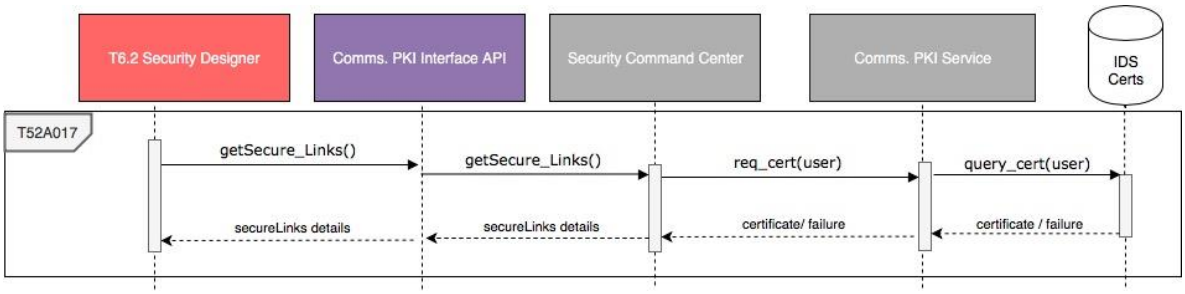


Figure 92: Retrieve Certificates Sequence Diagram

4.7 Marketplace (T6.2)

4.7.1 Overall functional characterisation & Context

The Marketplace component has a similar business model to the Google Play or the Apple App Store, allowing end users to search for and buy applications and services from manufacturing sector. Also, users can post their specifications and demand specific applications and developers can build their own applications and share them in the Marketplace. The main building blocks of the Marketplace component and their interactions were detailed in the Architecture Document.

4.7.2 Functions / Features

- **User authentication:** this is about access rights of users connected to Marketplace. This is facilitated by the User Management & Authorisation via T5.2 Security Run-time component
- **Negotiation**: this functionality allows ZDMP users and Application Providers to get in touch and negotiate with each other about possible new ZD Assets and the conditions of the development. This is possible due to Negotiation Environment component accessible from Negotiation UI
- **Browse items**: through the Store Frontend UI the users can browse and examine ZD Assets and their individual information such as screenshots, usage fees, and reviews or ratings of other users, as well as licensing policies and payment methods.
- **Administration:** this feature allows ZD Asset Providers and administrators to manage and upload their items and view usage and error statistics or manage the Marketplace as a whole. Both types of users are managed through granted rights and roles. Furthermore, it is used by support users to help users with potential issues.
- **Place order:** this feature allows users to initiate the buying process and it is managed by the Order Manager module. All information regarding orders are stored in the Order Database component.
- **Issue invoice**: this allows users to receive invoices for their bought ZD Assets and licenses. It is executed by the Invoicing module connected to Order Manager module. Invoices are sent to the users by the means of Notification module.
- **Notification**: all the information regarding the items from Marketplace exchanged between the users and administrators exchanged are possible due to the Notification Manager module linked directly to Human Collaboration component.
- **Payment**: this feature manages all exchanges of data with external payment providers, due to Payment Manager module. Besides payments, this include cancellation- and refund-handling, and fraud-prevention.
- **Record usage data**: for pay-per-use items it is necessary to collect and store usage data to invoice the service. This is possible due to the Usage Data Interface which is a gateway to the Usage Analytics & Error Reporting module for the ZDMP platform which provides this information from the ZD Assets.

These functions can be further decomposed into the following subtasks that have to be realised:

Subtask	Subtask description
T62A001 **Connect to Marketplace**	**Priority:** Must
	Who: Marketplace **Where:** Anywhere **When:** Anytime **What:** Connect to Marketplace and get user access rights set via T5.2 Security Run-time component **Why:** So that the user can access the UI and functionalities of the Marketplace
Acceptance Criteria	User successfully connected
Requirements filled	N/A
T62A002 User **Negotiation**	**Priority:** Must
	Who: Marketplace **Where:** Anywhere **When:** Anytime **What:** Users and application providers negotiate with each other about possible new ZD Assets and the conditions of the development **Why:** So that the users can request applications based on their specifications
Acceptance Criteria	Communication between users and developers facilitated.
Requirements filled	N/A
T62A003 **Search items**	**Priority:** Must
	Who: Marketplace **Where:** Anywhere **When:** Anytime **What:** Browse and search for specific ZDMP assets **Why:** So that the user can examine assets and their individual information such as screenshots, usage fees, and reviews or ratings of other users.
Acceptance Criteria	Asset successfully found
Requirements filled	N/A
T62A004 **Upload items**	**Priority:** Must
	Who: Marketplace **Where:** Anywhere **When:** Anytime **What:** Administrators and providers upload ZDMP assets **Why:** So that the users can search for specific assets
Acceptance Criteria	Asset successfully uploaded
Requirements filled	N/A
T62A005 **Place Order**	**Priority:** Must
	Who: Marketplace **Where:** Anywhere **When:** Anytime **What:** User make an order for a specific asset and save it in Order Database **Why:** So that the user can buy the specific asset
Acceptance Criteria	Order successfully saved
Requirements filled	N/A
T62A006	**Priority:** Must

Issue invoice	**Who:** Marketplace **Where:** Anywhere **When:** Anytime **What:** The Invoicing module creates an invoice for the order saved **Why:** So that the user can pay for the chosen asset
Acceptance Criteria	Invoice successfully created.
Requirements filled	None
T62A007 **Notification**	**Priority:** Must
	Who: Marketplace **Where:** Anywhere **When:** Anytime **What**: User receives information regarding the items **Why:** So that the user can be informed about a specific item
Acceptance Criteria	None
Requirements filled	Information sent and received successfully
T62A008 **Payment**	**Priority:** Must
	Who: Marketplace **Where:** Anywhere **When:** Anytime **What:** User pays the invoice for the chosen asset **Why:** So that the buying process can be processed
Acceptance Criteria	Payment successfully done
Requirements filled	None
T62A009 **Record Usage Data**	**Priority:** Must
	Who: Marketplace **Where:** Anywhere **When:** Anytime **What:** Collect and store usage data for pay-per-use items **Why:** So that the pay-per-use items can be invoiced
Acceptance Criteria	Usage data successfully saved
Requirements filled	None

Figure 93: Marketplace Features

4.7.3 Workflows

The following sub-sections describe the sequence diagrams of the Marketplace component.

4.7.3.1 Connect to Marketplace

The following diagram explains this function and the necessary interactions with other components.

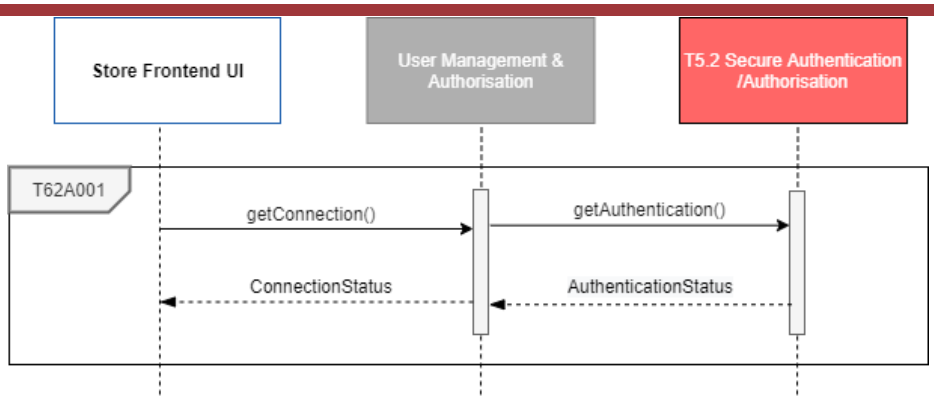


Figure 94: Connect to marketplace sequence diagram

4.7.3.2 User Negotiation

The following diagram explains this function and the necessary interactions with other components.

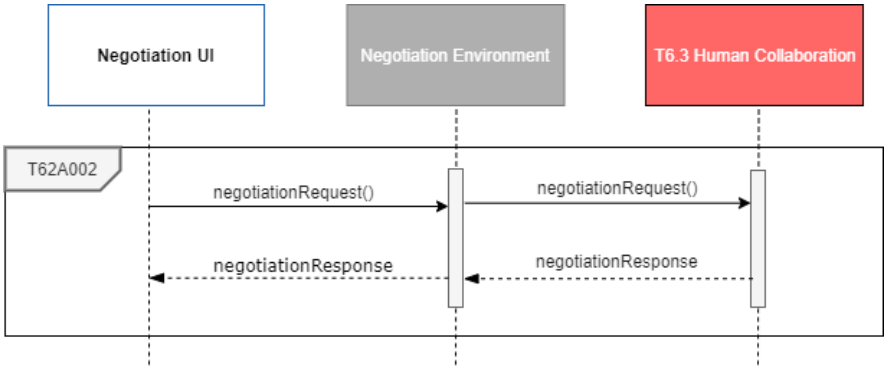


Figure 95: User negotiation sequence diagram

4.7.3.3 Search items

The following diagram explains this function and the necessary interactions with other components.

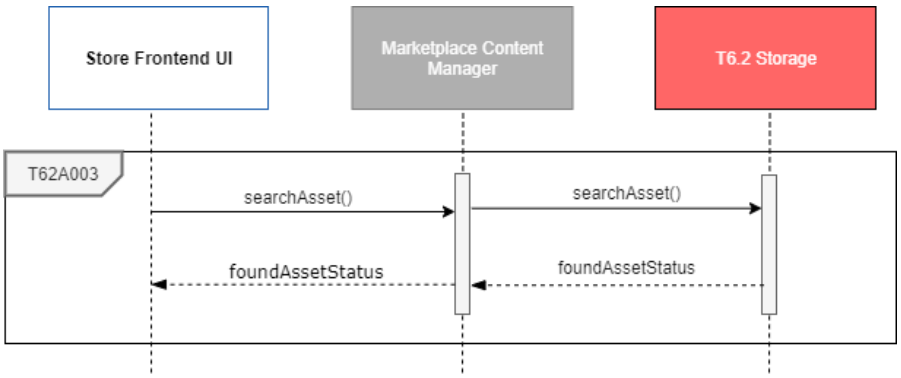


Figure 96: Search items sequence diagram

4.7.3.4 Upload items

The following diagram explains this function and the necessary interactions with other components.

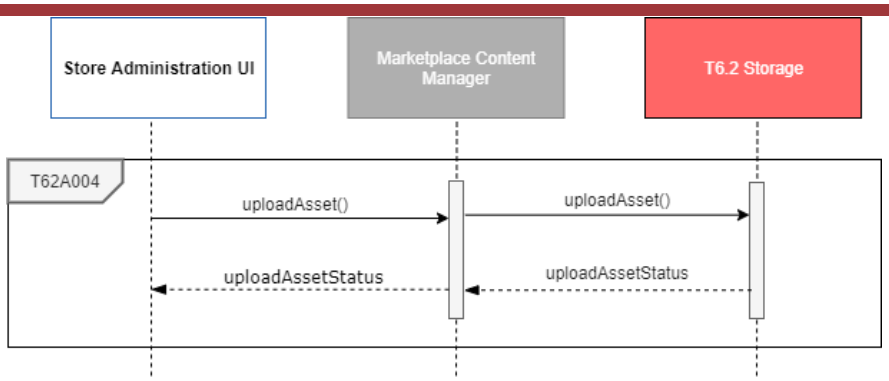


Figure 97: Upload items sequence diagram

4.7.3.5 Place order

The following diagram explains this function and the necessary interactions with other components.

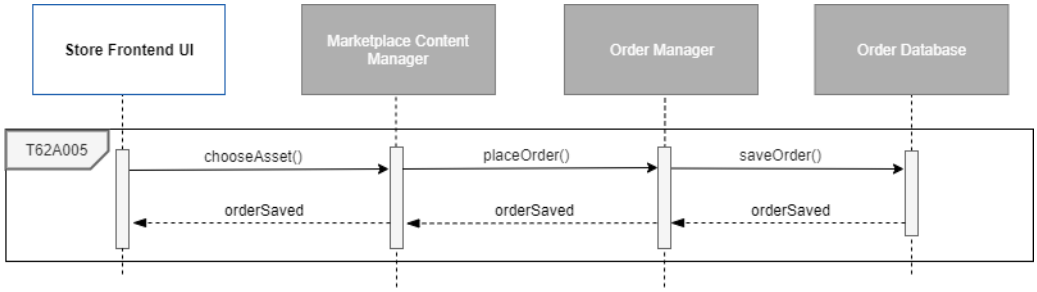


Figure 98: Place order sequence diagram

4.7.3.6 Issue invoice and notification

The following diagram explains this function and the necessary interactions with other components.

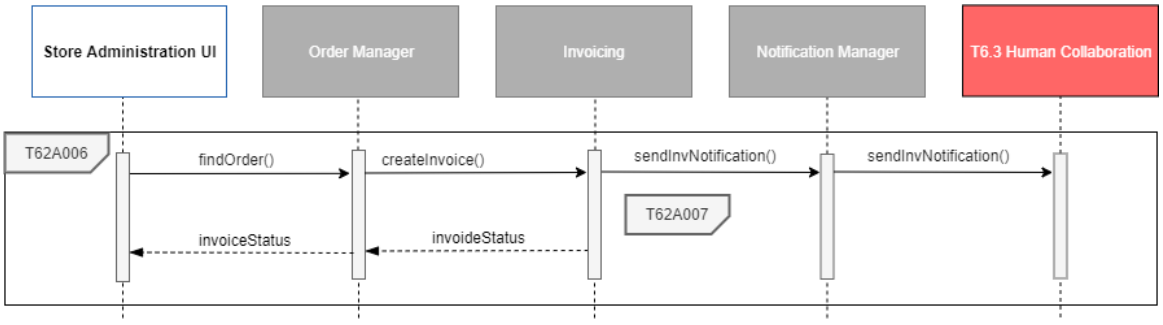


Figure 99: Issue invoice and notification sequence diagram

4.7.3.7 Payment

The following diagram explains this function and the necessary interactions with other components.

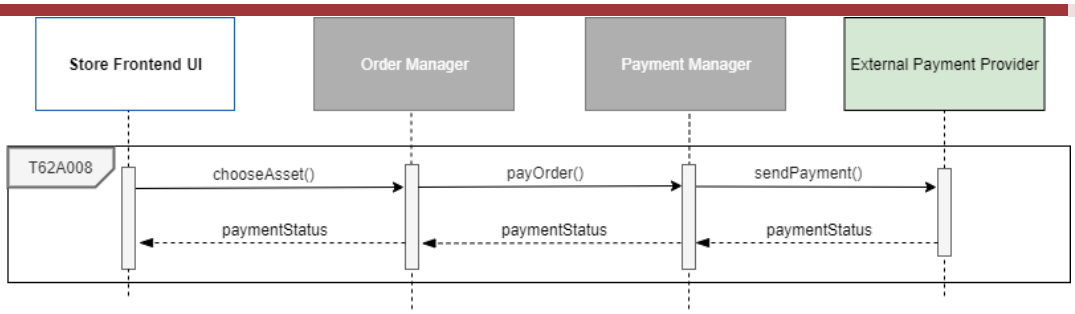


Figure 100: Payment sequence diagram

4.7.3.8 Record usage data and invoicing

The following diagram explains this function and the necessary interactions with other components.

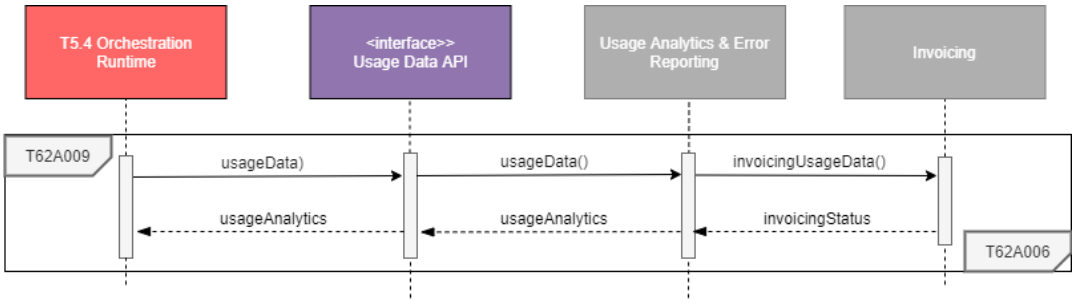
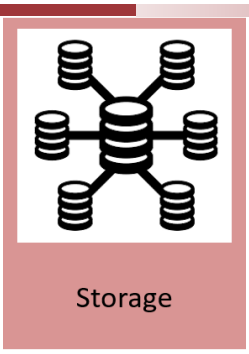


Figure 101: Record usage data and invoicing sequence diagram

4.8 Storage (T6.2)



4.8.1 Overall functional characterisation & Context

The Storage component is represented by the ZDMP platform data lake that hosts all ZDMP components' data, insuring their persistence and processing. This component acts as a central store of all enterprise data including back-up copies, machine learning models, reports, and anything else that needs to be accessed centrally. The data storage includes structured data from relational databases, semi-structured data as XML and JSON files, binary data as images and videos, as well as application components running as microservices.

4.8.2 Functions / Features

- **File Management**: This functionality uses Files Repository module that receives requests specific files management commands from Data Manager and serves the Binary Data Input API, Binary Data Output API and Cloud API, according to the security policies that user must have
- **Database Management**: This feature deals with both SQL and NoSQL databases hosted in Storage, which are used by most of components of ZDMP. It consists of receiving SQL commands and other specific data requests and retrieve records and structured data formats (ie JSON, BSON, XML) to users, according to data policies assigned. This feature is done by database commands received by Data Management module through Storage Frontend UI and input API and records and files input and output through specific database input and output API
- **Storage Administration**: This represents file repository and database administration commands sent through Storage Frontend UI

These functions can be further decomposed into the following subtasks that have to be realised:

Subtask	Subtask description
T62B001 Connect to file repository	**Priority:** Must
	Who: Storage **Where:** Anywhere **When:** Anytime **What:** Access and opens filesystem of the data source **Why:** So that a file transfer can be possible
Acceptance Criteria	Application connected successfully
Requirements filled	N/A
T62B002 Connect to SQL database	**Priority:** Must
	Who: Storage **Where:** Anywhere **When:** Anytime **What:** Connect to a SQL database **Why:** So that SQL operations can be run
Acceptance Criteria	Database successfully connected
Requirements filled	RQ_0015, RQ_0017, RQ_0020, RQ_0022, RQ_0023, RQ_0027, RQ_0030, RQ_0031, RQ_0032, RQ_0040, RQ_0085, RQ_0117, RQ_0119, RQ_0156, RQ_0157, RQ_0158, RQ_0160, RQ_0184, RQ_0185, RQ_0192, RQ_0214, RQ_0221, RQ_0249, RQ_0250, RQ_0251, RQ_0252, RQ_0255, RQ_0285, RQ_0288, RQ_0324, RQ_0381, RQ_0382, RQ_0383, RQ_0384, RQ_0385, RQ_0386, RQ_0412, RQ_0413, RQ_0414, RQ_0415, RQ_0416, RQ_0417, RQ_0418, RQ_0419, RQ_0420, RQ_0421, RQ_0422, RQ_0460, RQ_0461, RQ_0462, RQ_0463, RQ_0466, RQ_0467, RQ_0468, RQ_0515, RQ_0516, RQ_0517, RQ_0518, RQ_0519, RQ_0520, RQ_0521, RQ_0522, RQ_0523,

	RQ_0524, RQ_0525, RQ_0575, RQ_0576, RQ_0577, RQ_0578, RQ_0579, RQ_0580, RQ_0581, RQ_0582, RQ_0583, RQ_0584, RQ_0585, RQ_0623, RQ_0628, RQ_0629, RQ_0630, RQ_0631, RQ_0632, RQ_0633, RQ_0634, RQ_0635, RQ_0636, RQ_0637, RQ_0646, RQ_0647, RQ_0648, RQ_0649, RQ_0650, RQ_0651, RQ_0652, RQ_0653, RQ_0654, RQ_0655, RQ_0656, RQ_0657, RQ_0658, RQ_0659, RQ_0660, RQ_0681, RQ_0682, RQ_0683, RQ_0684, RQ_0685, RQ_0705, RQ_0707, RQ_0708, RQ_0709, RQ_0710, RQ_0711, RQ_0712, RQ_0713, RQ_0714, RQ_0715, RQ_0716, RQ_0717, RQ_0718, RQ_0719, RQ_0740, RQ_0741, RQ_0742, RQ_0743, RQ_0744, RQ_0745, RQ_0746, RQ_0747, RQ_0748, RQ_0749, RQ_0750, RQ_0751, RQ_0752, RQ_0753, RQ_0763, RQ_0764, RQ_0765, RQ_0766, RQ_0767, RQ_0769, RQ_0770, RQ_0771, RQ_0772, RQ_0773, RQ_0774, RQ_0775, RQ_0776, RQ_0777, RQ_0778, RQ_0794, RQ_0799, RQ_0801, RQ_0802, RQ_0803, RQ_0804, RQ_0805, RQ_0806, RQ_0807, RQ_0808, RQ_0809, RQ_0810, RQ_0811, RQ_0812, RQ_0814, RQ_0830, RQ_0831, RQ_0832, RQ_0833, RQ_0834, RQ_0835, RQ_0836, RQ_0837, RQ_0838, RQ_0851, RQ_0852, RQ_0853, RQ_0854, RQ_0855, RQ_0856, RQ_0857, RQ_0858, RQ_0859, RQ_0861, RQ_0862, RQ_0863, RQ_0865, RQ_0866, RQ_0870, RQ_0871, RQ_0875, RQ_0876, RQ_0877, RQ_0878, RQ_0889, RQ_0890, RQ_0891, RQ_0892, RQ_0893, RQ_0894, RQ_0906, RQ_0961, RQ_0962, RQ_0984, RQ_0985, RQ_0994
T62B003 Connect to NoSQL database	**Priority:** Must
	Who: Storage **Where:** Anywhere **When:** Anytime **What:** Connect to a NoSQL database **Why:** So that specific NoSQL database operations can be performed
Acceptance Criteria	Database successfully connected
Requirements filled	RQ_0015, RQ_0017, RQ_0020, RQ_0022, RQ_0023, RQ_0027, RQ_0030, RQ_0031, RQ_0032, RQ_0040, RQ_0085, RQ_0117, RQ_0119, RQ_0156, RQ_0157, RQ_0158, RQ_0160, RQ_0184, RQ_0185, RQ_0192, RQ_0214, RQ_0221, RQ_0249, RQ_0250, RQ_0251, RQ_0252, RQ_0255, RQ_0285, RQ_0288, RQ_0324, RQ_0381, RQ_0382, RQ_0383, RQ_0384, RQ_0385, RQ_0386, RQ_0412, RQ_0413, RQ_0414, RQ_0415, RQ_0416, RQ_0417, RQ_0418, RQ_0419, RQ_0420, RQ_0421, RQ_0422, RQ_0460, RQ_0461, RQ_0462, RQ_0463, RQ_0466, RQ_0467, RQ_0468, RQ_0515, RQ_0516, RQ_0517, RQ_0518, RQ_0519, RQ_0520, RQ_0521, RQ_0522, RQ_0523, RQ_0524, RQ_0525, RQ_0575, RQ_0576, RQ_0577, RQ_0578, RQ_0579, RQ_0580, RQ_0581, RQ_0582, RQ_0583, RQ_0584, RQ_0585, RQ_0623, RQ_0628, RQ_0629, RQ_0630, RQ_0631, RQ_0632, RQ_0633, RQ_0634, RQ_0635, RQ_0636, RQ_0637, RQ_0646, RQ_0647, RQ_0648, RQ_0649, RQ_0650, RQ_0651, RQ_0652, RQ_0653, RQ_0654, RQ_0655, RQ_0656, RQ_0657, RQ_0658, RQ_0659, RQ_0660, RQ_0681, RQ_0682, RQ_0683, RQ_0684, RQ_0685, RQ_0705, RQ_0707, RQ_0708, RQ_0709, RQ_0710, RQ_0711, RQ_0712, RQ_0713, RQ_0714, RQ_0715, RQ_0716, RQ_0717, RQ_0718, RQ_0719, RQ_0740, RQ_0741, RQ_0742, RQ_0743, RQ_0744, RQ_0745, RQ_0746, RQ_0747, RQ_0748, RQ_0749, RQ_0750, RQ_0751, RQ_0752, RQ_0753, RQ_0763, RQ_0764, RQ_0765, RQ_0766, RQ_0767, RQ_0769, RQ_0770, RQ_0771, RQ_0772, RQ_0773, RQ_0774, RQ_0775, RQ_0776, RQ_0777, RQ_0778, RQ_0794, RQ_0799, RQ_0801, RQ_0802, RQ_0803, RQ_0804, RQ_0805, RQ_0806, RQ_0807, RQ_0808, RQ_0809, RQ_0810, RQ_0811, RQ_0812, RQ_0814, RQ_0830, RQ_0831, RQ_0832, RQ_0833, RQ_0834, RQ_0835, RQ_0836, RQ_0837, RQ_0838, RQ_0851, RQ_0852, RQ_0853, RQ_0854, RQ_0855, RQ_0856, RQ_0857, RQ_0858, RQ_0859, RQ_0861, RQ_0862, RQ_0863, RQ_0865, RQ_0866, RQ_0870, RQ_0871, RQ_0875, RQ_0876, RQ_0877, RQ_0878, RQ_0889, RQ_0890, RQ_0891, RQ_0892, RQ_0893, RQ_0894, RQ_0906, RQ_0961, RQ_0962, RQ_0984, RQ_0985, RQ_0994
T62B004 Receive file command	**Priority:** Must
	Who: Storage **Where:** Anywhere **When:** Anytime **What:** Receive a valid file transfer command **Why:** So that a file operation can be performed
Acceptance Criteria	File command validated
Requirements filled	N/A
T62B005 Receive SQL command	**Priority:** Must
	Who: Storage **Where:** Anywhere **When:** Anytime **What:** Receive a valid SQL command. **Why:** So that a database operation can be executed.
Acceptance Criteria	SQL command validated.
Requirements filled	RQ_0015, RQ_0017, RQ_0020, RQ_0022, RQ_0023, RQ_0027, RQ_0030, RQ_0031, RQ_0032, RQ_0040, RQ_0085, RQ_0117, RQ_0119, RQ_0156, RQ_0157, RQ_0158, RQ_0160, RQ_0184, RQ_0185, RQ_0192, RQ_0214, RQ_0221, RQ_0249, RQ_0250, RQ_0251, RQ_0252, RQ_0255, RQ_0285, RQ_0288, RQ_0324, RQ_0381, RQ_0382, RQ_0383, RQ_0384, RQ_0385, RQ_0386, RQ_0412, RQ_0413, RQ_0414, RQ_0415, RQ_0416, RQ_0417, RQ_0418, RQ_0419, RQ_0420, RQ_0421, RQ_0422, RQ_0460, RQ_0461, RQ_0462, RQ_0463, RQ_0466, RQ_0467, RQ_0468, RQ_0515, RQ_0516, RQ_0517, RQ_0518, RQ_0519, RQ_0520, RQ_0521, RQ_0522, RQ_0523,

	RQ_0524, RQ_0525, RQ_0575, RQ_0576, RQ_0577, RQ_0578, RQ_0579, RQ_0580, RQ_0581, RQ_0582, RQ_0583, RQ_0584, RQ_0585, RQ_0623, RQ_0628, RQ_0629, RQ_0630, RQ_0631, RQ_0632, RQ_0633, RQ_0634, RQ_0635, RQ_0636, RQ_0637, RQ_0646, RQ_0647, RQ_0648, RQ_0649, RQ_0650, RQ_0651, RQ_0652, RQ_0653, RQ_0654, RQ_0655, RQ_0656, RQ_0657, RQ_0658, RQ_0659, RQ_0660, RQ_0681, RQ_0682, RQ_0683, RQ_0684, RQ_0685, RQ_0705, RQ_0707, RQ_0708, RQ_0709, RQ_0710, RQ_0711, RQ_0712, RQ_0713, RQ_0714, RQ_0715, RQ_0716, RQ_0717, RQ_0718, RQ_0719, RQ_0740, RQ_0741, RQ_0742, RQ_0743, RQ_0744, RQ_0745, RQ_0746, RQ_0747, RQ_0748, RQ_0749, RQ_0750, RQ_0751, RQ_0752, RQ_0753, RQ_0763, RQ_0764, RQ_0765, RQ_0766, RQ_0767, RQ_0769, RQ_0770, RQ_0771, RQ_0772, RQ_0773, RQ_0774, RQ_0775, RQ_0776, RQ_0777, RQ_0778, RQ_0794, RQ_0799, RQ_0801, RQ_0802, RQ_0803, RQ_0804, RQ_0805, RQ_0806, RQ_0807, RQ_0808, RQ_0809, RQ_0810, RQ_0811, RQ_0812, RQ_0814, RQ_0830, RQ_0831, RQ_0832, RQ_0833, RQ_0834, RQ_0835, RQ_0836, RQ_0837, RQ_0838, RQ_0851, RQ_0852, RQ_0853, RQ_0854, RQ_0855, RQ_0856, RQ_0857, RQ_0858, RQ_0859, RQ_0861, RQ_0862, RQ_0863, RQ_0865, RQ_0866, RQ_0870, RQ_0871, RQ_0875, RQ_0876, RQ_0877, RQ_0878, RQ_0889, RQ_0890, RQ_0891, RQ_0892, RQ_0893, RQ_0894, RQ_0906, RQ_0961, RQ_0962, RQ_0984, RQ_0985, RQ_0994
T62B006 Receive NoSQL command	**Priority:** Must
	Who: Storage **Where:** Anywhere **When:** Anytime **What:** Receive a valid NoSQL database command **Why:** So that a database operation can be executed
Acceptance Criteria	Database command validated
Requirements filled	RQ_0015, RQ_0017, RQ_0020, RQ_0022, RQ_0023, RQ_0027, RQ_0030, RQ_0031, RQ_0032, RQ_0040, RQ_0085, RQ_0117, RQ_0119, RQ_0156, RQ_0157, RQ_0158, RQ_0160, RQ_0184, RQ_0185, RQ_0192, RQ_0214, RQ_0221, RQ_0249, RQ_0250, RQ_0251, RQ_0252, RQ_0255, RQ_0285, RQ_0288, RQ_0324, RQ_0381, RQ_0382, RQ_0383, RQ_0384, RQ_0385, RQ_0386, RQ_0412, RQ_0413, RQ_0414, RQ_0415, RQ_0416, RQ_0417, RQ_0418, RQ_0419, RQ_0420, RQ_0421, RQ_0422, RQ_0460, RQ_0461, RQ_0462, RQ_0463, RQ_0466, RQ_0467, RQ_0468, RQ_0515, RQ_0516, RQ_0517, RQ_0518, RQ_0519, RQ_0520, RQ_0521, RQ_0522, RQ_0523, RQ_0524, RQ_0525, RQ_0575, RQ_0576, RQ_0577, RQ_0578, RQ_0579, RQ_0580, RQ_0581, RQ_0582, RQ_0583, RQ_0584, RQ_0585, RQ_0623, RQ_0628, RQ_0629, RQ_0630, RQ_0631, RQ_0632, RQ_0633, RQ_0634, RQ_0635, RQ_0636, RQ_0637, RQ_0646, RQ_0647, RQ_0648, RQ_0649, RQ_0650, RQ_0651, RQ_0652, RQ_0653, RQ_0654, RQ_0655, RQ_0656, RQ_0657, RQ_0658, RQ_0659, RQ_0660, RQ_0681, RQ_0682, RQ_0683, RQ_0684, RQ_0685, RQ_0705, RQ_0707, RQ_0708, RQ_0709, RQ_0710, RQ_0711, RQ_0712, RQ_0713, RQ_0714, RQ_0715, RQ_0716, RQ_0717, RQ_0718, RQ_0719, RQ_0740, RQ_0741, RQ_0742, RQ_0743, RQ_0744, RQ_0745, RQ_0746, RQ_0747, RQ_0748, RQ_0749, RQ_0750, RQ_0751, RQ_0752, RQ_0753, RQ_0763, RQ_0764, RQ_0765, RQ_0766, RQ_0767, RQ_0769, RQ_0770, RQ_0771, RQ_0772, RQ_0773, RQ_0774, RQ_0775, RQ_0776, RQ_0777, RQ_0778, RQ_0794, RQ_0799, RQ_0801, RQ_0802, RQ_0803, RQ_0804, RQ_0805, RQ_0806, RQ_0807, RQ_0808, RQ_0809, RQ_0810, RQ_0811, RQ_0812, RQ_0814, RQ_0830, RQ_0831, RQ_0832, RQ_0833, RQ_0834, RQ_0835, RQ_0836, RQ_0837, RQ_0838, RQ_0851, RQ_0852, RQ_0853, RQ_0854, RQ_0855, RQ_0856, RQ_0857, RQ_0858, RQ_0859, RQ_0861, RQ_0862, RQ_0863, RQ_0865, RQ_0866, RQ_0870, RQ_0871, RQ_0875, RQ_0876, RQ_0877, RQ_0878, RQ_0889, RQ_0890, RQ_0891, RQ_0892, RQ_0893, RQ_0894, RQ_0906, RQ_0961, RQ_0962, RQ_0984, RQ_0985, RQ_0994
T62B007 Send binary data	**Priority:** Must
	Who: Storage **Where:** Anywhere **When:** Anytime **What**: Perform a file transfer command **Why:** So that the user can get the result of his request
Acceptance Criteria	File successfully transferred
Requirements filled	RQ_0015, RQ_0017, RQ_0020, RQ_0022, RQ_0023, RQ_0027, RQ_0030, RQ_0031, RQ_0032, RQ_0040, RQ_0085, RQ_0117, RQ_0119, RQ_0156, RQ_0157, RQ_0158, RQ_0160, RQ_0184, RQ_0185, RQ_0192, RQ_0214, RQ_0221, RQ_0249, RQ_0250, RQ_0251, RQ_0252, RQ_0255, RQ_0285, RQ_0288, RQ_0324, RQ_0381, RQ_0382, RQ_0383, RQ_0384, RQ_0385, RQ_0386, RQ_0412, RQ_0413, RQ_0414, RQ_0415, RQ_0416, RQ_0417, RQ_0418, RQ_0419, RQ_0420, RQ_0421, RQ_0422, RQ_0460, RQ_0461, RQ_0462, RQ_0463, RQ_0466, RQ_0467, RQ_0468, RQ_0515, RQ_0516, RQ_0517, RQ_0518, RQ_0519, RQ_0520, RQ_0521, RQ_0522, RQ_0523, RQ_0524, RQ_0525, RQ_0575, RQ_0576, RQ_0577, RQ_0578, RQ_0579, RQ_0580, RQ_0581, RQ_0582, RQ_0583, RQ_0584, RQ_0585, RQ_0623, RQ_0628, RQ_0629, RQ_0630, RQ_0631, RQ_0632, RQ_0633, RQ_0634, RQ_0635, RQ_0636, RQ_0637, RQ_0646, RQ_0647, RQ_0648, RQ_0649, RQ_0650, RQ_0651, RQ_0652, RQ_0653, RQ_0654, RQ_0655, RQ_0656, RQ_0657, RQ_0658, RQ_0659, RQ_0660, RQ_0681, RQ_0682, RQ_0683, RQ_0684, RQ_0685, RQ_0705, RQ_0707, RQ_0708, RQ_0709, RQ_0710, RQ_0711, RQ_0712, RQ_0713, RQ_0714, RQ_0715, RQ_0716, RQ_0717, RQ_0718, RQ_0719, RQ_0740, RQ_0741, RQ_0742, RQ_0743, RQ_0744, RQ_0745, RQ_0746, RQ_0747, RQ_0748, RQ_0749, RQ_0750, RQ_0751, RQ_0752, RQ_0753, RQ_0763, RQ_0764, RQ_0765, RQ_0766, RQ_0767, RQ_0769, RQ_0770, RQ_0771, RQ_0772, RQ_0773, RQ_0774, RQ_0775, RQ_0776, RQ_0777, RQ_0778, RQ_0794, RQ_0799, RQ_0801,

	RQ_0802, RQ_0803, RQ_0804, RQ_0805, RQ_0806, RQ_0807, RQ_0808, RQ_0809, RQ_0810, RQ_0811, RQ_0812, RQ_0814, RQ_0830, RQ_0831, RQ_0832, RQ_0833, RQ_0834, RQ_0835, RQ_0836, RQ_0837, RQ_0838, RQ_0851, RQ_0852, RQ_0853, RQ_0854, RQ_0855, RQ_0856, RQ_0857, RQ_0858, RQ_0859, RQ_0861, RQ_0862, RQ_0863, RQ_0865, RQ_0866, RQ_0870, RQ_0871, RQ_0875, RQ_0876, RQ_0877, RQ_0878, RQ_0889, RQ_0890, RQ_0891, RQ_0892, RQ_0893, RQ_0894, RQ_0906, RQ_0961, RQ_0962, RQ_0984, RQ_0985, RQ_0994
T62B008 Send database records	**Priority:** Must
	Who: Storage **Where:** Anywhere **When:** Anytime **What:** Perform a database SQL command **Why:** So that the user can get the result of his request
Acceptance Criteria	SQL successfully executed
Requirements filled	RQ_0015, RQ_0017, RQ_0020, RQ_0022, RQ_0023, RQ_0027, RQ_0030, RQ_0031, RQ_0032, RQ_0040, RQ_0085, RQ_0117, RQ_0119, RQ_0156, RQ_0157, RQ_0158, RQ_0160, RQ_0184, RQ_0185, RQ_0192, RQ_0214, RQ_0221, RQ_0249, RQ_0250, RQ_0251, RQ_0252, RQ_0255, RQ_0285, RQ_0288, RQ_0324, RQ_0381, RQ_0382, RQ_0383, RQ_0384, RQ_0385, RQ_0386, RQ_0412, RQ_0413, RQ_0414, RQ_0415, RQ_0416, RQ_0417, RQ_0418, RQ_0419, RQ_0420, RQ_0421, RQ_0422, RQ_0460, RQ_0461, RQ_0462, RQ_0463, RQ_0466, RQ_0467, RQ_0468, RQ_0515, RQ_0516, RQ_0517, RQ_0518, RQ_0519, RQ_0520, RQ_0521, RQ_0522, RQ_0523, RQ_0524, RQ_0525, RQ_0575, RQ_0576, RQ_0577, RQ_0578, RQ_0579, RQ_0580, RQ_0581, RQ_0582, RQ_0583, RQ_0584, RQ_0585, RQ_0623, RQ_0628, RQ_0629, RQ_0630, RQ_0631, RQ_0632, RQ_0633, RQ_0634, RQ_0635, RQ_0636, RQ_0637, RQ_0646, RQ_0647, RQ_0648, RQ_0649, RQ_0650, RQ_0651, RQ_0652, RQ_0653, RQ_0654, RQ_0655, RQ_0656, RQ_0657, RQ_0658, RQ_0659, RQ_0660, RQ_0681, RQ_0682, RQ_0683, RQ_0684, RQ_0685, RQ_0705, RQ_0707, RQ_0708, RQ_0709, RQ_0710, RQ_0711, RQ_0712, RQ_0713, RQ_0714, RQ_0715, RQ_0716, RQ_0717, RQ_0718, RQ_0719, RQ_0740, RQ_0741, RQ_0742, RQ_0743, RQ_0744, RQ_0745, RQ_0746, RQ_0747, RQ_0748, RQ_0749, RQ_0750, RQ_0751, RQ_0752, RQ_0753, RQ_0763, RQ_0764, RQ_0765, RQ_0766, RQ_0767, RQ_0769, RQ_0770, RQ_0771, RQ_0772, RQ_0773, RQ_0774, RQ_0775, RQ_0776, RQ_0777, RQ_0778, RQ_0794, RQ_0799, RQ_0801, RQ_0802, RQ_0803, RQ_0804, RQ_0805, RQ_0806, RQ_0807, RQ_0808, RQ_0809, RQ_0810, RQ_0811, RQ_0812, RQ_0814, RQ_0830, RQ_0831, RQ_0832, RQ_0833, RQ_0834, RQ_0835, RQ_0836, RQ_0837, RQ_0838, RQ_0851, RQ_0852, RQ_0853, RQ_0854, RQ_0855, RQ_0856, RQ_0857, RQ_0858, RQ_0859, RQ_0861, RQ_0862, RQ_0863, RQ_0865, RQ_0866, RQ_0870, RQ_0871, RQ_0875, RQ_0876, RQ_0877, RQ_0878, RQ_0889, RQ_0890, RQ_0891, RQ_0892, RQ_0893, RQ_0894, RQ_0906, RQ_0961, RQ_0962, RQ_0984, RQ_0985, RQ_0994
T62B009 Send structured data	**Priority:** Must
	Who: Storage **Where:** Anywhere **When:** Anytime **What:** Perform a NoSQL command **Why:** So that the user can get the result of his request
Acceptance Criteria	Command successfully executed
Requirements filled	RQ_0015, RQ_0017, RQ_0020, RQ_0022, RQ_0023, RQ_0027, RQ_0030, RQ_0031, RQ_0032, RQ_0040, RQ_0085, RQ_0117, RQ_0119, RQ_0156, RQ_0157, RQ_0158, RQ_0160, RQ_0184, RQ_0185, RQ_0192, RQ_0214, RQ_0221, RQ_0249, RQ_0250, RQ_0251, RQ_0252, RQ_0255, RQ_0285, RQ_0288, RQ_0324, RQ_0381, RQ_0382, RQ_0383, RQ_0384, RQ_0385, RQ_0386, RQ_0412, RQ_0413, RQ_0414, RQ_0415, RQ_0416, RQ_0417, RQ_0418, RQ_0419, RQ_0420, RQ_0421, RQ_0422, RQ_0460, RQ_0461, RQ_0462, RQ_0463, RQ_0466, RQ_0467, RQ_0468, RQ_0515, RQ_0516, RQ_0517, RQ_0518, RQ_0519, RQ_0520, RQ_0521, RQ_0522, RQ_0523, RQ_0524, RQ_0525, RQ_0575, RQ_0576, RQ_0577, RQ_0578, RQ_0579, RQ_0580, RQ_0581, RQ_0582, RQ_0583, RQ_0584, RQ_0585, RQ_0623, RQ_0628, RQ_0629, RQ_0630, RQ_0631, RQ_0632, RQ_0633, RQ_0634, RQ_0635, RQ_0636, RQ_0637, RQ_0646, RQ_0647, RQ_0648, RQ_0649, RQ_0650, RQ_0651, RQ_0652, RQ_0653, RQ_0654, RQ_0655, RQ_0656, RQ_0657, RQ_0658, RQ_0659, RQ_0660, RQ_0681, RQ_0682, RQ_0683, RQ_0684, RQ_0685, RQ_0705, RQ_0707, RQ_0708, RQ_0709, RQ_0710, RQ_0711, RQ_0712, RQ_0713, RQ_0714, RQ_0715, RQ_0716, RQ_0717, RQ_0718, RQ_0719, RQ_0740, RQ_0741, RQ_0742, RQ_0743, RQ_0744, RQ_0745, RQ_0746, RQ_0747, RQ_0748, RQ_0749, RQ_0750, RQ_0751, RQ_0752, RQ_0753, RQ_0763, RQ_0764, RQ_0765, RQ_0766, RQ_0767, RQ_0769, RQ_0770, RQ_0771, RQ_0772, RQ_0773, RQ_0774, RQ_0775, RQ_0776, RQ_0777, RQ_0778, RQ_0794, RQ_0799, RQ_0801, RQ_0802, RQ_0803, RQ_0804, RQ_0805, RQ_0806, RQ_0807, RQ_0808, RQ_0809, RQ_0810, RQ_0811, RQ_0812, RQ_0814, RQ_0830, RQ_0831, RQ_0832, RQ_0833, RQ_0834, RQ_0835, RQ_0836, RQ_0837, RQ_0838, RQ_0851, RQ_0852, RQ_0853, RQ_0854, RQ_0855, RQ_0856, RQ_0857, RQ_0858, RQ_0859, RQ_0861, RQ_0862, RQ_0863, RQ_0865, RQ_0866, RQ_0870, RQ_0871, RQ_0875, RQ_0876, RQ_0877, RQ_0878, RQ_0889, RQ_0890, RQ_0891, RQ_0892, RQ_0893, RQ_0894, RQ_0906, RQ_0961, RQ_0962, RQ_0984, RQ_0985, RQ_0994
T62B010	**Priority:** Must

Receive Management command	**Who:** Storage **Where:** Anywhere **When:** Anytime **What:** Receive a valid management command **Why:** So that a management operation can be executed
Acceptance Criteria	Command validated
Requirements filled	N/A
T62B011 Send Management action	**Priority:** Must
	Who: Storage **Where:** Anywhere **When:** Anytime **What:** Execute a management command **Why:** So that the user can get the result of his request
Acceptance Criteria	Command successfully executed
Requirements filled	N/A

Figure 102: Storage Functions

4.8.3 Workflows

The following sub-sections describe the sequence diagrams of the Storage component.

4.8.3.1 Connect to file repository

The following diagram explains this function and the necessary interactions with other components.

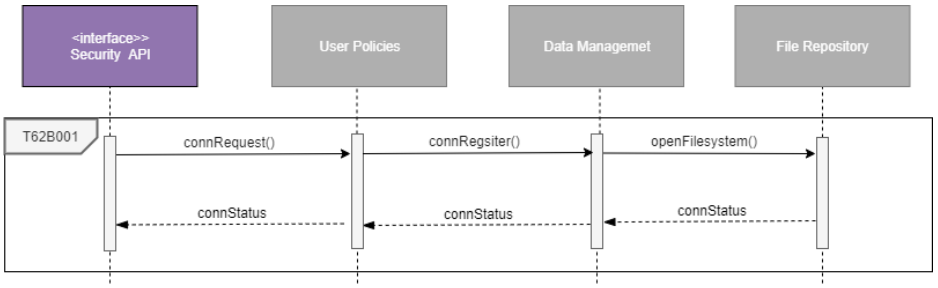


Figure 103: Connect to file repository sequence diagram

4.8.3.2 Connect to SQL Database

The following diagram explains this function and the necessary interactions with other components.

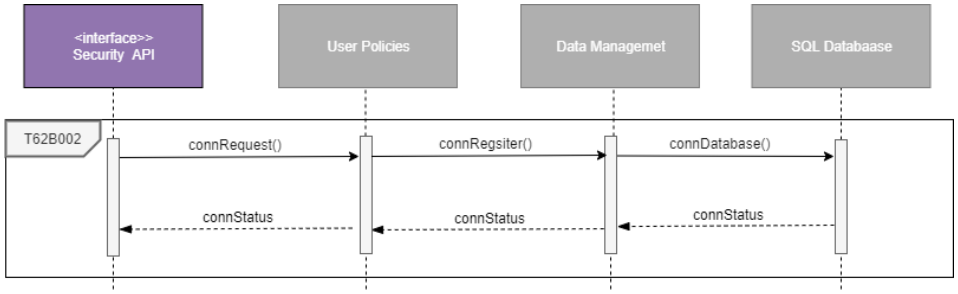


Figure 104: Connect to SQL database sequence diagram

4.8.3.3 Connect to NoSQL Database

The following diagram explains this function and the necessary interactions with other components.

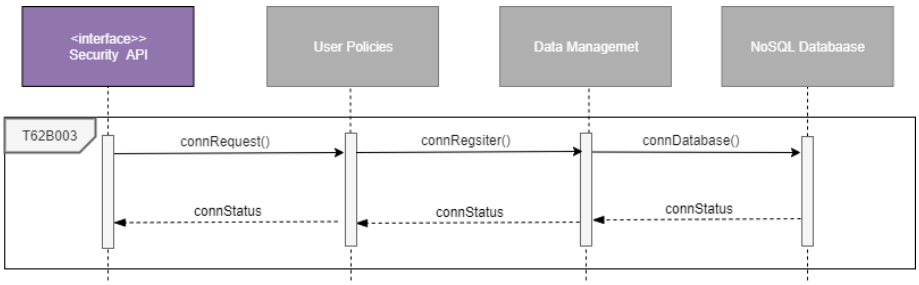


Figure 105: Connect to NoSQL database sequence diagram

4.8.3.4 Receive file command and send binary data

The following diagram explains this function and the necessary interactions with other components.

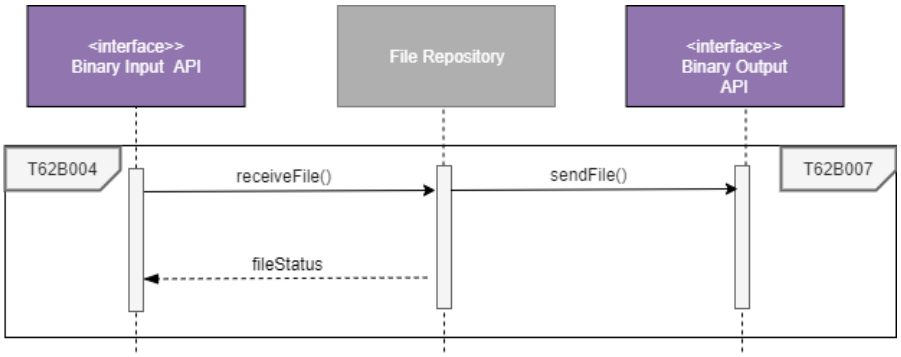


Figure 106: Receive file command and send binary data sequence diagram

4.8.3.5 Receive SQL commands and send records

The following diagram explains this function and the necessary interactions with other components.

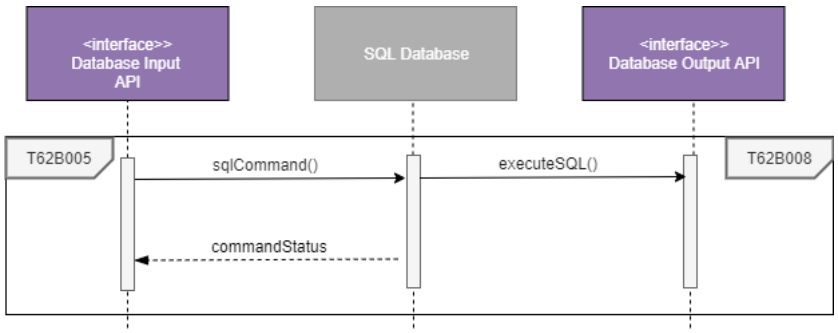


Figure 107: Receive SQL command and send structured data sequence diagram

4.8.3.6 Receive NoSQL commands and send structured data

The following diagram explains this function and the necessary interactions with other components.

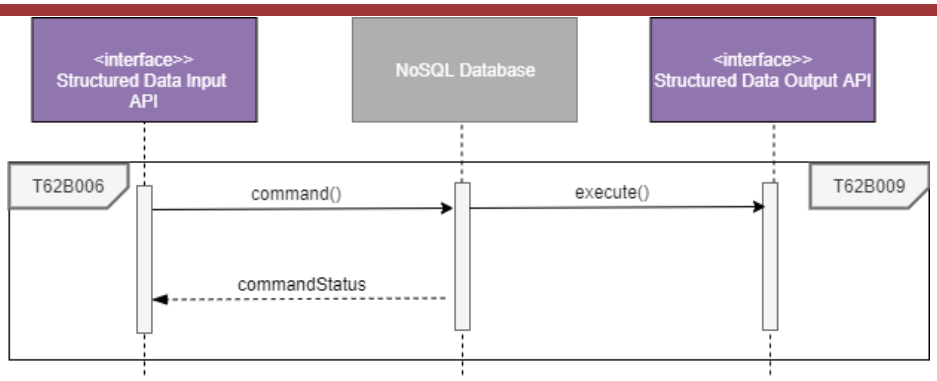


Figure 108: Receive NoSQL command and send structured data sequence diagram

4.8.3.7 Receive management command and execute actions

The following diagram explains this function and the necessary interactions with other components.

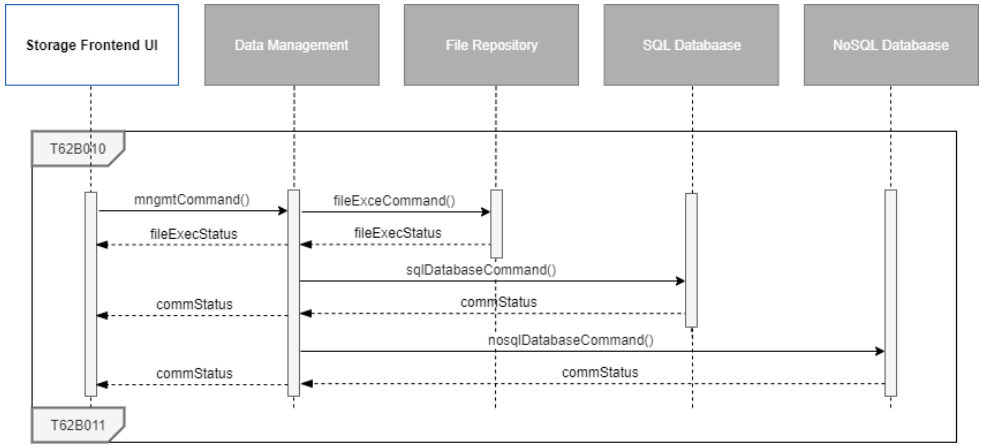
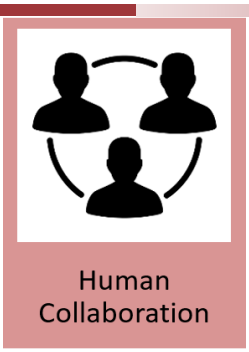


Figure 109: Receive management command and execute action sequence diagram

4.9 Human Collaboration (T6.3)



4.9.1 Overall functional characterisation & Context

The Human Collaboration component aims to facilitate the relationship between human users and manufacturing assets by providing holistic information and usage description in digital format. The collaboration environment may also include external developers and users who can request services and use communication platforms such as audio and video streaming, forums, workshops, and hackathons. The collaboration aspect is performed using multiple media channels (eg VoIP, Video Stream) and is also able to request service-related data for assets and tools being stored as services and visualised on the factory map.

4.9.2 Functions / Features

- **Browse Content**: the users/developers can browse and download zApps from the Marketplace, exchange service information, and images
- **Streaming**: users and developers can participate in audio and video calls. The stream manager component manages video content from a Video Streaming and allow user to watch videos and to participate in video conferences
- **Collaboration**: this functionality uses Collaboration Manager module, connected to Collaboration API, which allows users to get context information, exchange service information and forward the images and other data to central storage of the component which is Data Manager module
- **Assets location**: due to Location Based Information Manager module, the users can visualise the factory map and can get the technical data and by having location information of the assets by connecting the Physical and IoT-related data sources
- **Notifications**: the users and developers can get and send notifications about assets and context information due to Notification Manager and Notification API components, the last one being connected to External Notification Services

These functions can be further decomposed into the following subtasks that have to be realised:

Subtask	Subtask description
T63A001 Connect to Marketplace	**Priority:** Must
	Who: Human Collaboration **Where:** Anywhere **When:** Anytime **What:** Connect to Marketplace and open zApps page **Why:** So that the assets can be searched and downloaded
Acceptance Criteria	Application connected successfully
Requirements filled	None
T63A002 Search content	**Priority:** Must
	Who: Human Collaboration **Where:** Anywhere **When:** Anytime **What:** Browse and search for specific zApps from the Marketplace **Why:** So that the desired zApp can be found
Acceptance Criteria	zApp successfully found
Requirements filled	None
T63A003	**Priority:** Must

Download content	**Who:** Human Collaboration **Where:** Anywhere **When:** Anytime **What:** Download a specific zApp from the Marketplace **Why:** So that the zApp can be stored in Data Manager for further installation
Acceptance Criteria	zApp successfully downloaded
Requirements filled	None
T63A004 Send information	**Priority:** Must
	Who: Human Collaboration **Where:** Anywhere **When:** Anytime **What:** Send specific information regarding an asset **Why:** So that user can receive the information he requested
Acceptance Criteria	Information completed and sent successfully
Requirements filled	RQ_0014, RQ_0029, RQ_0108, RQ_0109, RQ_0145, RQ_0148, RQ_0149, RQ_0868
T63A005 Receive information	**Priority:** Must
	Who: Human Collaboration **Where:** Anywhere **When:** Anytime **What:** Receive specific information on request **Why:** So that the user can read the information he needed for a specific asset
Acceptance Criteria	Information received successfully
Requirements filled	RQ_0014, RQ_0029, RQ_0108, RQ_0109, RQ_0145, RQ_0148, RQ_0149, RQ_0868
T63A006 Upload images	**Priority:** Must
	Who: Human Collaboration **Where:** Anywhere **When:** Anytime **What:** Upload specific image regarding an asset **Why:** So that user can receive the information he requested
Acceptance Criteria	Image uploaded successfully
Requirements filled	None
T63A007 Download images	**Priority:** Must
	Who: Human Collaboration **Where:** Anywhere **When:** Anytime **What**: Download specific image **Why:** So that the user can have the asset image needed
Acceptance Criteria	Image downloaded successfully displayed
Requirements filled	None
T63A008 Upload video	**Priority:** Must
	Who: Human Collaboration **Where:** Anywhere **When:** Anytime **What:** Store a video using Stream Manager and open a link for users **Why:** So that the users can access and watch video
Acceptance Criteria	Video successfully uploaded
Requirements filled	None
T63A009 Join video conference	**Priority:** Must
	Who: Human Collaboration **Where:** Anywhere **When:** Anytime **What:** Create/join video conference **Why:** So that the user can participate in video calls
Acceptance Criteria	Video conference successfully created

Requirements filled	None
T63A010 Get location	**Priority:** Must
	Who: Human Collaboration **Where:** Anywhere **When:** Anytime **What:** Connect to asset and get its location **Why:** So that the user can visualise asset in the factory map
Acceptance Criteria	Asset successfully located
Requirements filled	None

4.9.3 Workflows

The following sub-sections describe the sequence diagrams of the Human Collaboration component.

4.9.3.1 Connect to Marketplace

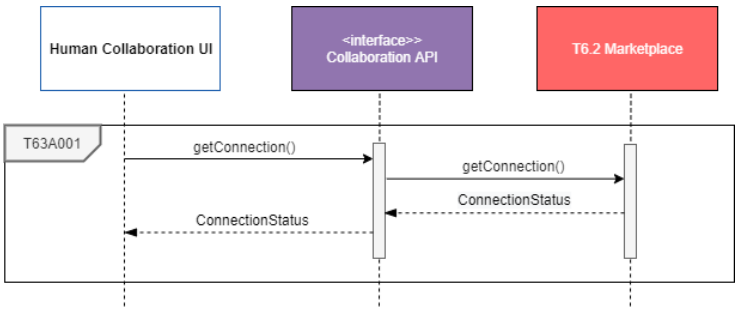


Figure 110 Connect to Marketplace sequence diagram

4.9.3.2 Search and download content

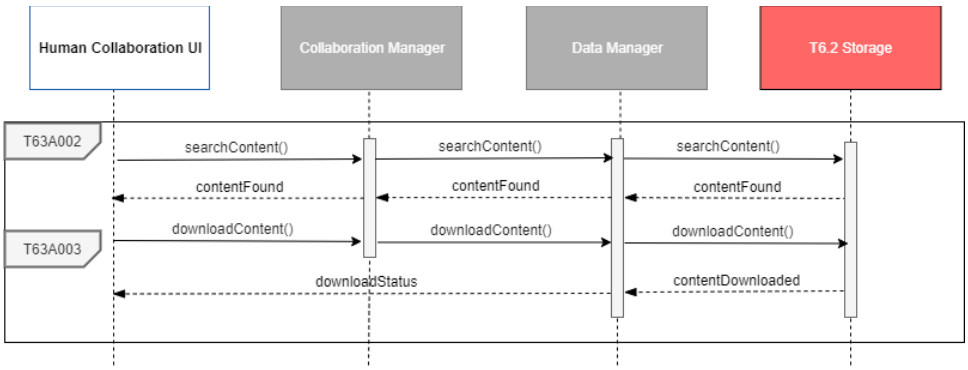


Figure 3 Search and download content sequence diagram

4.9.3.3 Send and receive information

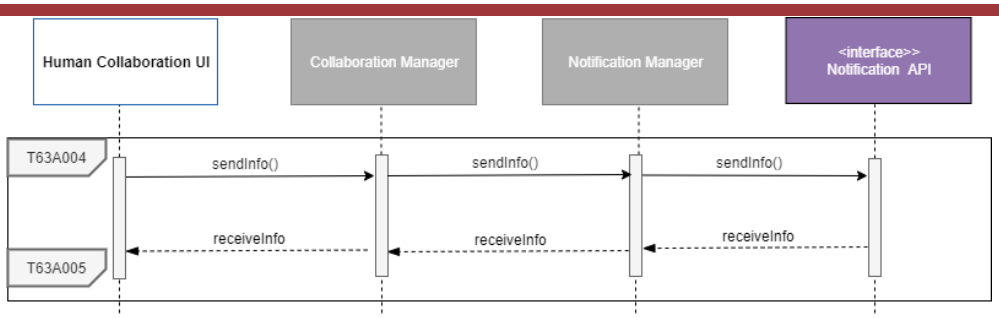


Figure 4 Send and receive information sequence diagram

4.9.3.4 Upload and download images

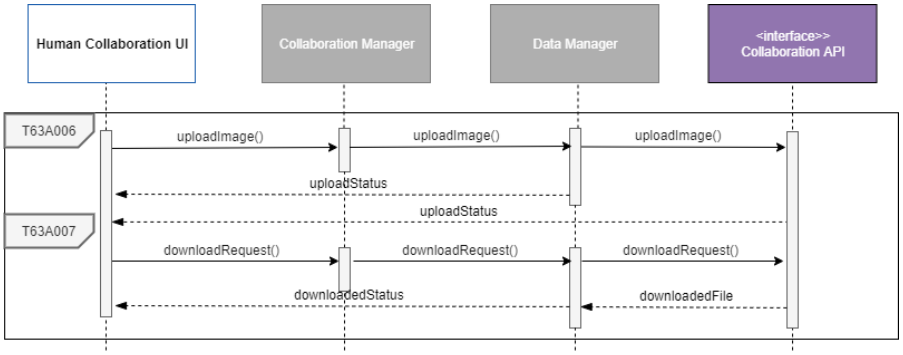


Figure 4 Upload and download images sequence diagram

4.9.3.5 Upload video

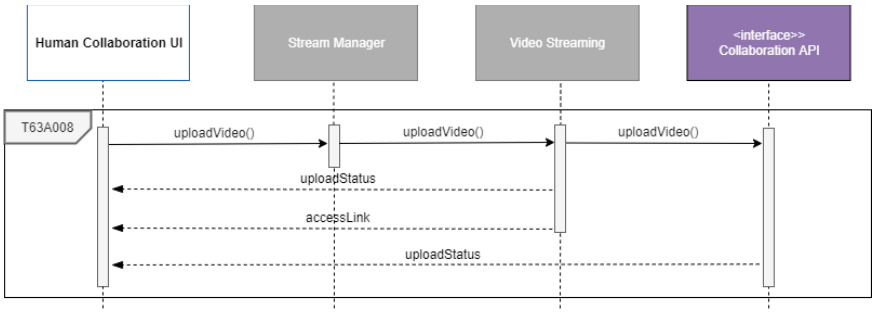


Figure 4 Upload video sequence diagram

4.9.3.6 Join video conference

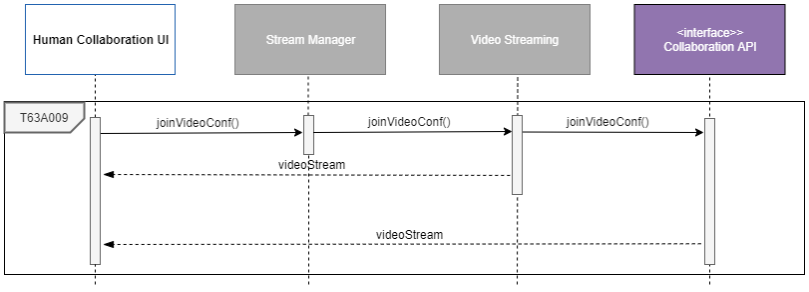


Figure 4 Join video conference sequence diagram

4.9.3.7 Get location

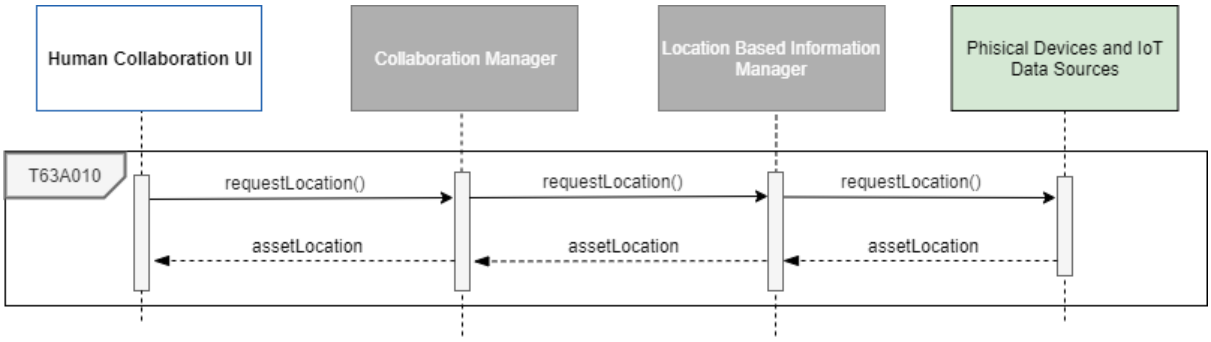
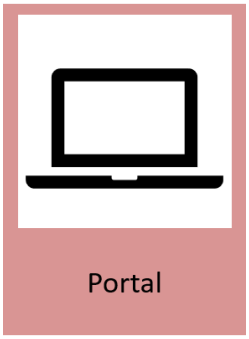


Figure 4 Get location sequence diagram

4.10 Portal (T6.4)

4.10.1 Overall functional characterisation & Context



This is the front facing interface to the platform. From here user can login to the platform see available applications (depending on their access). This portal allows access to the zApps UIs and creates a standard point for addressing the system.

This component is on the Enterprise tier and is part of “user frontend plane”. It needs to allow people access to various parts of the platform including the T6.2 the Marketplace to buy and install zApps as well as connecting to running applications within the T6.4 Application Run-time.

4.10.2 Functions / Features

The main function of this is to function as the front facing portal to the project. Including users, zApps, setup and configuration, and links to the marketplace.

Subtask	Subtask description
T64D01 Manage users	**Priority:** Must
	Who: Users **What**: To login and manage their page and anything they have permission to **Why:** For the functioning of the platform and the centralisation of administrative functions **When:** A user or administrator access requires it **Where:** On the platform
Acceptance Criteria	A user can login and see their profile
Requirements filled	RQ_0423, RQ_0442, RQ_0443, RQ_0444, RQ_0469, RQ_0486, RQ_0487, RQ_0488, RQ_0526, RQ_0546, RQ_0547, RQ_0548, RQ_0586, RQ_0605, RQ_0606, RQ_0607, RQ_0615, RQ_0616, RQ_0619, RQ_0620, RQ_0621, RQ_0668, RQ_0669, RQ_0672, RQ_0673, RQ_0674, RQ_0676, RQ_0727, RQ_0728, RQ_0731, RQ_0732, RQ_0733, RQ_0786, RQ_0787, RQ_0790, RQ_0791, RQ_0792, RQ_0794, RQ_0843, RQ_0844, RQ_0845, RQ_0847, RQ_0848, RQ_0849, RQ_0868, RQ_0869, RQ_0899, RQ_0900, RQ_0901, RQ_0903, RQ_0904, RQ_0905, RQ_0924, RQ_0980, RQ_0981, RQ_0982, RQ_0983
T64D02 Show appropriate content for a user’s role	**Who**: Users **What**: To show appropriate content based on their role. So, a developer receives different content compared to factory worker or to a factory manager **Why:** To distinguish the user roles and customise content effectively **When:** A user logs in to the portal **Where:** On the platform
Acceptance Criteria	Different user types can login and receive only appropriate
Requirements filled	N/A
T64D03 Getting a zApp UI and content	**Who**: User **What**: Requests access to a zApp UI **Why:** To centralise access to zApp UIs **When:** A user requests access to zApp UI, during runtime **Where:** On the platform
Acceptance Criteria	A user can access zApps from the portal
Requirements filled	N/A
T64D04 Managing zApps	**Who**: Administrator **What**: Can manage (request install or uninstall) of zApps. **Why:** To allow centralised access to zApp management **When:** At the request of the administrator **Where:** On the platform
Acceptance Criteria	An admin can manage zApps from the portal
Requirements filled	N/A

T64D05 Getting marketplace content	**Who**: User **What**: A user wishes to buy a zApp for this platform can get access to the marketplace features **Why:** To act a central interface for this instance of ZDMP **When:** When a user requests it **Where:** On the platform but to the central marketplace
Acceptance Criteria	A user can purchase a zApp for this platform
Requirements filled	N/A

Figure 111: Portal Features

4.10.3 Workflows

4.10.3.1 Show appropriate content for a user's role

As the front facing component to ZDMP it needs to respond differently for different users. It does this by utilising:

- User initiated requests
- Portal passes requests through the backend to the role's manager
- A connection to T5.2 Secure Authentication/Authorisation component gets the authentication of the user

The following figure shows the workflow to show appropriate content for any user.

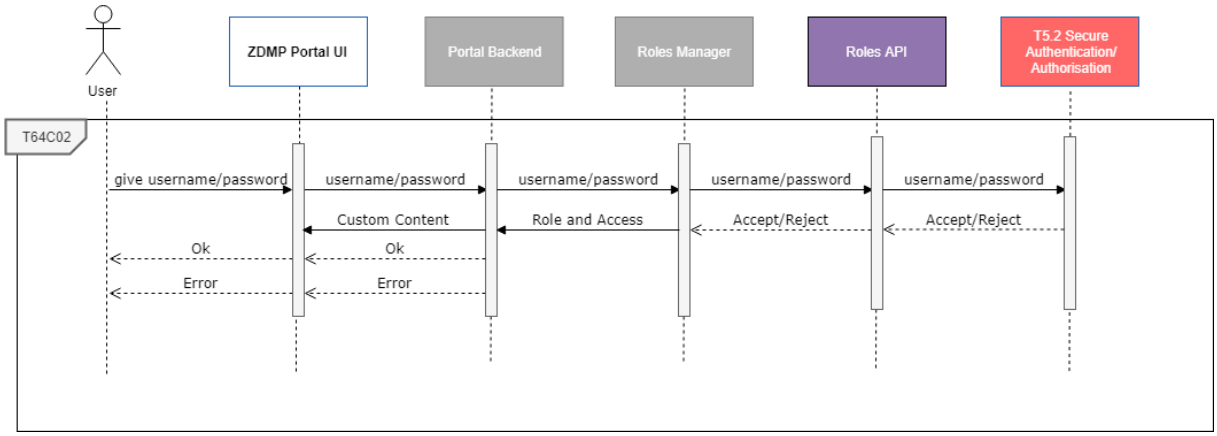


Figure 112: Workflow to get content for a user

4.10.3.2 Getting zApp UIs and content

The frontend portal also provides a central place to inspect the zApps. To access the UI from these zApps the portal redirects requests from the user to the ZDMP Asset UI API which will then get the UI content from the zApp. The following figure shows the workflow to get a zApp UI for a user.

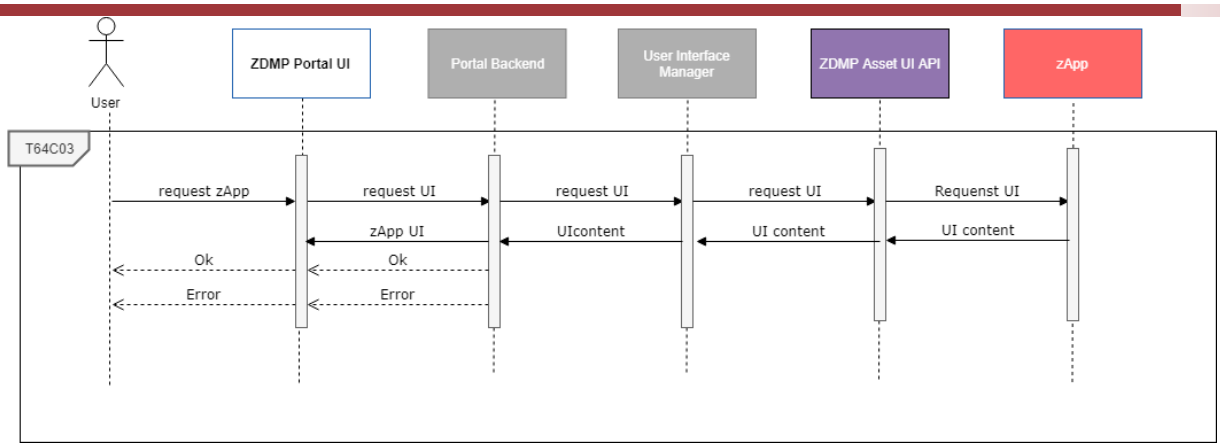


Figure 113: Workflow to get a zApp UI for a user

4.10.3.3 Get Marketplace Content

This will allow a user to access marketplace functionality from the portal (given appropriate permissions). The following figure shows the workflow to get marketplace content for a user.

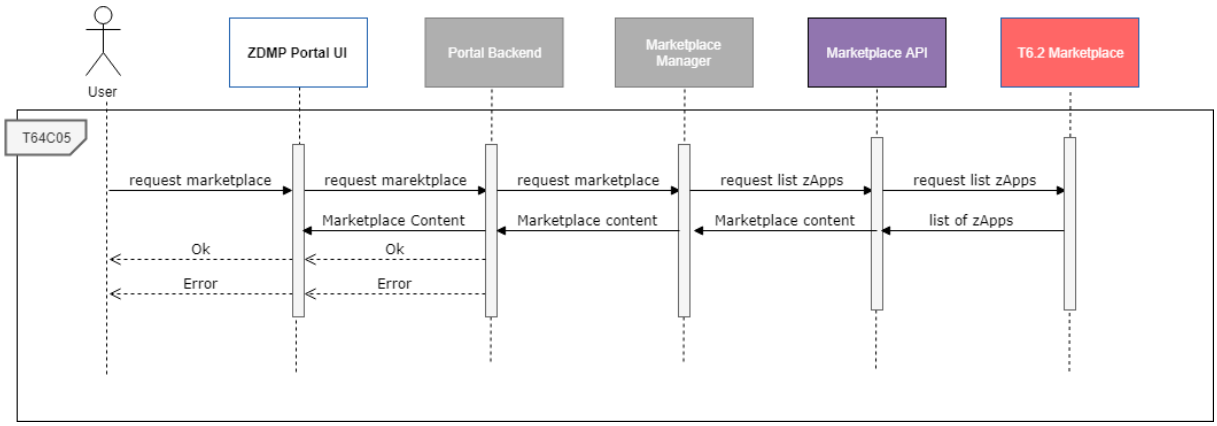


Figure 114: Workflow to get marketplace content for a user

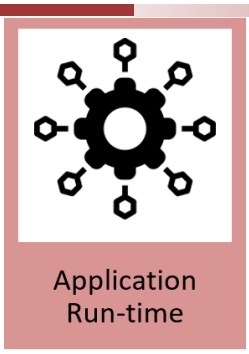
4.10.4 Additional Issues

Additional issues have appeared after the description of the architecture.

Issue	Description	Next Steps	Lead (Rationale)
Language requirements	Many use-cases have asked for multilanguage support. This was noted to be not the aim of the project. RQ_0162, RQ_0204, RQ_0220, RQ_0388, RQ_0446, RQ_0490, RQ_0550, RQ_0609, RQ_0663, RQ_0721, RQ_0781, RQ_0896, RQ_0954, RQ_0974, RQ_0840	Make sure our UI is prepared for internalisation but without translation.	T6.4 Portal
User roles	Many requirements relate to user roles. For central platform management this will be the interface. However every zApp will be able to access appropriate user permissions from T5.2 Secure Authentication and Authorisation RQ_0423, RQ_0442, RQ_0443, RQ_0444, RQ_0469, RQ_0486, RQ_0487, RQ_0488, RQ_0526, RQ_0546, RQ_0547, RQ_0548, RQ_0586, RQ_0605, RQ_0606, RQ_0607, RQ_0615, RQ_0616, RQ_0619, RQ_0620,	T5.2 Secure Authentication and Authorisation will be reasonable for users and roles the portal will support with a	T5.2 Secure Authentication and Authorisation

	RQ_0621, RQ_0668, RQ_0669, RQ_0672, RQ_0673, RQ_0674, RQ_0676, RQ_0727, RQ_0728, RQ_0731, RQ_0732, RQ_0733, RQ_0786, RQ_0787, RQ_0790, RQ_0791, RQ_0792, RQ_0794, RQ_0843, RQ_0844, RQ_0845, RQ_0847, RQ_0848, RQ_0849, RQ_0868, RQ_0869, RQ_0899, RQ_0900, RQ_0901, RQ_0903, RQ_0904, RQ_0905, RQ_0924, RQ_0980, RQ_0981, RQ_0982, RQ_0983	frontend interface.	

Figure 115: Additional Issues Portal



4.11 Application Run-time (T6.4)

4.11.1 Overall functional characterisation & Context

Application Run-time represents the running environment for zApps, components, and ZDMP Assets in general. It utilizes zApps, ZDMP Assets and core components as containerized applications with a RESTFUL interface.

Unlike many other components there is only one Application Run-time per platform instance. This component is part of the Enterprise tier, and is “communication, storage or management related infrastructure”. This component can be run either in the Cloud or servers in the factory (FOG).

4.11.2 Functions / Features

Functional this component allows for installing of a component, running of components, and allowing components to interact with each other:

- **Getting an application from the Marketplace or the Developer Tier:** Applications will be built in the design tier and uploaded to the marketplace or directly to the T6.4 Application run-time. Though the T5.2 Secure Installation component will be used to check that the Applications can be installed
- **Deploying/Installing an application:** Once an application has been purchased from the Marketplace or been uploaded from the Developer Tier then the application needs placing on the portal. Here it will be placed in a service registry
- **Running an application:** Applications are then deployed as per instructions in their manifest file. This will allow them to be deployed locally to the platform or at the “edge” (by using T5.5 Distributed Computing)
- **Running multiple applications:** The platform supports running multiple applications as services therefore the applications are deployed to allow for multiple instances of the applications and to be connected using the T6.4 Service and Message Bus

Subtask	Subtask description
T64C01 install a new component or zApp	**Priority:** Must
	Who: Manufacturing and Logistics User - IT Manager **What**: Install a new component or zApp on any available host, including all dependencies, in the correct location **Why:** To add new functionality for users to the platform **When:** A user requests it **Where:** On the platform
Acceptance Criteria	Successfully install a zApp from the marketplace, or a Developer Tier component
Requirements filled	RQ_0259, RQ_0332, RQ_0351, RQ_0362, RQ_0371, RQ_0380, RQ_0296, RQ_0300, RQ_0341, RQ_0392, RQ_0394, RQ_0450, RQ_0452, RQ_0554, RQ_0556, RQ_0614, RQ_0231, RQ_0495
T64C02	**Priority:** Should

manage installed components and zApps	**Who:** Manufacturing and Logistics User - IT Manager **What:** Use basic management functions for installed components and zApps: view the list of installed components and zApps, update the settings of a component or zApp, or remove installed components or zApps **Why:** To be able to maintain the functionalities provided by the platform to users **When:** A user requests it **Where:** On the platform
Acceptance Criteria	After changes have been confirmed, they are immediately considered. Dependencies of components / zApps need to be removed explicitly and separately
Requirements filled	RQ_0058, RQ_0111
T64C03 **manage component or zApp settings**	**Priority:** Must
	Who: Manufacturing and Logistics User - IT Manager **What:** View and edit the settings for a component or zApp, including its access to other components and zApps, as well as its runtime state (activated/deactivated, start/stop) **Why:** To alter the configuration of a certain component or zApp and to enable or disable its usage **When:** A user requests it **Where:** On the platform
Acceptance Criteria	After changes in the settings have been confirmed, they are immediately enacted.
Requirements filled	N/A
T64C05 **Networking a zApp**	**Priority:** Must
	Who: Service Registry and Distributed Computing **What**: connect endpoints and location to host zApp **Why:** so that a zApp has connectivity to the rest of the platform **When:** a zApp is installed **Where:** On the platform
Acceptance Criteria	A zApp can be networked in with the platform
Requirements filled	RQ_0023, RQ_0167, RQ_0168
T64C06 **Instantiating a zApp**	**Priority:** Must
	Who: Container Orchestrator, zApp, user **What:** Starts a zApp on a host **Why:** So, the zApp service can be up and running **When:** A user, zApp, or orchestrator requests it **Where:** On the platform
Acceptance Criteria	A zApp running on a host
Requirements filled	N/A

Figure 116: Application Runtime Features

4.11.3 Workflows

This component works to install any components or service associated with ZDMP or zApps (in the most general term)

4.11.3.1 Installing a zApp

Once an zApp has been purchased from the Marketplace it will be installed onto the platform using T5.2 Secure Installation component. Figure 117 shows this workflow. The workflow consists of the following steps:

- Receive request to install a zApp
- Read manifest file and

- Add docker file to registry

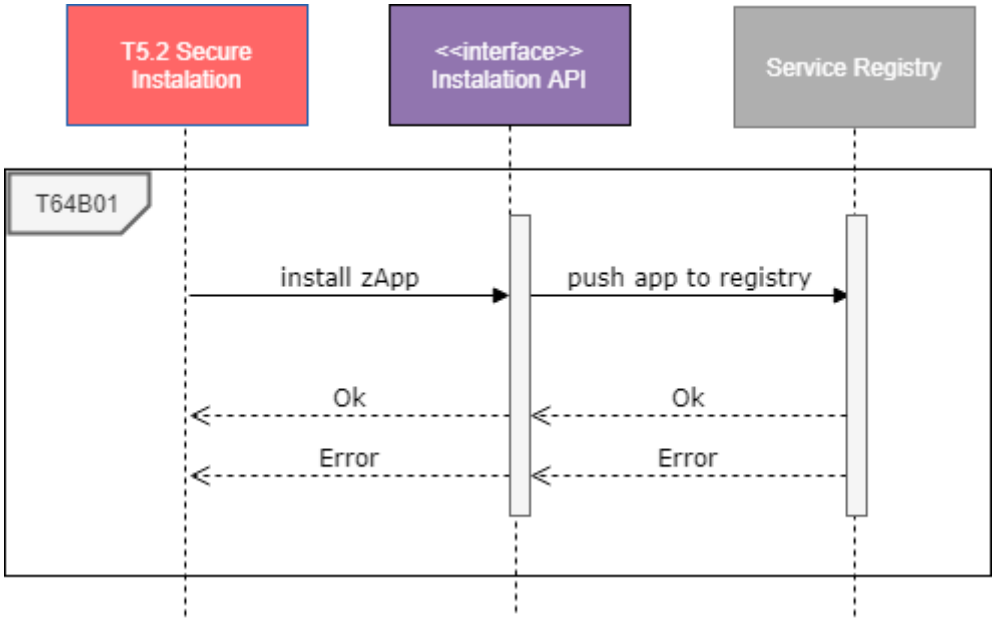


Figure 117: The workflow diagram describing how to install a zApp

4.11.3.2 Networking a zApp

The zApps endpoints and host need to be assigned. The endpoints are stored in the manifest file and the host will be defined by the T5.5 Distributed Computing component. This workflow is shown in Figure 118.

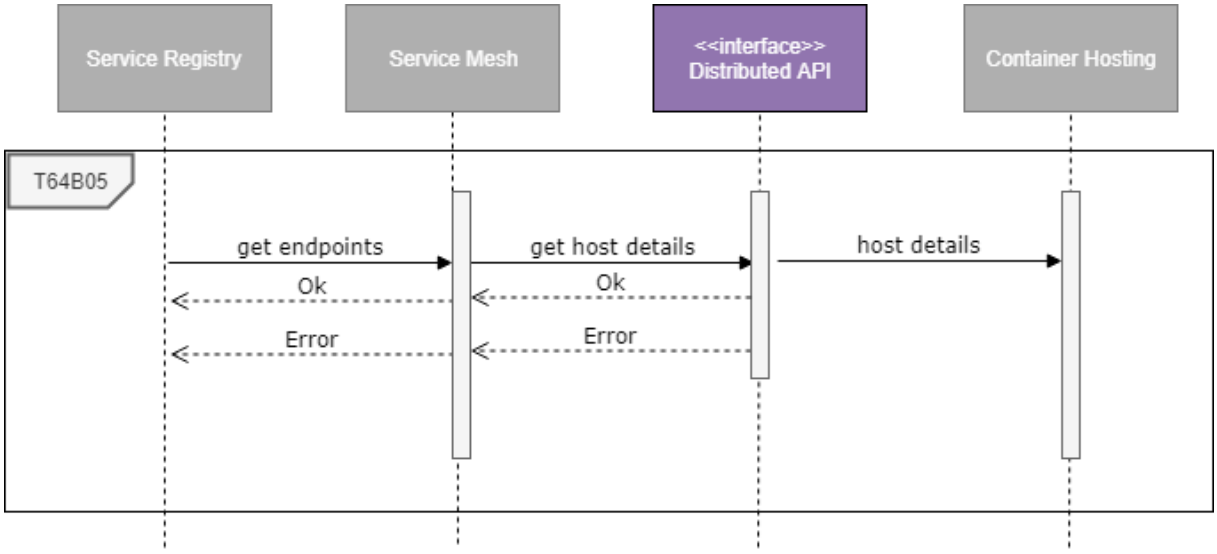


Figure 118: Workflow diagram describing how to network a zApp

4.11.3.3 Instantiating a zApp

The container orchestration module will start (instantiate) a zApp. This deploy the Docker Image as a container, as shown in Figure 119.

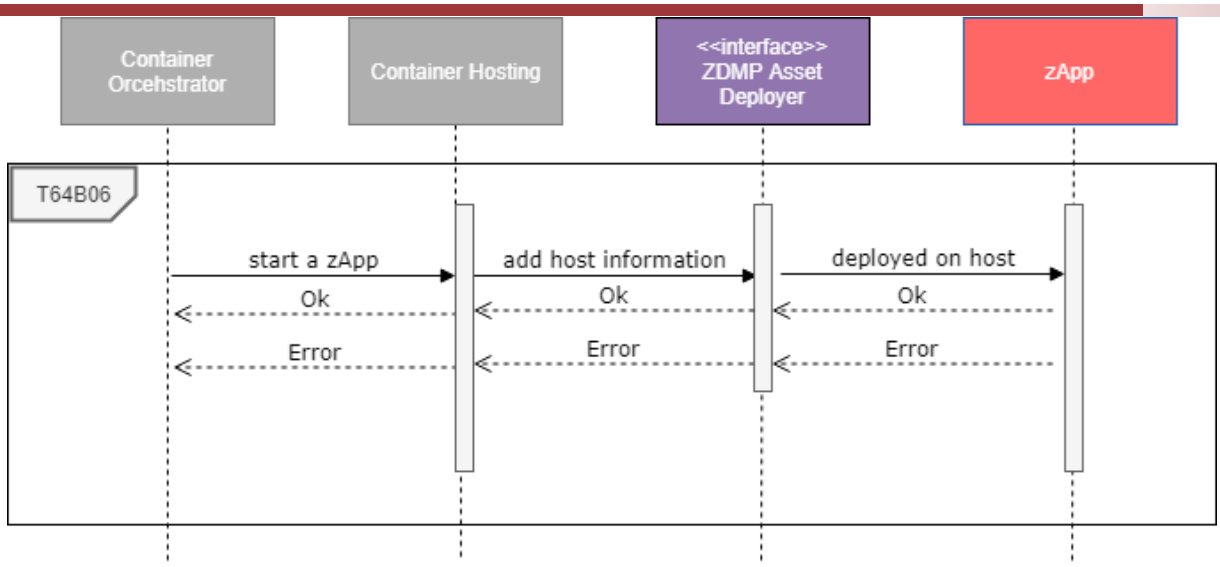


Figure 119: The workflow diagram explaining how to instantiate a zApp

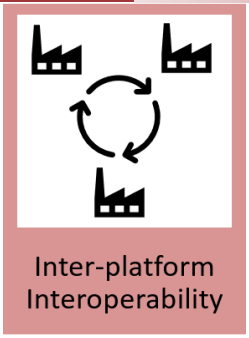
4.11.4 Additional Issues

Additional issues have appeared after the description of the architecture.

Issue	Description	Next Steps	Lead (Rationale)
General Requirement of the Platform	This component represents a core component of ZDMP and can be used by a wide range of zApps, and other components and is required for a working project.	Clarification of the general usage of ZDMP	T6.4 Application Run-time
Requirements - Performance	The following requirements mention the performance of the platform. This will be considered by this component and is related to all the functionality of the running platform. RQ_0349, RQ_0018, RQ_0019, RQ_0025, RQ_0036, RQ_0082, RQ_0090, RQ_0104, RQ_0110, RQ_0132, RQ_0144, RQ_0150, RQ_0257, RQ_0278, RQ_0279, RQ_0339, RQ_0360, RQ_0369, RQ_0378, RQ_0386, RQ_0451, RQ_0555	Develop this component to help with these requirements	T6.4 Application Run-time
Requirements – Location of Platform	The following requirements mention the location of the platform. This will be considered by this component but cannot be related to the functionality. RQ_0002, RQ_0016, RQ_0059, RQ_0060, RQ_0092, RQ_0112, RQ_0134, RQ_0666, RQ_0726, RQ_0785	Develop this component to help with these requirements	T6.4 Application Run-time
Platform Installation	RQ_0227 represents a requirement on the installation of this component. (Requiring Windows 7 and 10)	Make sure this component satisfies this	T6.4 Application Run-time

Figure 120: Additional Issues Application Runtime

4.12 Inter-platform Interoperability (T6.5)



This task addresses the interconnectivity of the ZDMP platform with other 3rd party platforms and with other instances of ZDMP. It connects three main themes, security, data, and marketplaces to allow the secure transfer of information and services to other platforms. The following platforms are considered as candidate platform for integration:

- **eFactory**: A federated smart factory ecosystem and a digital platform that interlink different stakeholders of the digital manufacturing domain
- **ADAMOS:** Open, manufacturer-independent IIoT platform run by SAG, designed by a joint-venture of leading engineering and technology companies. The module will work as backend module based on APIs.
- **SDAIM:** The system was originally developed as a surveillance, detection and alerts information management system designed for geo-distributed and federated near-real-time monitoring and decision support. Nevertheless, as the SDAIM design is based on a multi-level data and information fusion model, it is a generic Streaming Data Analytics and Information Management platform
- **Other** – Other platforms that may be considered

4.12.1 Overall Functional Characterisation & Context

This component acts functionally as an interconnector between an instance of the ZDMP platform and other platforms. It first establishes and allows for secure connections with the external platform then proceeds to share information and services with the other platform. It shares data from the T6.4 Service and Message Bus and shares services from the T6.2 Marketplace.

This components functionality is in the Enterprise Tier of the platform (see D4.3a Global Architecture and Specification). It connects to other applications, directly or in the Cloud. It needs secure access to the data and zApps available in the platform and provides communication, storage, or management related infrastructure.

4.12.2 Functions / Features

The Inter-platform Interoperability Component is split into several major sub-components which offer the following major functionalities:

- **Interoperability UI**: This is a user interface to manage connections with other platforms and giving the appropriate permissions needed for a connection, as managed by the security API and Security Standardisation
- **Interoperability Centre**: This controls the whole component and directs flow of information and services as designated by the user
- **Security Standardisation**: This feature enables the T5.2 Secure Authentication/Authorisation component to check permissions of users and make sure connections are valid across different role and access models used by the interconnected external platforms
- **Security API and Security Standardisation**: This takes links with the T5.2 Secure Authentication/Authorisation component to check permissions of users and make sure connections are valid

- **Marketplace Standardisation**: This feature allows the integration of external marketplaces into the ZDMP platform and enables external platforms to access the ZDMP Marketplace to give and receive links to available services
- **Marketplace API and Marketplace Standardisation**: This links with the T6.2 Marketplace component to give and receive links to available services
- **Data Standardisation**: This feature enables ZDMP Assets to call services and use data sources provided by external platforms and vice versa. Data API and Data Standardisation: This connects with the T6.4 Service and Message Bus to allow data connectivity and, possibly, the ability to call remote applications or for remote platform to call local zApps
- **Harmonisation API**: This is an additional API to aide in the standardisation of data or Marketplace zApps
- **ADAMOS Service Plugin**: This plugin creates a connection with an instance of ADAMOS IIoT and can send data, security, and device information to this platform. Also acts as a good example for software platforms at manufacturer side. Thus, similar functionalities can be provided for other existing software platforms as well
- **eFactory Service Plugin**: This plugin creates a connection with the eFactory data spine. Allowing further connections to other eFactory connected platforms or other the eFactory platform itself
- **SDAIM Service Plugin**: This plugin creates and manages connections to the SDAIM platform, including data, security, and services

These high-level functions can be grouped to the following features which have to be realised:

Subtask	Subtask description
T65A001 Interconnection of ZDMP with external platforms	**Priority:** Must
	Who: ZDMP Platform Administrator **Where:** Anywhere **When:** Runtime, whenever interconnectivity to an external platform is needed **What:** Bi-directional access to service APIs and data sources **Why:** To enable ZDMP Assets to use the services and data provided by external platforms and vice versa
Acceptance Criteria	External platforms can be interconnected with ZDMP through External Platform Plugins. These plugins allow ZDMP Assets to use services and data provided by the interconnected platform and vice versa. Supported external platforms include ADAMOS, eFactory and SDAIM.
Requirements filled	RQ_0008, RQ_0009, RQ_0010, RQ_0011, RQ_0012, RQ_0167
T65B001 ZDMP and eFactory Marketplace Integration	**Priority:** Must
	Who: ZDMP Platform Administrator **Where:** Anywhere **When:** Runtime **What:** Integration for the Marketplaces of ZDMP and eFactory **Why:** To allow requests to each Marketplace for buying and selling of zApps and other 3rd party apps across the system
Acceptance Criteria	The eFactory and ZDMP Marketplaces can be accessed from either platform and allow buying and selling of zApps as well as other 3rd party apps across the system
Requirements filled	N/A
T65C001 Exchange of device- and sub-device Information	**Priority:** Must
	Who: Application Builder, Secure Business Cloud **Where:** Anywhere **When:** Runtime, whenever exchange of device- and sub-device information needed

	What: Device and sub-device information **Why:** To export devices and sub-devices to the ADAMOS platform
Acceptance Criteria	The T6.5 ADAMOS service plugin provides the functionality to export the devices and sub-devices per end-user/owner to the ADAMOS platform on request. This includes a full description of the device including technical information, a unique id of the device and a unique id of the end-user/owner
Requirements filled	RQ_0008, RQ_0009, RQ_0010, RQ_0011, RQ_0012, RQ_0167
T65C002 **Exchange of supported operations per device**	**Priority:** Must
	Who: T6.5 ADAMOS service plugin **Where:** Anywhere **When**: Runtime, whenever exchange of supported operations of a device needed **What:** Information of supported operations of a device **Why:** To make the list of supported operations of a device available to the ADAMOS platform
Acceptance Criteria	The T6.5 ADAMOS service plugin provides the functionality to export the list of supported operations per device to the ADAMOS platform on request. This includes a description of the operations and a unique id of the operation type
Requirements filled	RQ_0008, RQ_0009, RQ_0010, RQ_0011, RQ_0012, RQ_0167
T65C003 **Exchange of historical data of devices**	**Priority:** Must
	Who: T6.5 ADAMOS service plugin **Where:** Anywhere **When**: Runtime, whenever historical data needs to be exported to another platform **What:** Historical data of a device **Why:** To make historical data for a certain period available to the ADAMOS platform
Acceptance Criteria	The T6.5 ADAMOS service plugin provides the functionality to export historical data of a device for a certain time period, eg 3 months, including a unique id of the device, a unique id of the device type, and a unique id of the end-user/owner
Requirements filled	RQ_0008, RQ_0009, RQ_0010, RQ_0011, RQ_0012, RQ_0167
T65C004 **Exchange of end user/owner information**	**Priority:** Should
	Who: T6.5 ADAMOS service plugin **Where:** Anywhere **When**: Runtime, whenever information of user and groups needs to be exported to another platform **What:** Information of users and groups **Why:** To make user and groups available to the ADAMOS platform
Acceptance Criteria	The T6.5 ADAMOS service plugin provides the functionality to export data of users and groups including a description of the user/group and a unique id of the user/group and the permission (read-only, write) of the user/group
Requirements filled	RQ_0008, RQ_0009, RQ_0010, RQ_0011, RQ_0012, RQ_0167

Figure 121: Inter-Platform Interoperability Features

4.12.3 Workflows

4.12.3.1 Connect to an external platform

This feature enables the ZDMP platform to connect to external platforms. Connection means the exchange of pieces of information in both directions. This includes the export of information from the ZDMP platform as well as the import of information from other platforms. Depending on the use and the capabilities of the platform it may be the case that information exchange in only one direction is supported, eg the ZDMP platform may export data to another platform without importing data from that platform. The workflow is an example and is similar for all supported platforms.

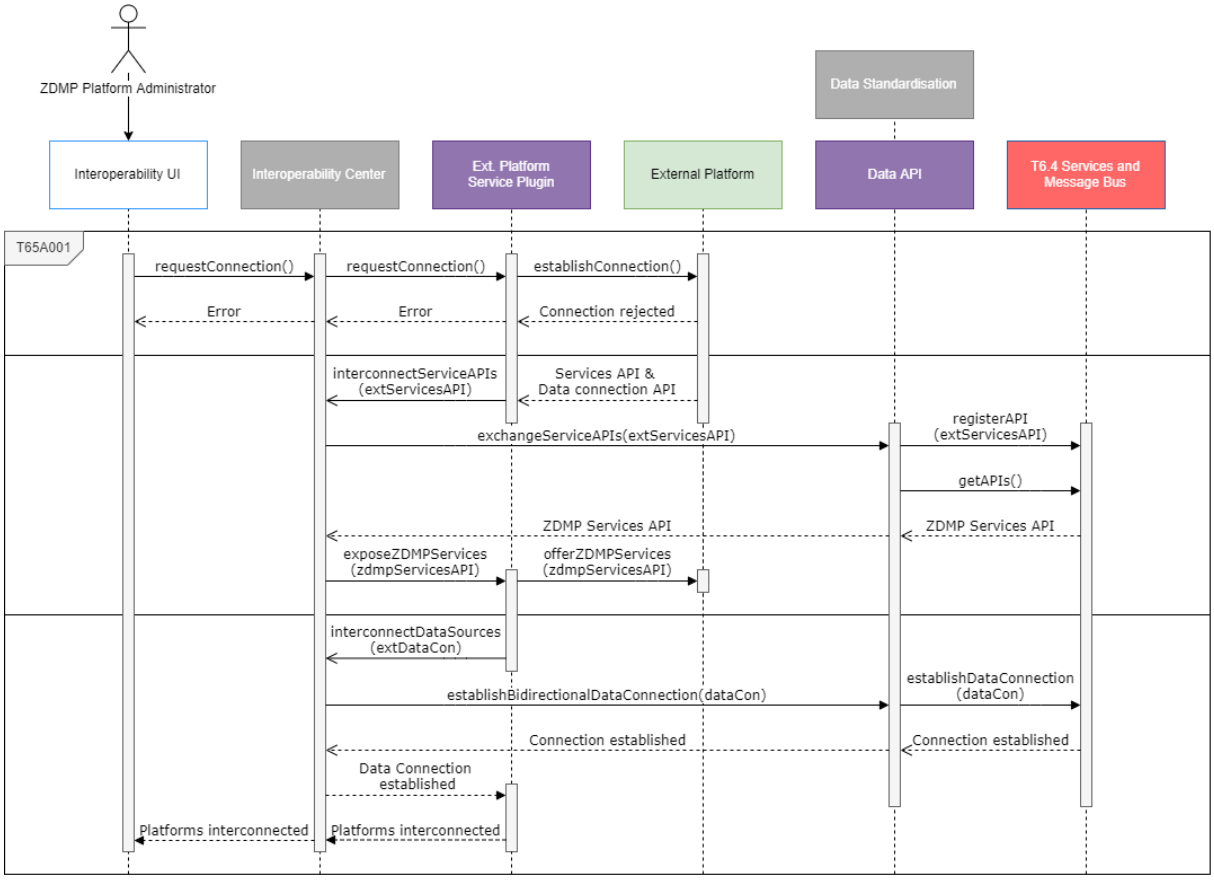


Figure 122: Interoperability to different external platforms

4.12.3.2 Marketplace Integration

This feature enables the harmonisation of security information between the platforms. This allows for secure communication and access between the platforms, including roles, organisations, and data owners.

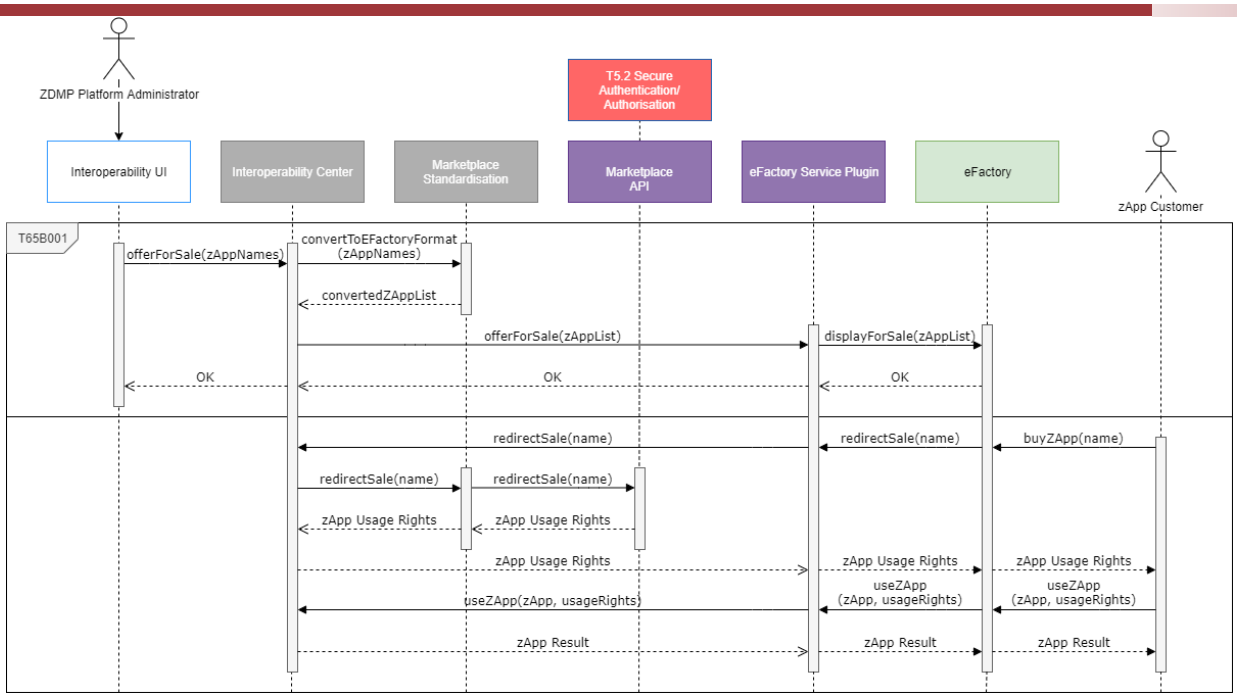


Figure 123: Marketplace Integration Sequence Diagram

4.12.4 Additional Issues

Additional issues have appeared after the description of the architecture.

Issue	Description	Next Steps	Lead (Rationale)
Intercompany connectivity	These requirements are addressed by the functionality of this component as a whole. RQ_0033, RQ_0066	Allow for intercompany connections of ZDMP platform	T6.5 Inter-platform Interoperability
Connectivity to legacy system	These requirements relate to a connection to a “legacy system”. More needs to be known as to whether this is included in T5.1 Data Acquisition or under T6.5 Inter-platform Interoperability. RQ_0622, RQ_0675, RQ_0734, RQ_0793, RQ_0879, RQ_0884, RQ_0975	Decide which component is responsible for this legacy system	T5.1 Data Acquisition and T6.5 Inter-platform Interoperability.

Figure 124: Interplatform Interoperability Additional Issues

5 Platform Tier: Run-time

The Platform Tier groups the main functionality of ZDMP as run-time components. These components are running within the T6.4 Application Run-time component. It includes elements that have been designed either separately or zApps created by the T6.1 Application Builder. It also includes utilities that are used by these zApps, such as the T6.4 Message Bus and T5.3 Data Harmonisation.

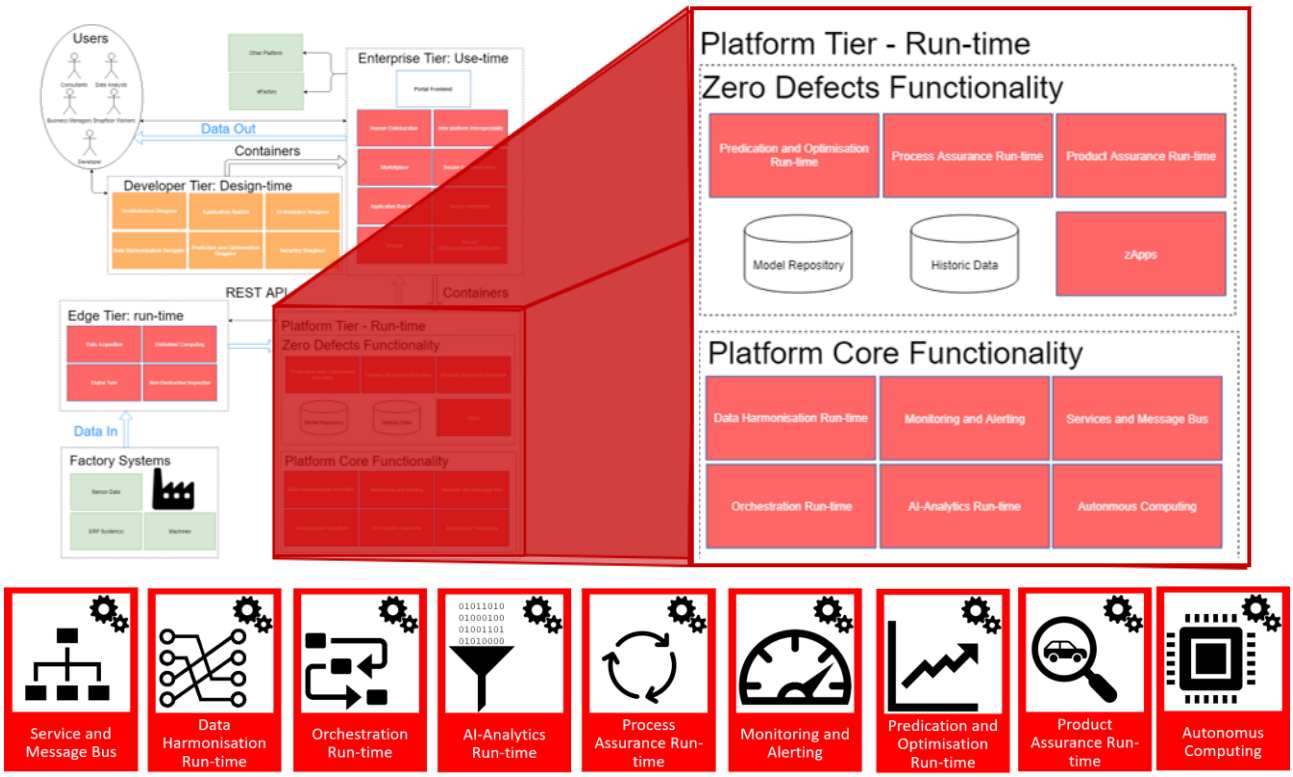
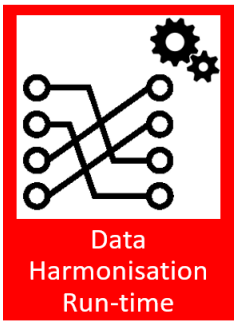


Figure 125: Platform-Tier Components

5.1 Data Harmonisation Run-time (T5.3)



5.1.1 Overall functional characterisation & Context

The Data Harmonisation Run-time component integrates data from existing software systems by executing the Manufacturing Maps created by the Data Harmonisation Designer component, ie transforming data from its source format to its destination format during the run-time. The maps created in the Data Harmonisation Designer component are deployed and encapsulated as services to be finally exposed as software mini-packages, ie Docker containers. These mini-packages, containing the transformation routines, are uploaded, and published in the ZDMP Marketplace to advertise and commercialise them.

5.1.2 Functions / Features

The Data Harmonisation and Interoperability component provides a set of functionalities that can be grouped on the following features:

- **Transform:** Manufacturing Maps can be executed in the form of a standalone service. This service contains the rules defined and deployed from the Manufacturing Map to transform a specific syntax format A into format B which could then, for example, be used as part of a process
- **Submit Usage Data:** Usage data will be captured and communicated to the Platform

Their function can be grouped to the following features:

Subtask	Subtask description
T53B001 **Get invocation**	**Priority:** Must
	Who: Data Harmonisation **Where:** Anywhere **When:** Runtime **What:** Get invocation request from Service Call **Why:** So that the transformation engine can execute the right transformation service
Acceptance Criteria	The invocation is relayed to the transformation engine
Requirements Filled	RQ_0886
T53B002 **Connect to Store**	**Priority:** Must
	Who: Data Harmonisation **Where:** Anywhere **When:** Runtime **What:** Connect to ZDMP Store with the credentials as directed by the T5.2 Secure Authentication/Authorisation component **Why:** So that the transformed data can be stored
Acceptance Criteria	The ZDMP Store is accessible
Requirements Filled	RQ_0759, RQ_0761, RQ_0880, RQ_0881, RQ_0885, RQ_0886, RQ_0921, RQ_0923
T53B003 **Unpack Routines**	**Priority:** Must
	Who: Data Harmonisation **Where:** Anywhere **When:** Runtime **What:** Unpack a mapping routine set **Why:** So that the Transformation engine can read the transformation steps determined in the Map **When/Where**: Design-time in the Data Harmonisation Designer

Acceptance Criteria	The routines are successfully unpacked, and the transformation steps are available for the transformation engine
Requirements Filled	RQ_0135
T53B004 Read Data	**Priority:** Must
	Who: Data Harmonisation **Where:** Anywhere **When:** Runtime (after T53B001) **What:** Reads source data as input parameter from the invocation **Why:** So that the input data can be transformed by executing the transformation services
Acceptance Criteria	The input data is available for transformation
Requirements Filled	RQ_0768, RQ_0775, RQ_0825, RQ_0880, RQ_0881, RQ_0885, RQ_0921, RQ_0923
T53B005 Transform	**Priority:** Must
	Who: Data Harmonisation **Where:** Anywhere **When:** Runtime **What:** Transforms the data **Why:** So that the routines of the mapping are executed
Acceptance Criteria	The transformation is successfully executed
Requirements Filled	N/A
T53B006 Push Transformed Data	**Priority:** Must
	Who: Data Harmonisation **Where:** Anywhere **When:** Runtime (after T35B005) **What:** Pushes transformed data back to the calling zApp **Why:** So that the routines of the mapping are executed
Acceptance Criteria	The transformation is successfully executed
Requirements Filled	RQ_0085, RQ_0697, RQ_699, RQ_0703, RQ_0805, RQ_0806, RQ_0807, RQ_0827
T53B007 Store Transformed Data	**Priority:** Should
	Who: Data Harmonisation **Where:** Anywhere **When:** Runtime (after T35B005) **What:** Store the transformed data **Why:** So that the transformed data is accessible without having to re-execute the transformation service again
Acceptance Criteria	The transformed data is successfully stored
Requirements Filled	RQ_0032, RQ_0085, RQ_0119, RQ_0637, RQ_0705, RQ_0753
T53B008 Submit Usage Data	**Priority:** Should
	Who: Data Harmonisation **Where:** Anywhere **When:** Runtime **What:** Submit usage data **Why:** So that the platform can make use of this data for monitoring purposes
Acceptance Criteria	The usage data is successfully received by the Platform
Requirements Filled	RQ_0032, RQ_0033, RQ_067, RQ_0085, RQ_0145, RQ_0637

Figure 126: Data Harmonization Run-time Features

5.1.3 Workflows

As the Data Harmonisation Run-time component is a service-based component, there are no UIs attached to it. Therefore, this sub-section only describes its sequence diagrams.

5.1.3.1 Transform

This feature deals with the preparation steps prior to executing a transformation service and the steps to execute a map. Figure 127 shows the sequence diagram of the transform feature.

The main steps/functionalities are as follows:

- Preparations:
  - Get Invocation (see function T53B001)
  - Connect to T6.2 ZDMP Marketplace (see function T53B002)
  - Execute Map:
  - Unpack Routines (see function T53B003)
  - Read Data (see function T53B004)
  - Transform (see function T53B005)
  - Push Transformed Data (see function T53B006)
  - Store Transformed Data (see function T53B007)

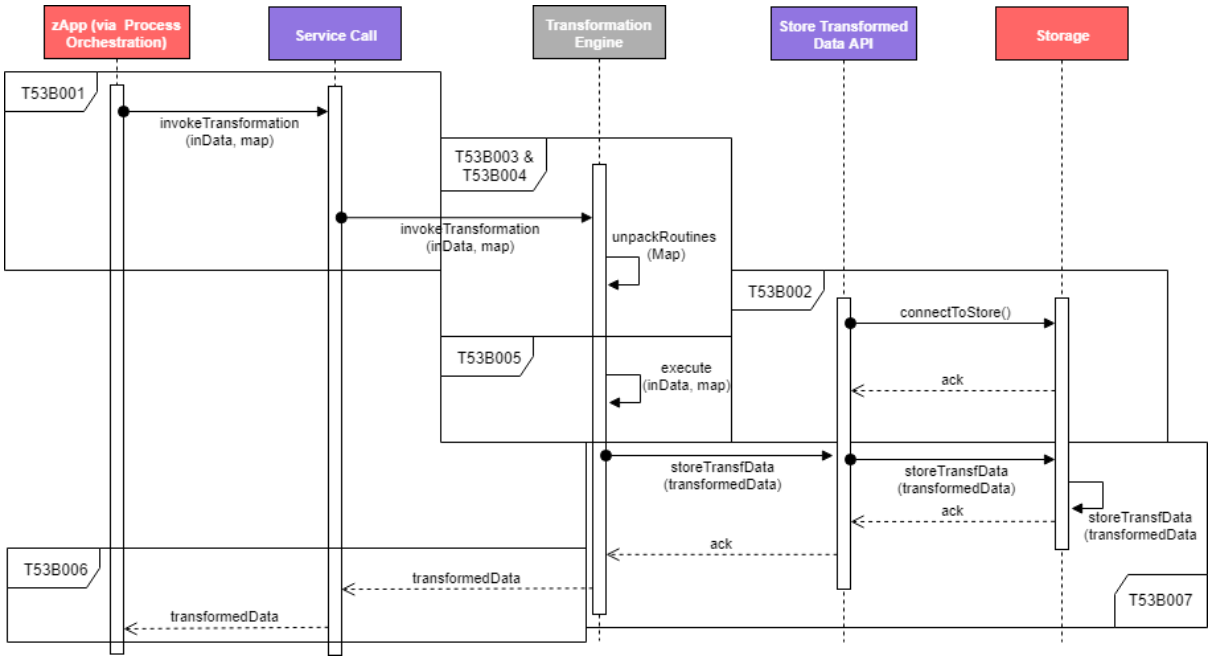


Figure 127: Transform Sequence Diagram

5.1.3.2 Submit Usage Data

This feature provides the capability to deploy and publish a map after it has been generated by the Business Analyst. Figure 128 shows the sequence diagram.

There is only one step corresponding to this feature:

- Submit Usage Data

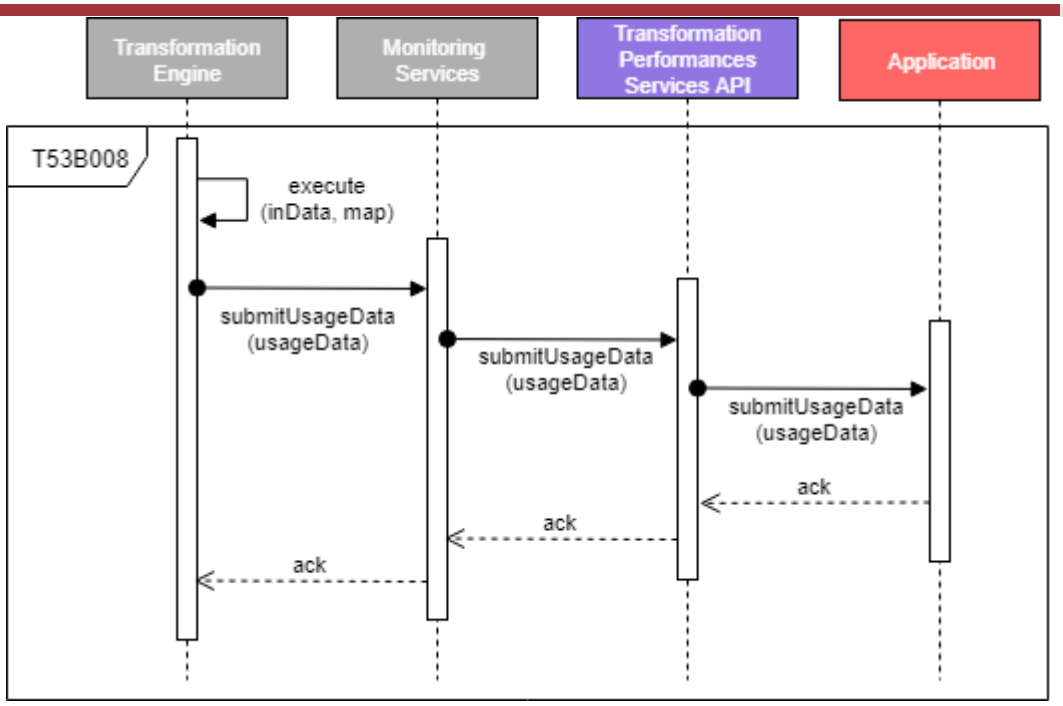


Figure 128: Submit Usage Data Sequence Diagram

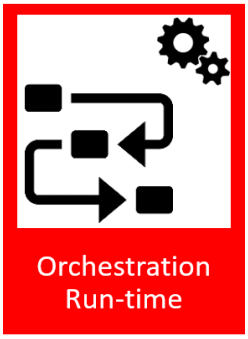
5.1.4 Additional Issues

Additional issues have appeared after the description of the architecture.

Issue	Description	Next Steps	Lead (Rationale)
Performance Issues	The following requirements with a “must“-priority were targeted at the task 5.3, but there are concerns about performance: *RQ_0036, RQ_0082, RQ_0090, RQ_0110, RQ_0118, RQ_0132, RQ_0136, RQ_0150*	Discuss with requirement providers who to solve the issue	T5.3 Data Harmonisation
Subtasks without specified fulfilled requirements	The following subtasks do not fulfil specific requirements, but are there to fulfil a more general purpose: *T53B005*	Ensure that all the tasks have requirements	T5.3 Data Harmonisation

Figure 129: Additional Issues Data Harmonization Run-time

5.2 Orchestration Run-time (T5.4)



The interaction between Orchestration Designer and other components of the ZDMP platform (Orchestration Runtime, Monitoring and Alerting, Process Engine, Marketplace, Storage, etc) will be facilitated using the T6.4 Service and Message Bus. The Monitoring and Alerting provides alerts based on a set of incoming data and rules.

5.2.1 Overall functional characterisation & Context

For BPMN process execution the Camunda Process Engine is used, due to its open source design and flexibility. The Process API acts as the bridge between the Orchestration Designer and the other components like the Process Engine, Marketplace, etc.

5.2.2 Functions / Features

The Process API component is composed of the following modules:

- **BPMN Parser**: This module is responsible for parsing BPMN files to ensure their validity. It is also used when importing a new process from an external file
- **Process Execution Manager:** This module bridges the Orchestration Designer and the Camunda Process Engine, managing the deployment of process models
- **Process Instance Controller**: This module interacts with the Camunda Process Engine, and handles running process instances, eg starting/stopping and getting metrics to message bus topics so other components can get information about process instances
- **Service Manager**: The Service Manager module is responsible for the organisation of services that are needed for the execution of the process, returning to the BPMS information such as service execution URI, parameters, or availability. This module reserves all needed services at the beginning of the execution. The instantiated service is called from the BPMS module when a new service is used

The functions can be grouped to the following features which have to be realised:

Subtask	Subtask description
T54B001 Deploy to Camunda Engine	**Priority:** Must
	Who: Developer **What:** Connect to Process Execution Manager **Why:** Prepare deployment for process model **When:** During installation of a process **Where:** On the platform
Acceptance Criteria	The process is available to start on the platform
Requirements filled	See additional issues table (Section 5.2.4) below
T54B002 New Process Instance	**Priority:** Must
	Who: Process API **What:** Prepare process model for execution **Why:** So that process can be executed **When:** During installation of a process **Where:** On the platform
Acceptance Criteria	Allocate resources for new process instance, notify process engine about a new instance to be created
Requirements filled	See additional issues table (Section 5.2.4) below
T54B003	**Priority:** Must

Request Services	**Who:** Process API **What:** Connect to the Service Manager **Why:** Get metadata information for each service used in the process model for further execution **When:** During installation of a process **Where:** On the platform
Acceptance Criteria	Service Manager returns all metadata information
Requirements filled	See additional issues table (Section 5.2.4) below
T54B004 **Launch and start process (and get metrics)**	**Priority:** Must
	Who: User **What:** Launches the process and start getting metrics **Why:** A new instance is created, and metrics start being collected **When:** On user command **Where:** On the platform
Acceptance Criteria	Metrics about the instance are being collected
Requirements filled	See additional issues table (Section 5.2.4) below
T54B005 **Manage process instances runtime values**	**Priority:** Must
	Who: Process API **What:** Manage process instances and get/set process instance variable values **Why:** So, run-time values can be read and write **When:** During process run-time **Where:** On the platform
Acceptance Criteria	Information about each process instance can be retrieved, and process instance variable values can be read/write
Requirements filled	See additional issues table (Section 5.2.4) below
T54B006 **Manage process instances runtime values**	**Priority:** Must
	Who: Process API **What:** Compile external service metadata so they can be called as part of the process execution **Why:** So that a process is effectively helped by the execution of 3rd party services **When:** During process run-time **Where:** On the platform
Acceptance Criteria	Process instance interacts with external services
Requirements filled	See additional issues table (Section 5.2.4) below
T54B007 **Start process instance**	**Priority:** Must
	Who: Orchestration Designer **What:** Start a new instance from an existing process model **Why:** Manually trigger instance start from process engine **When:** During process run-time **Where:** On the platform
Acceptance Criteria	A new instance of an existing process model is running in the Process Runtime
Requirements Fields	See additional issues table (Section 5.2.4) below
T5BA008 **Stop/Suspend/Resume process instance**	**Priority:** Must
	Who: Orchestration Designer **What:** Stop/Suspend/Resume process instances **Why:** User can interact with process instances **When:** Due to API call **Where:** In the platform
Acceptance Criteria	The running process instance is stopped, suspended, or resumed
Requirements filled	See additional issues table (Section 5.2.4) below

Figure 130: Orchestration Run-Time Features

5.2.3 Workflows

This section is highlighting the user interaction and the interaction of the component in the single function.

5.2.3.1 Process deployment and instance creation sequence

This deploys a process from the Designer to the Process Instance Controller.

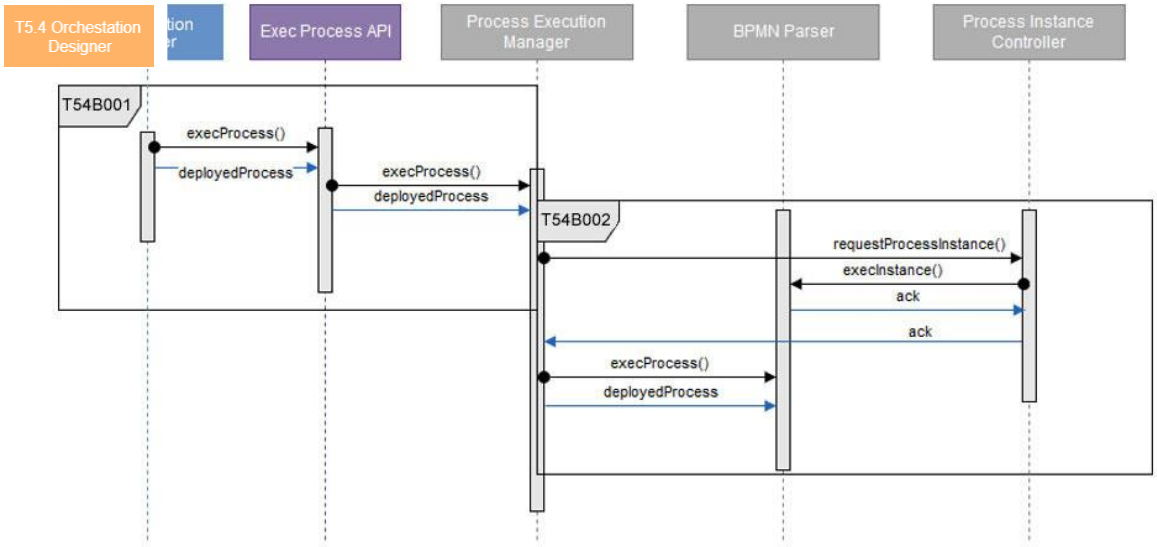


Figure 131: Process deployment and instance creation

5.2.3.2 Process runtime execution and service provisioning

This executes the process and connects it to the Camunda process engine.

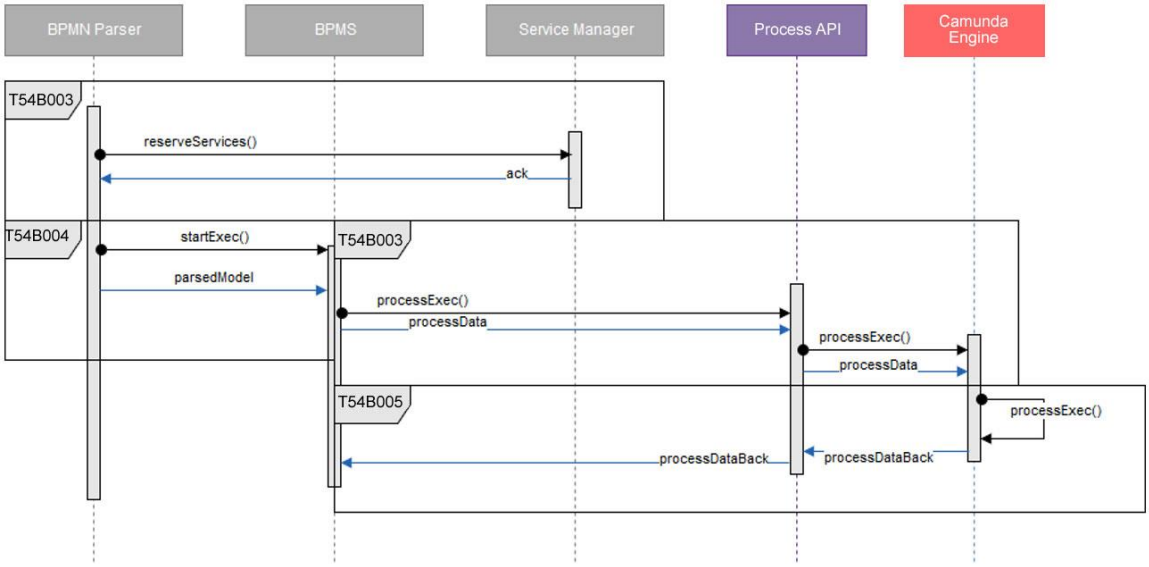


Figure 132: Process deployment and instance creation

5.2.4 Additional Issues

Additional issues have appeared after the description of the architecture.

Issue	Description	Next Steps	Lead (Rationale)
General component	These functions do not match any requirements as this component has very general functionality that could be used in many of the zApps. This is discussed more below	See Below	T5.4 Orchestration Runtime
Customisation level of zApps	This component can be used to create a business process flow which could be customised by technical users. This could be useful in the following requirements: RQ_0258, RQ_0331, RQ_0340, RQ_0370	Discuss with zApp developers on the usefulness of the orchestration services	T5.4 Orchestration Runtime
Performance	The following requirements relate to the performance. In some cases, the Process Orchestration may slow these down as its built to run on human timescales and longer running processes rather than for highly optimised real-time systems: RQ_0349, RQ_0036, RQ_0110, RQ_0014, RQ_0018, RQ_0019, RQ_0025, RQ_0104, RQ_0118, RQ_0132, RQ_0144, RQ_0150, RQ_0278, RQ_0279, RQ_0339, RQ_0360, RQ_0369, RQ_0378, RQ_0386, RQ_0451, RQ_0555, RQ_0090	Discuss with zApp developers on the usefulness of the orchestration services	T5.4 Orchestration Runtime
Linking components, zApps and other services.	The whole functionality of T5.4 Orchestration Designer and Runtime is to link processes and services together. The following could utilise this functionality: RQ_0020, RQ_0047, RQ_0054, RQ_0072, RQ_0074, RQ_0076, RQ_0078, RQ_0080, RQ_0081, RQ_0098, RQ_0101, RQ_0105, RQ_0106, RQ_0107, RQ_0128, RQ_0129, RQ_0130, RQ_0140, RQ_0141, RQ_0145, RQ_0147, RQ_0151, RQ_0263, RQ_0271, RQ_0274, RQ_0306, RQ_0308, RQ_0309, RQ_0322, RQ_0325, RQ_0326, RQ_0327, RQ_0335, RQ_0338, RQ_0343, RQ_0696, RQ_0700, RQ_0759, RQ_0761, RQ_0818, RQ_0920, RQ_0925, RQ_0932, RQ_0933, RQ_0934, RQ_0935, RQ_0037	Discuss with zApp developers on the usefulness of the orchestration services	T5.4 Orchestration Runtime

Figure 133: Orchestration Runtime Additional Issues

5.3 Monitoring and Alerting (T5.4)

5.3.1 Overall functional characterization & Context

The Monitoring and Alerting component is composed of an API and a HTML / CSS / JS based web frontend where the users can see snapshots of the incoming data from the Services and Message Bus (T6.4) and the Orchestration Run-Time (T5.4). A name for a KPI can be defined as well as the topics and filtering expression needed to obtain the KPI from the message bus. Based on the KPIs defined rules can be created. A rule is made up of a comparator and a notification. Users can then react on a notification, indicating that someone is already taking care of the notification, and then the system knows it is not necessary to escalate the notification to the next user in the chain of users to be notified.

The rules themselves incorporate time span validity, start and end date validity rules, multiple equality comparators selectable for users. For the selection of the KPIs simple JSON queries for JSON documents are used, simple XPATH queries for XML documents are used. For all other transformations necessary, a service from T5.3 Data Harmonisation can be created and called in between.

5.3.2 Functions / Features

These elements and functionalities that can be added to such a template app are described as follows:

- **KPI Definer**: The user can define a name for a KPI as well as the topics and filtering expression needed to obtain the KPI from the message bus
- **Monitoring:** The user can select KPIs to monitor its incoming data from the Services and Message Bus (T6.4) and the Orchestration Run-Time (T5.4)
- **Alerting:** The user can define rules based on the KPIs. A rule is made up of a comparator and an alert. Users can then react on an alert, indicating that someone is already taking care of the alert, and then the system knows it is not necessary to escalate the alert to the next user in the chain of users to be alerted

The function can be grouped to the following features which have to be realised:

Subtask	Subtask description
T54C001 **CRUD KPI**	**Priority:** Must
	Who: User **When:** Run time Where: Anywhere (in the Monitoring and Alerting UI) **What:** Allow the user to define a KPI and its filtering expressions. **Why:** Structure data to be monitored
Acceptance Criteria	KPIs were created, read, updated, or deleted, HTTP 200
Requirements filled	RQ_0174, RQ_0175, RQ_0194, RQ_0201, RQ_0222, RQ_0244, RQ_0396, RQ_0397, RQ_0398, RQ_0399, RQ_0400, RQ_0401, RQ_0402, RQ_0403, RQ_0406, RQ_0407, RQ_0410, RQ_0557, RQ_0558, RQ_0559, RQ_0560, RQ_0642, RQ_0643, RQ_0644, RQ_0645, RQ_0759, RQ_0760, RQ_0761, RQ_0762, RQ_0864, RQ_0868, RQ_0869, RQ_0872, RQ_0873, RQ_0874, RQ_0882, RQ_0883, RQ_0887, RQ_0888, RQ_0907, RQ_0908, RQ_0914, RQ_0915, RQ_0916, RQ_0917, RQ_0922, RQ_0925, RQ_0932, RQ_0933, RQ_0934, RQ_0935, RQ_0995, RQ_1010, RQ_1018
T54C002 **Get KPI List**	**Priority:** Must
	Who: Monitoring and Alerting Component **When:** Run time **Where**: Anywhere (in the Storage Component)

	What: Lists existing KPIs, and its filtering expression. **Why:** Browse existing KPIs to Monitor its data or create alerts.
Acceptance Criteria	List of KPIs was received in JSON structure, HTTP 200
Requirements filled	RQ_0174, RQ_0175, RQ_0557, RQ_0558, RQ_0559, RQ_0560, RQ_0759, RQ_0760, RQ_0761, RQ_0762, RQ_0864, RQ_0868, RQ_0869, RQ_0872, RQ_0873, RQ_0874, RQ_0882, RQ_0883, RQ_0887, RQ_0888, RQ_0907, RQ_0908, RQ_0914, RQ_0915, RQ_0916, RQ_0917, RQ_0922, RQ_0925, RQ_0932, RQ_0933, RQ_0934, RQ_0935, RQ_0995, RQ_1010, RQ_1018
T54C003 **Monitor KPI data**	**Priority:** Must
	Who: User **When:** Run time **Where**: Anywhere (in the Monitoring and Alerting UI) **What:** Monitor incoming KPI data from Message Bus and Orchestration Run-time **Why:** Monitor KPIs value, defining Alerts and reducing risks
Acceptance Criteria	Connection to the Message Bus was established and if defined queries matched data to identify KPIs, bounds of rules are checked and actions started if necessary
Requirements filled	RQ_0174, RQ_0175, RQ_0194, RQ_0201, RQ_0222, RQ_0396, RQ_0397, RQ_0398, RQ_0399, RQ_0400, RQ_0401, RQ_0402, RQ_0403, RQ_0406, RQ_0407, RQ_0410, RQ_0557, RQ_0558, RQ_0559, RQ_0560, RQ_0642, RQ_0643, RQ_0644, RQ_0645, RQ_0759, RQ_0760, RQ_0761, RQ_0762, RQ_0864, RQ_0868, RQ_0869, RQ_0872, RQ_0873, RQ_0874, RQ_0882, RQ_0883, RQ_0887, RQ_0888, RQ_0907, RQ_0908, RQ_0914, RQ_0915, RQ_0916, RQ_0917, RQ_0922, RQ_0925, RQ_0932, RQ_0933, RQ_0934, RQ_0935, RQ_0995, RQ_1010, RQ_1018
T54C004 **CRUD Alert**	**Priority:** Must
	Who: User **When:** Run time **Where**: Anywhere (in the Monitoring and Alerting UI) **What:** Create, read, update, and delete alerts, its conditions and the users and components that should be alerted **Why:** Alert users and components in case a KPI value is not in the desired range
Acceptance Criteria	An alert was created, read, updated, or deleted, HTTP 200
Requirements filled	RQ_0193, RQ_0201, RQ_0222, RQ_0244, RQ_0266, RQ_0267, RQ_0277, RQ_0343, RQ_0396, RQ_0397, RQ_0398, RQ_0399, RQ_0400, RQ_0401, RQ_0402, RQ_0403, RQ_0406, RQ_0407, RQ_0410, RQ_0557, RQ_0558, RQ_0559, RQ_0560, RQ_0642, RQ_0643, RQ_0644, RQ_0645, RQ_0759, RQ_0760, RQ_0761, RQ_0762, RQ_0864, RQ_0868, RQ_0869, RQ_0872, RQ_0873, RQ_0874, RQ_0882, RQ_0883, RQ_0887, RQ_0888, RQ_0907, RQ_0908, RQ_0914, RQ_0915, RQ_0916, RQ_0917, RQ_0922, RQ_0925, RQ_0932, RQ_0933, RQ_0934, RQ_0935, RQ_0995, RQ_1010, RQ_1018
T54C005 **Alert users**	**Priority:** Must
	Who: Monitoring and Alerting Component **When**: Run time **Where:** Anywhere (in the Human Collaboration Component) **What:** Alert users and components in case a KPI value is not in the desired range **Why:** Keep users informed about important or critical situations related to the KPI
Acceptance Criteria	Alert is received by user for the chosen action trigger
Requirements filled	RQ_0014, RQ_0036, RQ_0037, RQ_0038, RQ_0089, RQ_0147, RQ_0149, RQ_0265, RQ_0277, RQ_0343, RQ_0344, RQ_0396, RQ_0397, RQ_0398, RQ_0399, RQ_0400, RQ_0401, RQ_0402, RQ_0403, RQ_0406, RQ_0407, RQ_0410, RQ_0557, RQ_0558, RQ_0559, RQ_0560, RQ_0642, RQ_0643, RQ_0644, RQ_0645, RQ_0759, RQ_0760, RQ_0761, RQ_0762, RQ_0864, RQ_0868, RQ_0869, RQ_0872, RQ_0873, RQ_0874, RQ_0882, RQ_0883, RQ_0887, RQ_0888, RQ_0907, RQ_0908, RQ_0914, RQ_0915, RQ_0916, RQ_0917, RQ_0922, RQ_0925, RQ_0932, RQ_0933, RQ_0934, RQ_0935, RQ_0995, RQ_1010, RQ_1018
T54C006 **React on Alert**	**Priority:** Must
	Who: User **When:** Run time **Where**: Anywhere (in the Monitoring and Alerting UI) **What:** The user may react to a received alert, indicating that the alert situation is already under control or someone is already taking control of the situation, stopping the notification to other users **Why:** Keep track of the alerts and the user's notifications response
Acceptance Criteria	An alert contains URLs for possible user responses and can relate those uniquely to the choice the user has, which is part of the Human Interaction component

Requirements filled	RQ_0201, RQ_0222, RQ_0557, RQ_0558, RQ_0559, RQ_0560, RQ_0642, RQ_0643, RQ_0644, RQ_0645, RQ_0759, RQ_0760, RQ_0761, RQ_0762, RQ_0864, RQ_0868, RQ_0869, RQ_0872, RQ_0873, RQ_0874, RQ_0882, RQ_0883, RQ_0887, RQ_0888, RQ_0907, RQ_0908, RQ_0914, RQ_0915, RQ_0916, RQ_0917, RQ_0922, RQ_0925, RQ_0932, RQ_0933, RQ_0934, RQ_0935, RQ_0995, RQ_1010, RQ_1018

Figure 134: Monitoring and Alerting Functions

5.3.3 Workflows

This section is highlighting the user interaction and the interaction of the component in the single function.

5.3.3.1 CRUD KPI

The Monitoring and Alerting Component allow the user to define a KPI and its filtering expressions, as can be seen in the following figure.

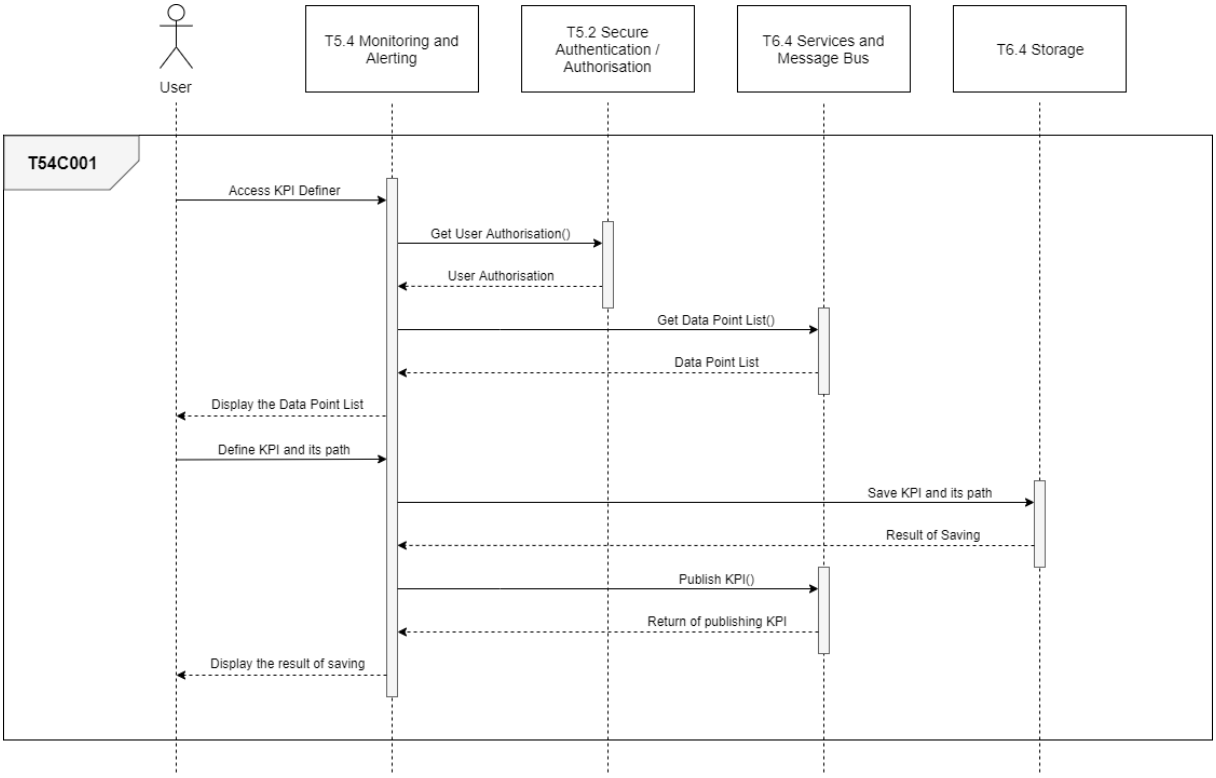


Figure 135: Define KPI

5.3.3.2 Get KPI List and Monitor KPI

The KPIs previously defined can be retrieved from the Storage, considering the Authorisations the user has. The KPI data can be monitored, receiving the values from the Message Bus and the Orchestration Run-time. The flow is depicted on the following figure.

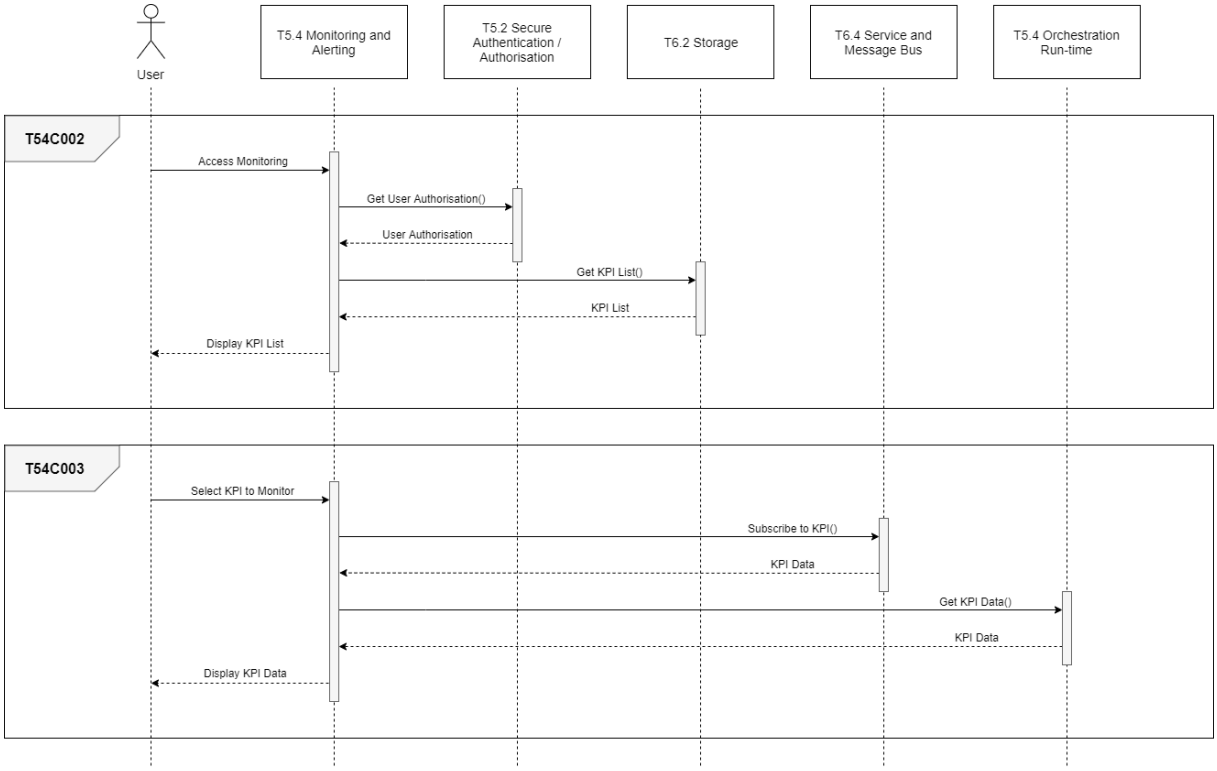


Figure 136: Get KPI list and Monitor KPI

5.3.3.3 Create Alerts

The user can create alerts, where the conditions are based on the KPI data, and the user can select other users and components to be alerted. The conditions contain a comparator that is applied to the KPI values, and the user can specify a chain of users and components to be notified, as well as the time between notifications. The flow is depicted in the following figure.

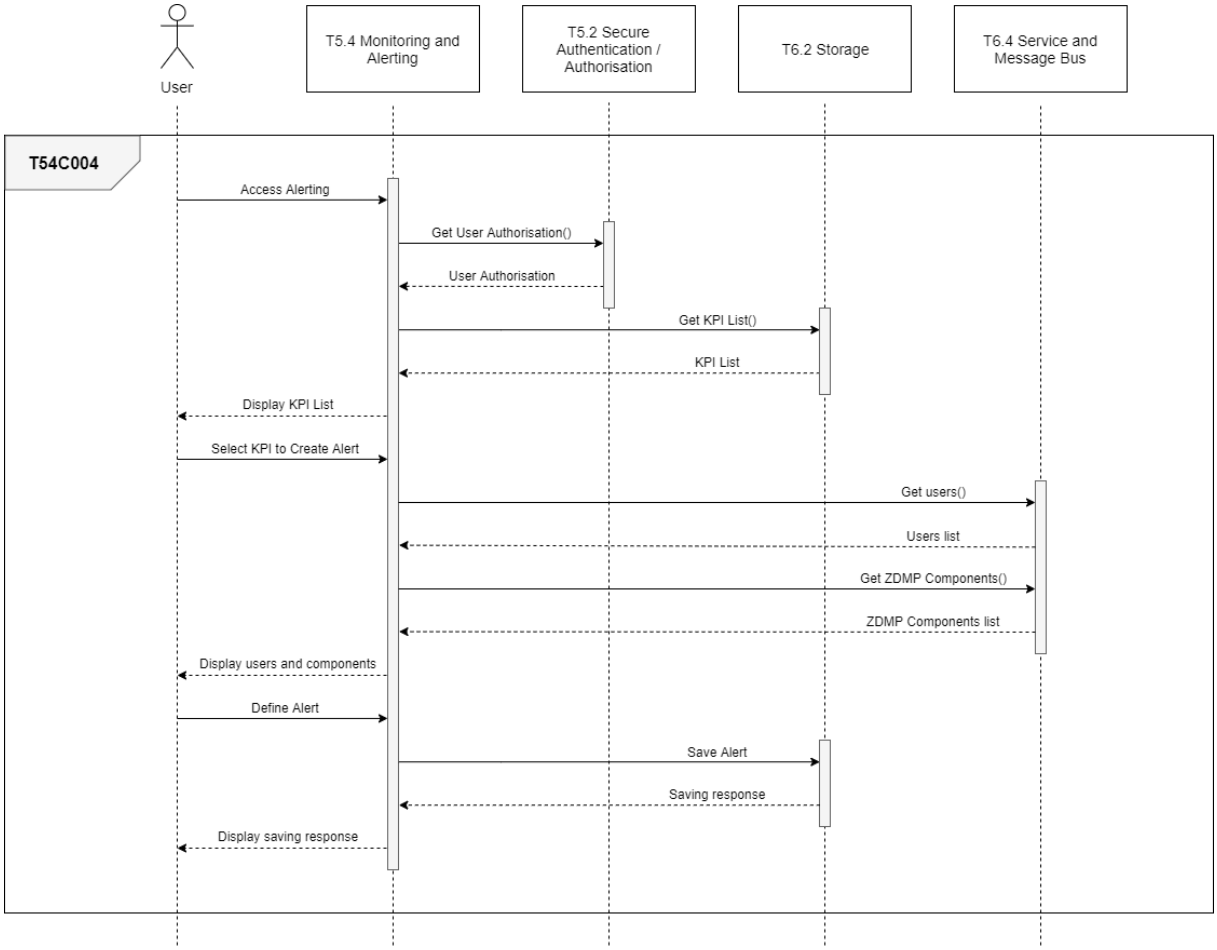


Figure 137: Creating Alert

5.3.3.4 Send Alerts and React to Alerts

The Monitoring and Alerting component will send user and component alerts when a KPI data matches the condition defined for an Alert, the user will be notified through the Human Collaboration Component (T6.3). After the notification is received, a user can react on the notification, indicating that the other users in the chain of notification should not receive the alert. The flow is depicted in the following figure.

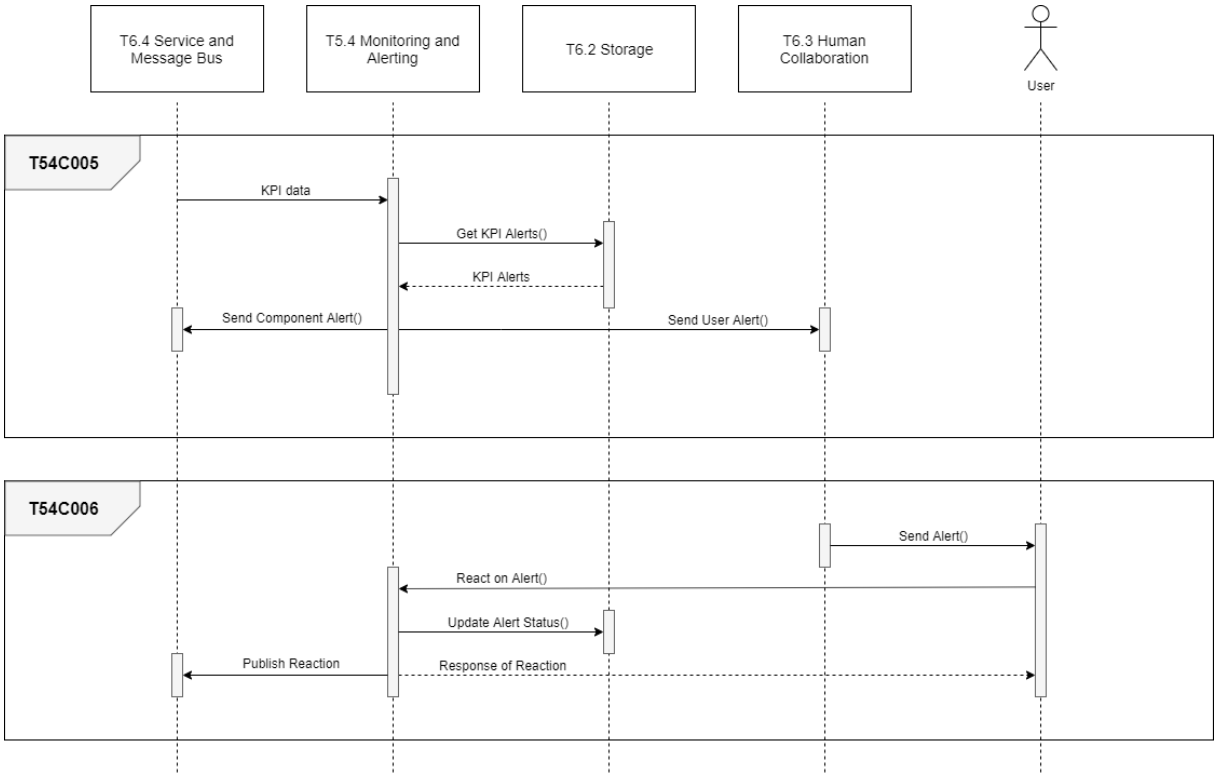
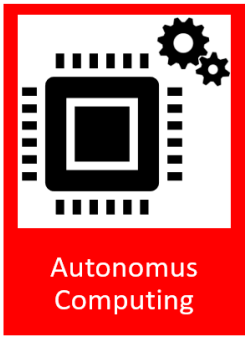


Figure 138: Send Alerts and User reaction

5.4 Autonomous Computing (T5.5)

5.4.1 Overall functional characterization & Context



The Autonomous Computing component is composed of an API and a HTML / CSS / JS-based web frontend where users define KPIs, their desired ranges of values and the processes (designed in T5.4 Orchestration component) to be executed when the KPI's value leaves the defined range. The UI also provides graphs and reports regarding the KPI's values.

5.4.2 Functions / Features

The elements and functionalities that can be found in the component are described as follows:

- **KPI Subscriber**: Where the user subscribes and unsubscribes to/from KPIs
- **Autonomous Processes Manager**: With this feature the user defines one or more KPIs, their desired range, and which processes should be executed if the KPI gets out of range
- **KPI Dashboard**: Where the user can inspect graphs, reports, and other information regarding the subscribed KPIs, the Autonomous Processes defined, and the reports of the processes started autonomously as well

The function can be grouped to the following features:

Subtask	Subtask description
T55A001 **Subscribe to KPI**	**Priority:** Must
	Who: User, must be authorized to read KPI data **When:** Design-time / Runtime of the application (topic must be known) **Where**: Anywhere on the same level the application publishes events to the bus **What:** Subscribe to an existing KPI to react on it and / or push the data to Storage component for historic data collection and analytics reasons **Why:** Monitor and define autonomous processes related to the KPI
Acceptance Criteria	Caller gets HTTP 200 and continuously gets new data pushed to KPI topic on the Message Bus; data gets pushed to Storage
Requirements filled	RQ_0333, RQ_0338, RQ_0339, RQ_0348, RQ_0356, RQ_0358, RQ_0359, RQ_0365, RQ_0366, RQ_0367, RQ_0368, RQ_0369, RQ_0372, RQ_0373, RQ_0374, RQ_0375, RQ_0376, RQ_0377, RQ_0378, RQ_0395, RQ_0396, RQ_0397, RQ_0398, RQ_0399, RQ_0400, RQ_0401, RQ_0402, RQ_0403, RQ_0411, RQ_0459, RQ_0493, RQ_0494, RQ_0496, RQ_0514, RQ_0549, RQ_0557, RQ_0558, RQ_0559, RQ_0560, RQ_0574, RQ_0585, RQ_0629, RQ_0638, RQ_0639, RQ_0741, RQ_0742, RQ_0757, RQ_0758
T55A002 **Unsubscribe to KPI**	**Priority:** Must
	Who: User **When:** Design-time / Runtime of the application (topic must be known) **Where**: Anywhere on the same level the application publishes events to the bus **What:** Remove subscription to a KPI previously subscribed **Why:** Stop monitoring and defining autonomous processes related to the KPI
Acceptance Criteria	Caller gets HTTP 200 and stop getting new data pushed to KPI topic on the Message Bus; data gets updated in Storage
Requirements filled	RQ_0333, RQ_0339, RQ_0348, RQ_0359, RQ_0368, RQ_0369, RQ_0377, RQ_0378, RQ_0395, RQ_0396, RQ_0397, RQ_0398, RQ_0399, RQ_0400, RQ_0401, RQ_0402, RQ_0403, RQ_0496, RQ_0514, RQ_0549, RQ_0557, RQ_0558, RQ_0559, RQ_0560, RQ_0574, RQ_0585, RQ_0629, RQ_0638, RQ_0639
T55A003 Get KPI Data	**Priority:** Must
	Who: Autonomous Computing Component

	When: Design-time / Runtime of the application (topic must be known) **Where**: Anywhere on the same level the application publishes events to the bus **What:** Gets KPI data, such as values over time, starting processes, etc **Why:** Present KPI information and graphs to the user.
Acceptance Criteria	Invoker got structured data from the KPI
Requirements filled	RQ_0338, RQ_0339, RQ_0345, RQ_0346, RQ_0356, RQ_0358, RQ_0365, RQ_0366, RQ_0367, RQ_0369, RQ_0378, RQ_0395, RQ_0396, RQ_0397, RQ_0398, RQ_0399, RQ_0400, RQ_0401, RQ_0402, RQ_0403, RQ_0411, RQ_0459, RQ_0514, RQ_0557, RQ_0558, RQ_0559, RQ_0560, RQ_0574, RQ_0585, RQ_0638, RQ_0639, RQ_0757, RQ_0758
T55A004 CRUD Autonomous Process for KPI	**Priority:** Must
	Who: User **When:** Design-time / Runtime of the application (topic must be known) **Where**: Anywhere on the same level the application publishes events to the bus **What:** Define / Update / Remove conditions and actions **Why:** Control a KPI and optimize the response time for countermeasures
Acceptance Criteria	Caller gets HTTP response, and a Json object with the Autonomous Process object.
Requirements filled	RQ_0333, RQ_0334, RQ_0336, RQ_0337, RQ_0338, RQ_0339, RQ_0340, RQ_0348, RQ_0352, RQ_0353, RQ_0354, RQ_0356, RQ_0358, RQ_0359, RQ_0365, RQ_0366, RQ_0367, RQ_0368, RQ_0369, RQ_0372, RQ_0373, RQ_0374, RQ_0375, RQ_0376, RQ_0377, RQ_0378, RQ_0395, RQ_0396, RQ_0397, RQ_0398, RQ_0399, RQ_0400, RQ_0401, RQ_0402, RQ_0403, RQ_0411, RQ_0493, RQ_0494, RQ_0496, RQ_0497, RQ_0498, RQ_0499, RQ_0500, RQ_0501, RQ_0502, RQ_0503, RQ_0504, RQ_0549, RQ_0557, RQ_0558, RQ_0559, RQ_0560, RQ_0574, RQ_0585, RQ_0629, RQ_0638, RQ_0639, RQ_0741, RQ_0742, RQ_0757, RQ_0758
T55A005 Get Autonomous Process for KPI	**Priority:** Must
	Who: Autonomous Computing Component **When:** Design-time / Runtime of the application (topic must be known) **Where**: Anywhere on the same level the application publishes events to the bus **What:** Gets the defined Autonomous Processes for a KPI **Why:** Browse the KPI's autonomous processes, to present a UI
Acceptance Criteria	A JSON with a structured list of processes and the currently set rules are returned with a HTTP 200.
Requirements filled	RQ_0336, RQ_0337, RQ_0338, RQ_0339, RQ_0345, RQ_0346, RQ_0354, RQ_0359, RQ_0368, RQ_0369, RQ_0378, RQ_0585
T55A006 Monitor KPI's values and Start Actions	**Priority:** Must
	Who: Autonomous Computing Component **When:** Design-time / Runtime of the application (topic must be known) **Where**: Anywhere on the same level the application publishes events to the bus **What:** Receive the KPI data/values from the message bus and apply the rules to see if any process must be started. **Why:** Start processes or actions when the KPI data matches the rules defined
Acceptance Criteria	When KPIs are out of bounds, notifications, or actions to correct the KPI are started.
Requirements filled	RQ_0333, RQ_0334, RQ_0335, RQ_0336, RQ_0337, RQ_0338, RQ_0339, RQ_0340, RQ_0348, RQ_0352, RQ_0353, RQ_0354, RQ_0356, RQ_0358, RQ_0359, RQ_0365, RQ_0366, RQ_0367, RQ_0368, RQ_0369, RQ_0372, RQ_0373, RQ_0374, RQ_0375, RQ_0376, RQ_0378, RQ_0395, RQ_0396, RQ_0397, RQ_0398, RQ_0399, RQ_0400, RQ_0401, RQ_0402, RQ_0403, RQ_0411, RQ_0459, RQ_0496, RQ_0497, RQ_0498, RQ_0499, RQ_0500, RQ_0501, RQ_0502, RQ_0503, RQ_0504, RQ_0514, RQ_0549, RQ_0557, RQ_0558, RQ_0559, RQ_0560, RQ_0574, RQ_0585, RQ_0629, RQ_0638, RQ_0639, RQ_0741, RQ_0742, RQ_0757, RQ_0758

Figure 139: Autonomous Computing Function

5.4.3 Workflows

5.4.3.1 Search and Subscribe to a KPI

The Autonomous Computing component allows a user to subscribe to a KPI in order to monitor its values and define conditions and actions (autonomous processes) related to it, and for this it must obtain the list of KPIs the user has permission to subscribe to. The subscribing process is described in the following figure.

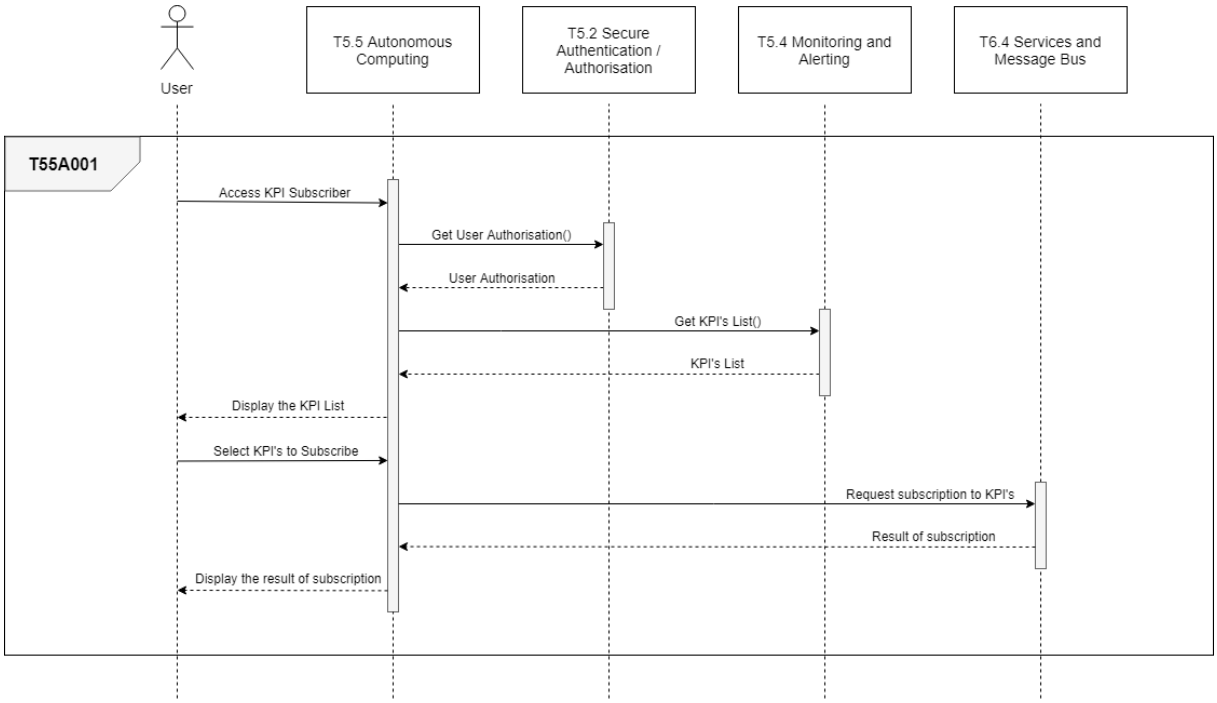


Figure 140: Retrieve KPI's available and Subscribe to it

5.4.3.2 Search Subscribed KPI's and Unsubscribe from KPIs

The Autonomous Computing component allows a user to unsubscribe to a KPI to stop monitoring its values, and the user can decide to keep or not the conditions and actions they previously defined to the unsubscribed KPI. The unsubscribing process is described in the following figure.

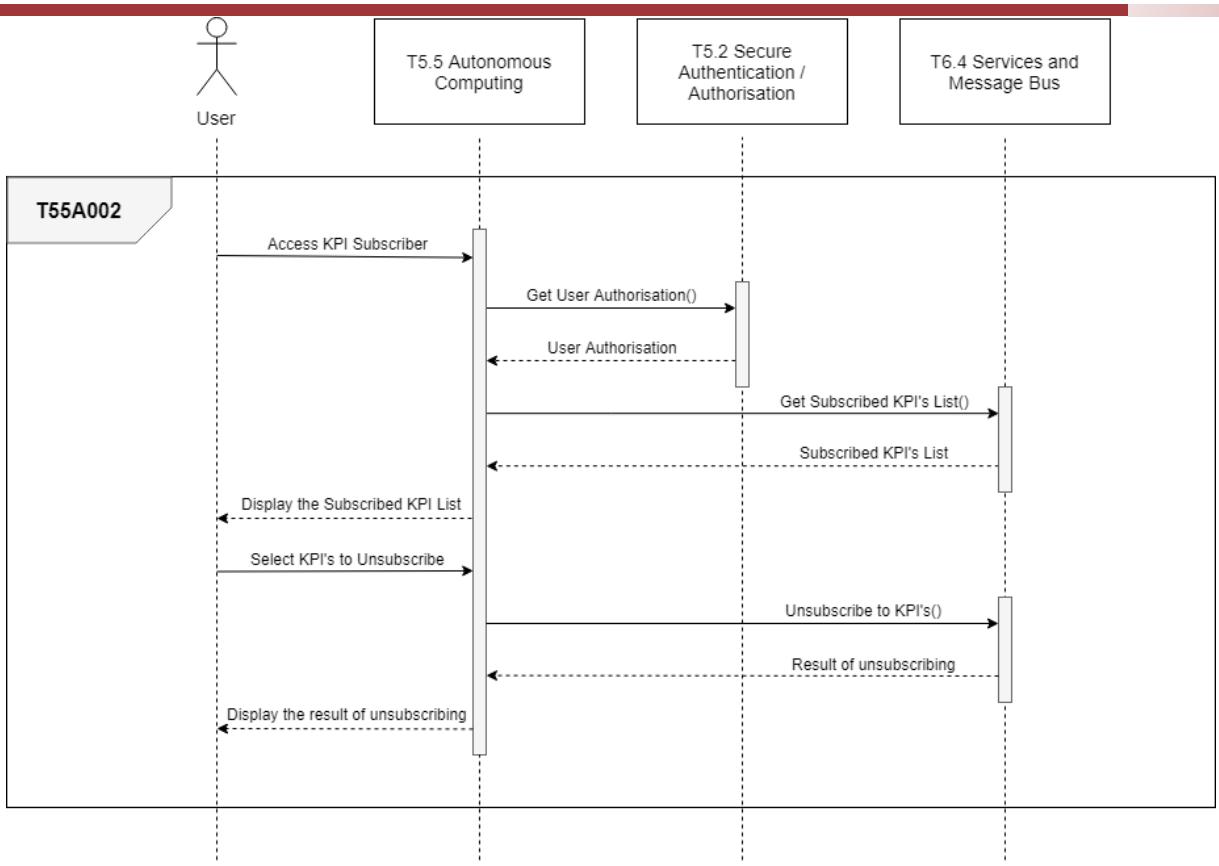


Figure 141: Retrieve subscribed KPI's and unsubscribe to it

5.4.3.3 Get KPI data

The Autonomous Computing component displays information about the subscribed KPI's, such as historic data, autonomous actions log, etc and as is described in the following figure.

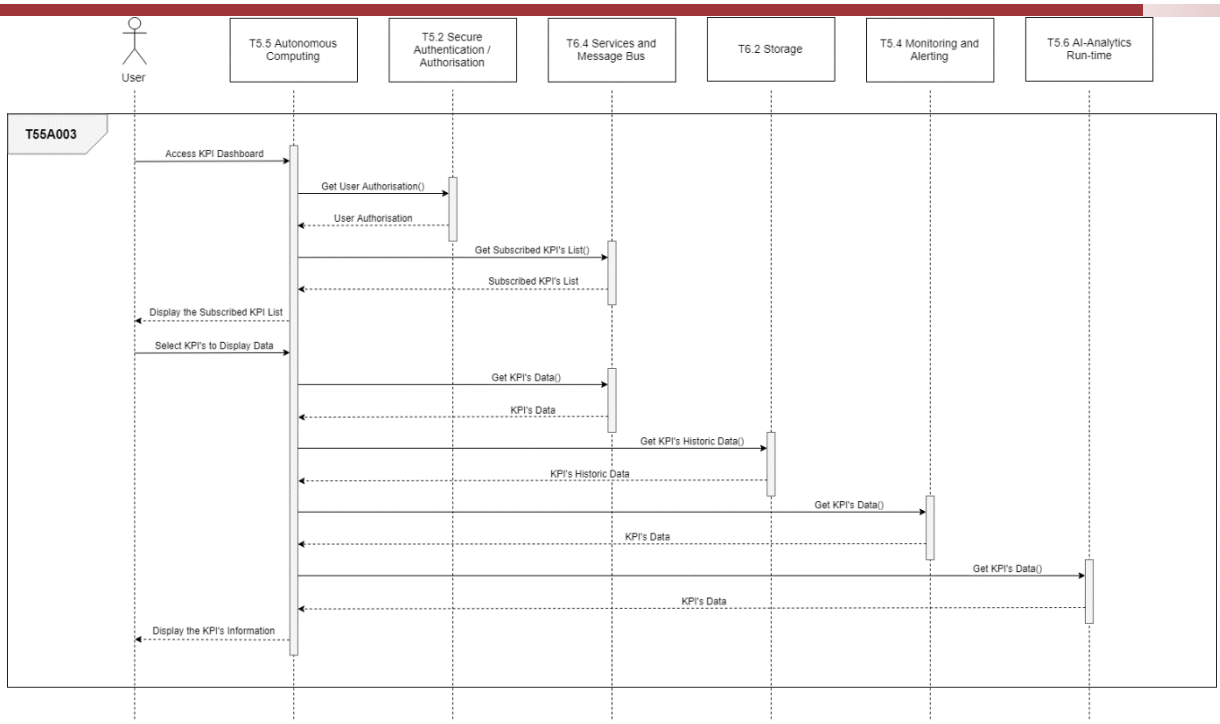


Figure 142: Display subscribed KPI's information

5.4.3.4 Define Conditions and Actions related to KPI

The Autonomous Computing component allows the definition of conditions and actions related to KPI's, so that an action is started whenever the KPI meets the conditions defined. The following figure describes the definition process, the processes of editing and deleting the conditions and actions will not be described in the diagram.

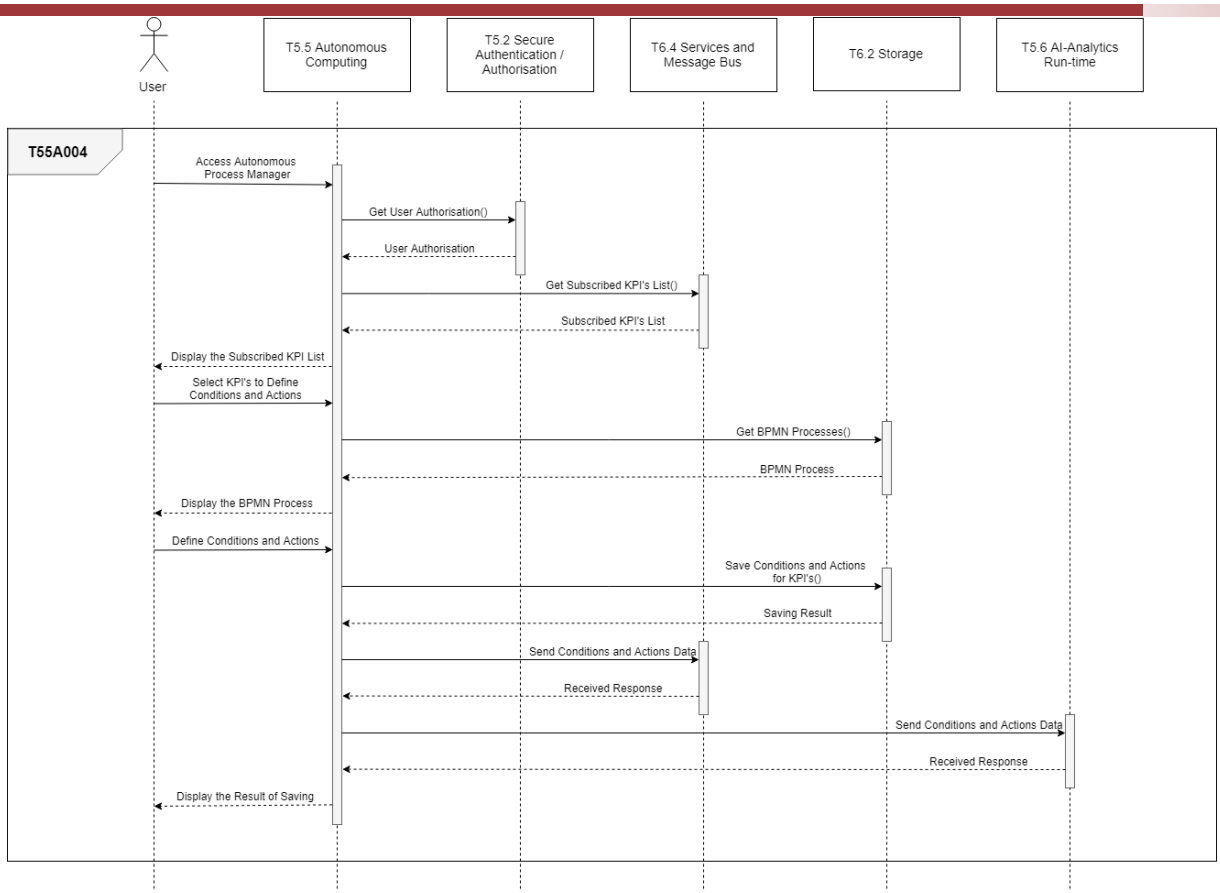


Figure 143: Display processes and define conditions and actions for the KPI

5.4.3.5 Get Conditions and Actions related to KPI

The Autonomous Computing component shall allow other ZDMP assets to obtain the definition of conditions and actions related to KPI's previously defined.

The main steps are:

- An asset requests the Conditions and Actions related to the KPI it specified
- The Autonomous Computing component obtains the list of Conditions and Actions related to the KPI
- The Autonomous Computing request to the Secure Authentication / Authorisation component the authorisations the requesting asset must see the KPI and its Conditions and Actions
- The Autonomous Computing return to the asset the Conditions and Actions it has authorisation to access from the specified KPI

5.4.3.6 Monitor KPI's values and Start Actions

The Autonomous Computing component monitors the KPI's values, so that an action is started whenever the KPI meets the conditions defined. The following figure describes this process.

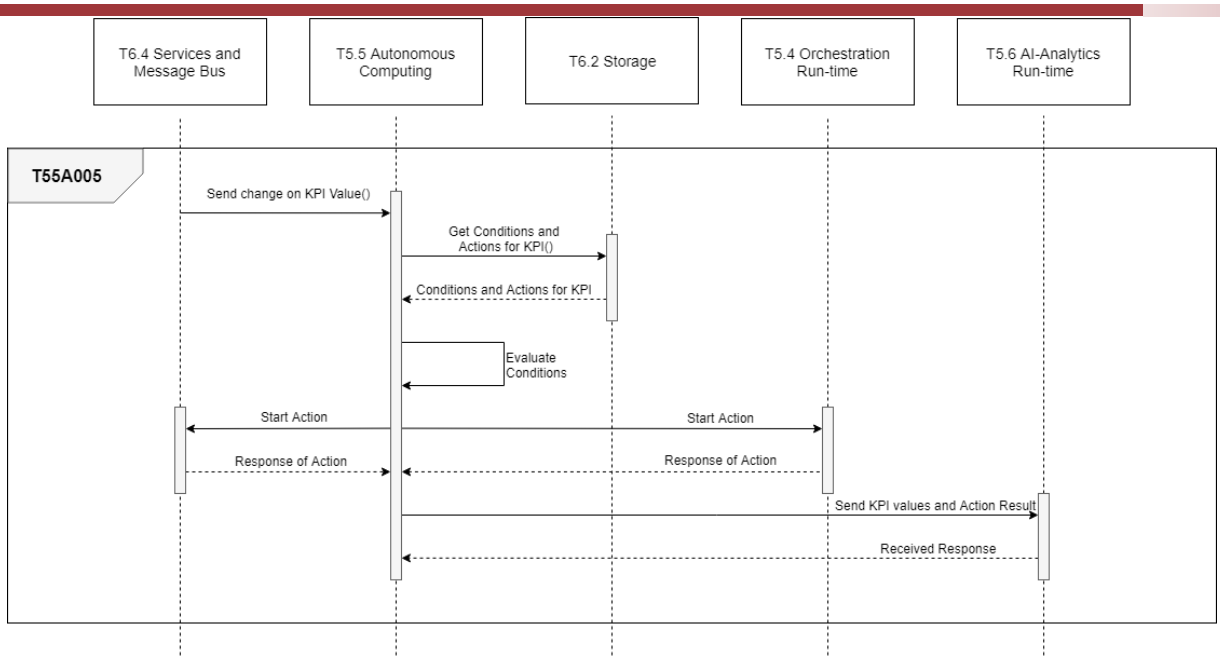
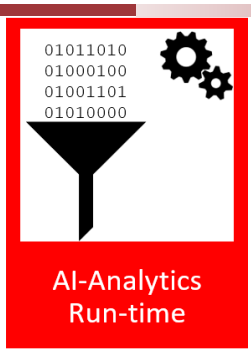


Figure 144: Monitor KPI's values and Start Actions



5.5 AI-Analytics Run-time (T5.6)

5.5.1 Overall functional characterisation & Context

The AI-Analytics Run-time component deploys machine learning models to the ZDMP platform where they can be run both offline and online using data from real-time production process. Thus, users can get in real-time notifications and alerts of some uncommon behaviour that could appear, and, in the meantime, ZDMP applications can adjust the production parameters to avoid defects.

5.5.2 Functions / Features

- **Model download**: Models created in AI-Analytics component and uploaded in Marketplace can be imported using the Deployer module, via Deployer API, to be run later against production data
- **Model run**: Models imported from Marketplace can be run by the Production Model Server, using online or offline data, to discover anomalies and to prevent failures
- **Production data feed**: The machine learning model needs real production data to run. These data can be pulled from running zApps installed on production assets or from Storage, via Analytics API
- **Notification issue**: In the machine learning running process, Production Model Server component can raise usual status messages and, when anomalies occurred can raise notifications and alerts. All messages are broadcasted to other ZDMP components via Notification API
- **Process Control**: Analytics UI has the role of allowing users to connect to Marketplace and fetch models, connect to zApps and Storage data sources, schedule and run machine learning models and display status, messages, and notifications
- **Analytics Visualisation**: The user can see numerical and/or graphical representations of machine learning running processes and to create aggregated queries and retrieve data in the form of reports (data tables, spreadsheet and/or CSV files) or graphic (charts, graphs), using Analytics UI and Visualisation modules
- **Analytics storage**: Data Warehouse module is the central repository for AI-Analytics Run-time component where all data used in machine learning running process can be stored for further running processes and analytics queries

These functions can be further decomposed into the following subtasks that have to be realised:

Subtask	Subtask description
T56B001 Connect to marketplace	**Priority:** Must
	Who: AI-Analytics Run-time **Where:** Anywhere **When:** Run-time **What:** Access Marketplace and open machine learning models page **Why:** So that a machine learning model can be imported
Acceptance Criteria	Application connected successfully
Requirements filled	N/A
T56B002	**Priority:** Must

Import model	**Who:** AI-Analytics Run-time **Where:** Anywhere **When:** Run-time **What:** Find machine learning model and import into Deployer module **Why:** So that the model can be run against production data
Acceptance Criteria	Model successfully downloaded
Requirements filled	N/A
T56B003 **Schedule model**	**Priority:** Must
	Who: AI-Analytics Run-time **Where:** Anywhere **When:** Run-time **What:** Select a model and program it for running at a specific date and time **Why:** So that the model can be run against production data
Acceptance Criteria	Model successfully scheduled
Requirements filled	RQ_0034, RQ_0035, RQ_0038, RQ_0045, RQ_0048, RQ_0051, RQ_0084, RQ_0095, RQ_0101, RQ_0103, RQ_0120, RQ_0137, RQ_0173, RQ_0174, RQ_0175, RQ_0178, RQ_0179, RQ_0180, RQ_0181, RQ_0189, RQ_0194, RQ_0201, RQ_0217, RQ_0218, RQ_0222, RQ_0305, RQ_0305, RQ_0307, RQ_0325, RQ_0329
T56B004 **Connect to zApps Data source**	**Priority:** Must
	Who: AI-Analytics Run-time **Where:** Anywhere **When:** Run-time **What:** Find a proper zApps and connect to it for access their data source **Why:** So that the model can run against real-time data provided by zApps
Acceptance Criteria	zApps successfully connected
Requirements filled	RQ_0034, RQ_0035, RQ_0038, RQ_0045, RQ_0048, RQ_0051, RQ_0084, RQ_0095, RQ_0101, RQ_0103, RQ_0120, RQ_0137, RQ_0173, RQ_0174, RQ_0175, RQ_0178, RQ_0179, RQ_0180, RQ_0181, RQ_0189, RQ_0194, RQ_0201, RQ_0217, RQ_0218, RQ_0222, RQ_0305, RQ_0305, RQ_0307, RQ_0325, RQ_0329
T56B005 **Connect to storage data source**	**Priority:** Must
	Who: AI-Analytics Run-time **Where:** Anywhere **When:** Run-time **What:** Connect to Storage to a specific data source **Why:** So that the model can run against offline data provided by Storage database
Acceptance Criteria	Storage successfully connected
Requirements filled	N/A
T56B006 **Run model**	**Priority:** Must
	Who: AI-Analytics Run-time **Where:** Anywhere **When:** Run-time **What:** Run a machine learning model against real production data **Why:** So that the anomalies can be discovered, and failures can be prevented
Acceptance Criteria	Model successfully run
Requirements filled	RQ_0034, RQ_0035, RQ_0038, RQ_0045, RQ_0048, RQ_0051, RQ_0084, RQ_0095, RQ_0101, RQ_0103, RQ_0120, RQ_0137, RQ_0173, RQ_0174, RQ_0175, RQ_0178, RQ_0179, RQ_0180, RQ_0181, RQ_0189, RQ_0194, RQ_0201, RQ_0217, RQ_0218, RQ_0222, RQ_0305, RQ_0305, RQ_0307, RQ_0325, RQ_0329
T56B007 **Model control UI**	**Priority:** Must
	Who: AI-Analytics Run-time **Where:** Anywhere **When:** Run-time **What**: Displays data, status and anomalies detected on a running model **Why:** So that the user can be informed about the running model
Acceptance Criteria	The information is successfully displayed
Requirements filled	RQ_0034, RQ_0035, RQ_0038, RQ_0045, RQ_0048, RQ_0051, RQ_0084, RQ_0095, RQ_0101, RQ_0103, RQ_0120, RQ_0137, RQ_0173, RQ_0174, RQ_0175, RQ_0178,

	RQ_0179, RQ_0180, RQ_0181, RQ_0189, RQ_0194, RQ_0201, RQ_0217, RQ_0218, RQ_0222, RQ_0305, RQ_0305, RQ_0307, RQ_0325, RQ_0329
T56B008 Notification issue	**Priority:** Must
	Who: AI-Analytics Run-time **Where:** Anywhere **When:** Run-time **What:** Send notifications other component about a running model **Why:** So that the ZDMP components can be notified about anomalies occurred
Acceptance Criteria	Notification successfully sent
Requirements filled	RQ_0034, RQ_0035, RQ_0038, RQ_0045, RQ_0048, RQ_0051, RQ_0084, RQ_0095, RQ_0101, RQ_0103, RQ_0120, RQ_0137, RQ_0173, RQ_0174, RQ_0175, RQ_0178, RQ_0179, RQ_0180, RQ_0181, RQ_0189, RQ_0194, RQ_0201, RQ_0217, RQ_0218, RQ_0222, RQ_0305, RQ_0305, RQ_0307, RQ_0325, RQ_0329
T56B009 Analytics queries	**Priority:** Must
	Who: AI-Analytics Run-time **Where:** Anywhere **When:** Run-time **What:** Creates analytics queries on data processed by the machine learning models. **Why:** So that the user can get the data computed with analytic functions.
Acceptance Criteria	Queries successfully created
Requirements filled	RQ_0034, RQ_0035, RQ_0038, RQ_0045, RQ_0048, RQ_0051, RQ_0084, RQ_0095, RQ_0101, RQ_0103, RQ_0120, RQ_0137, RQ_0173, RQ_0174, RQ_0175, RQ_0178, RQ_0179, RQ_0180, RQ_0181, RQ_0189, RQ_0194, RQ_0201, RQ_0217, RQ_0218, RQ_0222, RQ_0305, RQ_0305, RQ_0307, RQ_0325, RQ_0329
T56B010 Display graphs	**Priority:** Must
	Who: AI-Analytics Run-time **Where:** Anywhere **When:** Run-time **What:** Display graphs based on analytics queries **Why:** So that the user can see graphical representations of analytics data
Acceptance Criteria	Graphs successfully displayed
Requirements filled	RQ_0034, RQ_0035, RQ_0038, RQ_0045, RQ_0048, RQ_0051, RQ_0084, RQ_0095, RQ_0101, RQ_0103, RQ_0120, RQ_0137, RQ_0173, RQ_0174, RQ_0175, RQ_0178, RQ_0179, RQ_0180, RQ_0181, RQ_0189, RQ_0194, RQ_0201, RQ_0217, RQ_0218, RQ_0222, RQ_0305, RQ_0305, RQ_0307, RQ_0325, RQ_0329
T56B011 Analytics storage	**Priority:** Must
	Who: AI-Analytics Run-time **Where:** Anywhere **When:** Run-time **What:** Stores the data computed with analytics functions **Why:** So that a future representation of data is possible
Acceptance Criteria	Data successfully saved.
Requirements filled	RQ_0034, RQ_0035, RQ_0038, RQ_0045, RQ_0048, RQ_0051, RQ_0084, RQ_0095, RQ_0101, RQ_0103, RQ_0120, RQ_0137, RQ_0173, RQ_0174, RQ_0175, RQ_0178, RQ_0179, RQ_0180, RQ_0181, RQ_0189, RQ_0194, RQ_0201, RQ_0217, RQ_0218, RQ_0222, RQ_0305, RQ_0305, RQ_0307, RQ_0325, RQ_0329

Figure 145: AI Analytics Runtime Functions

5.5.3 Workflows

The following sub-sections describe the sequence diagrams of the AI-Analytics Run-time component.

5.5.3.1 Connect to marketplace

The following diagram explains this function and the necessary interactions with other components.

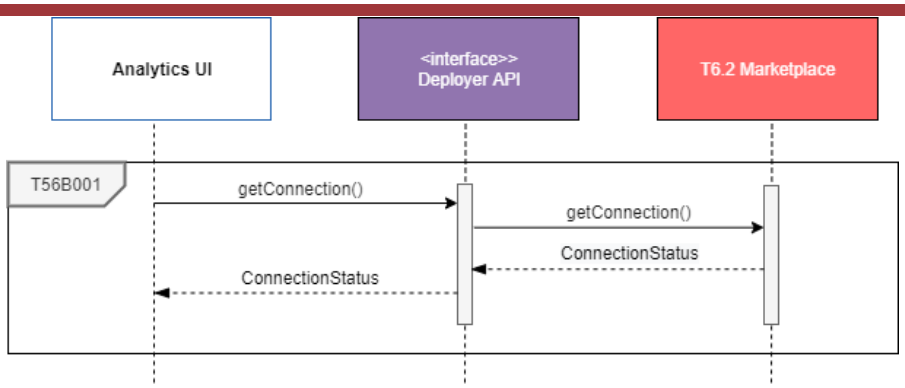


Figure 146: Connect to Marketplace sequence diagram

5.5.3.2 Import model

The following diagram explains this function and the necessary interactions with other components.

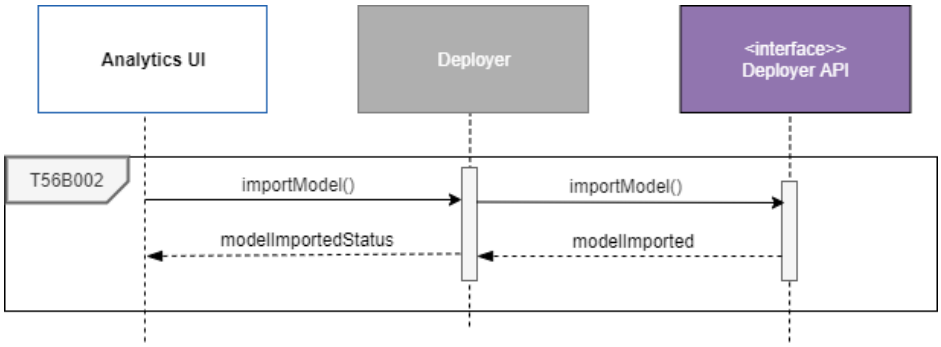


Figure 147: Import model sequence diagram

5.5.3.3 Connect to data sources

The following diagram explains this function and the necessary interactions with other components.

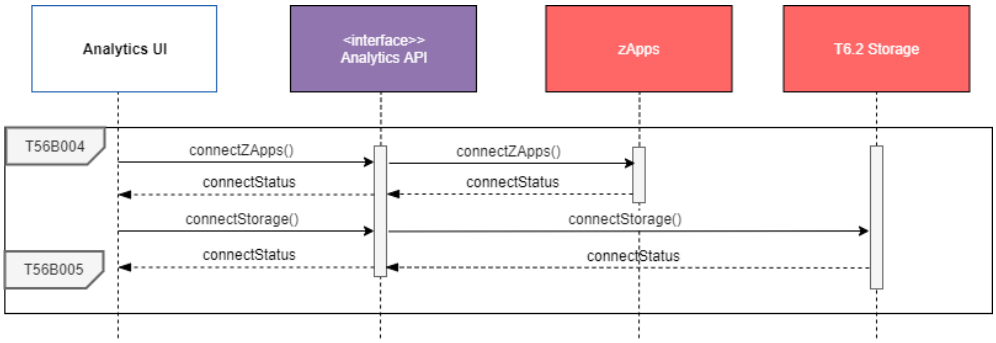


Figure 148: Connect to Marketplace sequence diagram

5.5.3.4 Schedule, run, control model and notifications

The following diagram explains this function and the necessary interactions with other components.

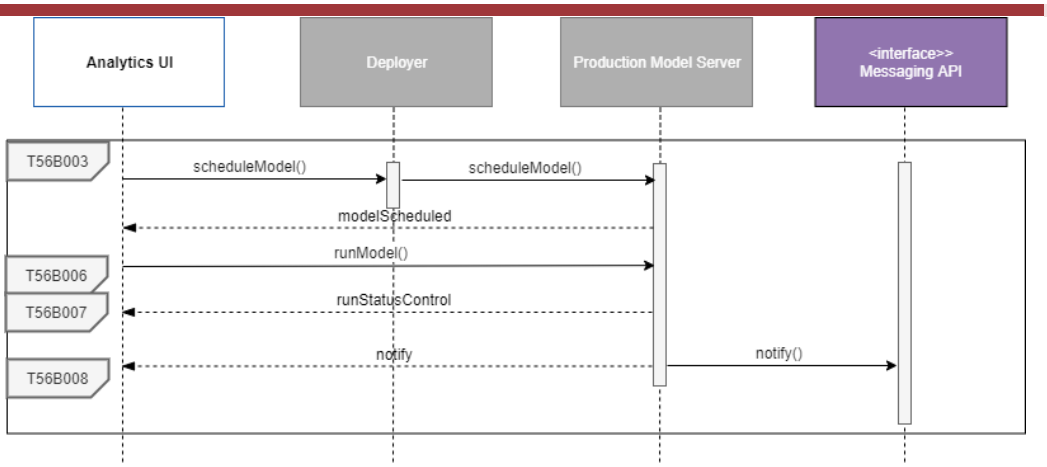


Figure 149: Schedule, run, control model and notifications sequence diagram

5.5.3.5 Analytics queries, display graphs and save data

The following diagram explains this function and the necessary interactions with other components.

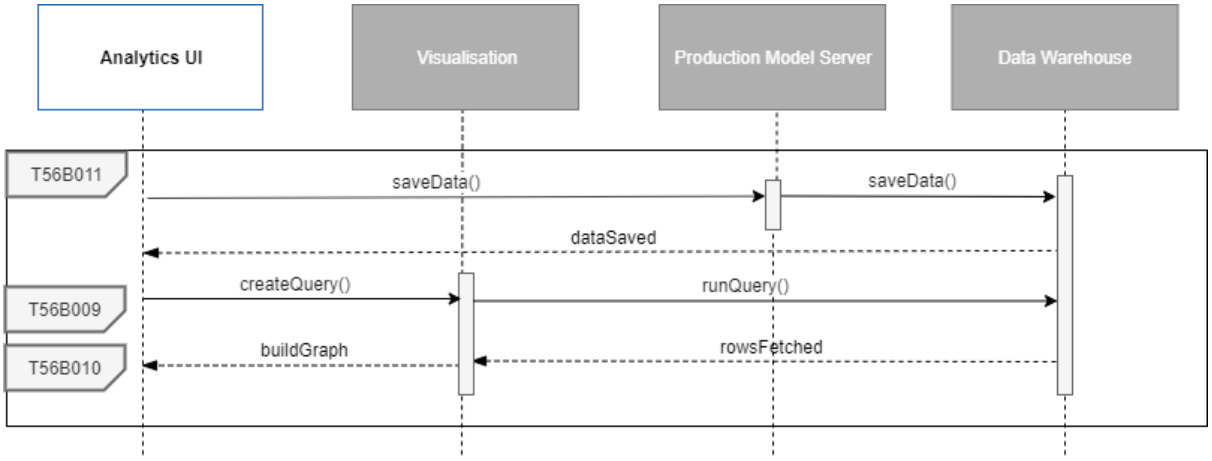
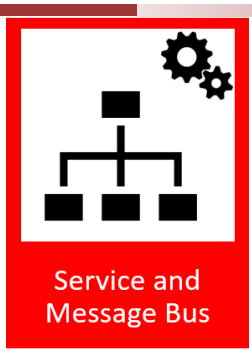


Figure 150: Analytics queries, display graphs and save data sequence diagram

5.6 Service and Message Bus (T6.4)



5.6.1 Overall Functional Characterisation & Context

The T6.4 Services and Message Bus component belongs to the "communication, storage or management related infrastructure". It represents the communication layer of the ZDMP Platform by providing a holistic communication for all services, end-user applications and components being used within ZDMP:

First, it enables ZDMP Assets to use other ZDMP Assets in a standardized and secure way. This is achieved through manageable REST APIs provided by a Services API Management which exposes specific services offered by the connected ZDMP Assets. These APIs can be fully customized, eg to only expose certain functions or to restrict the API access to specific entities.

Second, the T6.4 Services and Message Bus component provides ZDMP Assets with a message bus – a standardized communication interface to exchange messages, events, and data. This message bus implements a publish/subscribe messaging concept, which allows the connected ZDMP Assets to broadcast (publish) information on specific topics and to listen for certain events on these topics(subscribe).

Third, the T6.4 Services and Message Bus provides an alternative way to integrate ZDMP Assets and other components into the ZDMP Platform that could not be directly connected via a REST API and/or the message bus, eg because of proprietary protocols or missing service interfaces. This is achieved by implementing custom connectors that connect these sources to the Services API Management and/or the Message Bus via the Integration Server subcomponent. Note that the usage of the Integration Server subcomponent is optional and depends on the final use cases.

5.6.2 Functions / Features

The T6.4 Services and Message Bus provides the following high-level functions:

- **API Management:** The T6.4 Services API Management module allows ZDMP Platform and zApp developers to define and configure REST APIs to expose services provided by their ZDMP Assets. This also includes services provided by external platforms via the T6.5 Inter-platform Interoperability component.
- **API Access Control:** The T6.4 Services API Management module restricts the access to the provided APIs by using access tokens issued by the T5.2 Secure Authentication/Authorisation component. Through this, the APIs can only be accessed by ZDMP Assets that are authorized by the T5.2 Security Authentication/Authorisation component.
- **API Logging and Usage Statistics:** The T6.4 Services API Management logs and monitors the API calls for usage statistics, error reports and fraud detection.
- **Message Bus:** The T6.4 Services and Message Bus provides a Message Bus to enable ZDMP Assets to exchange information and data using the Publish/Subscribe Messaging concept.
- **Message Bus Access Control:** Like the T6.4 Services API Management, the T6.4 Message Bus restricts the access to its functionality via access tokens issued by the T5.2 Secure Authentication/Authorisation component. This ensures that only authorised ZDMP Assets get access to confidential information and data.

- **Integration of ZDMP Assets:** The T6.4 Services and Message Bus provides an optional Integration Server component to offer an alternative way for integrating ZDMP Assets that could not be directly connected to the T6.4 Services API Management or the T6.4 Message Bus.
- **Streaming Analytics:** The Services and Message Bus provides components to perform streaming analytics and complex event processing.

These high-level functions can be grouped to the following features, which must be realised:

Subtask	Subtask description
T64A001 **Management of APIs**	**Priority:** Must
	Who: ZDMP Platform Developers and zApp Developers **Where:** Anywhere **When:** Design time and during the lifecycle of ZDMP Assets **What:** Creation, configuration, and deletion of REST APIs **Why:** To expose services provided by ZDMP Assets via (parameterized) API functions to other ZDMP Assets in a standardised and secure way
Acceptance Criteria	The T6.4 Services API Management provides the functionality to create and delete individually configurable REST APIs. These APIs provide authorised ZDMP Assets with access to specific services provided by other ZDMP Assets via (parameterized) API functions. An API function definition includes • a predefined set of accepted parameters, • a service call that will be performed when the API function is called, and • a return value that is returned to the caller of the API function once the service call has been performed APIs as well as API functions can be changed and deleted at any time. The API management functionality provided by this feature can be used both via a graphical user interface and via a REST API
Requirements filled	RQ_0017, RQ_0030, RQ_0033, RQ_0066, RQ_0094, RQ_0099, RQ_0100, RQ_0160, RQ_0168, RQ_0202, RQ_0321, RQ_0322, RQ_0626, RQ_0679, RQ_0738, RQ_0797, RQ_0867
T64A002 **API access tokens**	**Priority:** Must
	Who: T6.4 Services API Management **Where:** Anywhere **When:** Design time, whenever a new API is created. Runtime, whenever an access token is re-issued or revoked by the T5.2 Secure Authentication / Authorisation component **What:** (De-)Registration of API access token issued by the T5.2 Secure Authentication/Authorisation component **Why:** To restrict the API access to only those ZDMP Assets that are authorised by the T5.2 Secure Authentication/Authorisation component
Acceptance Criteria	The T6.4 Services API Management stores individual access tokens for each managed API. These access tokens are requested from the T5.2 Secure Authentication/Authorisation component by the T6.4 Services API Management upon API creation and are required by ZDMP Assets to get access to the API's functions. The T6.4 Services API Management marks the API's access token as obsolete at the T5.2 Secure Authentication/Authorisation component if an existing API is deleted. Additionally, the T6.4 Services API Management replaces or deletes a stored access token if the original token is re-issued or revoked by the T5.2 Secure Authentication/Authorisation component
Requirements filled	N/A
T64A003 **API function calls**	**Priority:** Must
	Who: ZDMP Assets **Where:** Anywhere **When:** Runtime, whenever a ZDMP Asset needs the functionality of a service exposed through the T6.4 Services API Management and has a valid access token

	What: Calling a service via an API function **Why:** To perform a certain action that requires or uses the functionality provided by an exposed service
Acceptance Criteria	The API functions defined in the T6.4 Services API Management can be called from authorised ZDMP Assets. For authorisation, the ZDMP Assets must provide a valid access token issued by the T5.2 Secure Authentication/Authorisation component. Subsequently, the API function calls the underlying service and returns its result to the calling ZDMP Asset
Requirements filled	RQ_0017, RQ_0030, RQ_0033, RQ_0066, RQ_0094, RQ_0099, RQ_0100, RQ_0160, RQ_0168, RQ_0202, RQ_0321, RQ_0322, RQ_0626, RQ_0679, RQ_0738, RQ_0797, RQ_0867
T64A04 **API logging**	**Priority:** Must
	Who: T6.4 Services API Management **Where:** Anywhere **When:** Runtime, whenever an API is invoked **What:** Logging of API calls **Why:** To derive meta information about the individual APIs such as usage statistics and error reports. This information is used by the T5.4 Monitoring and Alerting component to monitor the execution of APIs and services.
Acceptance Criteria	The T6.4 Services API Management logs each API call including the following information: • Timestamps • Calling entity • Accessed function • Encountered errors
Requirements filled	N/A
T64A005 **API documentation**	**Priority:** Must
	Who: ZDMP Platform Developers and zApp Developers **Where:** Anywhere **When:** Design time and during the API lifecycle **What:** Documentation of APIs and API functions **Why:** To provide a description and details about the APIs and their functions including function parameters and return values
Acceptance Criteria	The T6.4 Services API Management provides the functionality to enter documentation for each API. This includes a textual description of the API and its functions as well as details about a function's parameters and return values. This documentation can be queried by the T5.4 Orchestration Designer, the T6.1 Application Builder and the T6.2 Marketplace
Requirements filled	N/A
T64A006 **(De-)Activation of APIs**	**Priority:** Should
	Who: ZDMP Platform Developers and zApp Developers **Where:** Anywhere **When:** Runtime, whenever APIs need to be (temporarily) switched off or on **What: (**De-)Activation of APIs and API functions **Why:** To (temporarily) prevent exposed services from being called by ZDMP Assets
Acceptance Criteria	APIs can be activated and deactivated. If an API is activated, its functions can be called by ZDMP Assets as described in T64A003. If an API is deactivated, an error message is returned to the calling ZDMP Asset stating that the API is (temporarily) not available
Requirements filled	N/A
T64A007 **Versioning of APIs**	**Priority:** Should
	Who: ZDMP Platform Developers and zApp Developers **Where:** Anywhere **When:** Design time and Runtime, whenever an API is changed **What:** Creation, deletion, and modification of different API versions

	Why: To provide different feature sets per API version and to maintain downward compatibility over the lifecycle of an API
Acceptance Criteria	Different versions of the same API can be created, deleted, and modified. Each API version is tagged with a unique identifier and can be managed independently from other versions of the same API
Requirements filled	N/A
T64A008 Mocking of APIs	**Priority:** Should
	Who: ZDMP Platform Developers and zApp Developers **Where:** Anywhere **When:** Design time, whenever the functionality of an API function must be evaluated **What:** Definition of predefined responses (return values) for all functions provided by an API and corresponding parameterisations **Why:** To evaluate and debug the functionality of the API functions under specific parametrizations and conditions
Acceptance Criteria	For each function provided by an API and corresponding parameterisations, a specific mocked response can be defined. If defined, this mocked response is returned instead of the result of the real service call whenever the corresponding API function and parameterisation is called
Requirements filled	N/A
T64B001 Message Bus access tokens	**Priority:** Must
	Who: T6.4 Message Bus **Where:** Anywhere **When:** On system boot and whenever an access token is re-issued or revoked by the T5.2 Secure Authentication/Authorisation component **What:** (De-)Registration of Message Bus access tokens issued by the T5.2 Secure Authentication/Authorisation component **Why:** To restrict the access to the Message Bus API to only those ZDMP Assets that are authorised by the T5.2 Secure Authentication/Authorisation
Acceptance Criteria	The T6.4 Message Bus stores individual access tokens to authorise ZDMP Assets to access the Message Bus's functions provided by the T6.4 Message Bus API. These access tokens are requested from the T5.2 Secure Authentication/Authorisation component by the T6.4 Message Bus • on system boot to create a token pair that grant access to the topic management API functions (create & delete) • upon creation of new topics to create a token pair that grant access to the publish and subscribe API functions of this new topic The T6.4 Message Bus marks a topic-specific token pair as obsolete at the T5.2 Secure Authentication/Authorisation component if a topic is deleted. Additionally, the T6.4 Message Bus replaces or deletes a stored access token or pair if the original token or the pair is re-issued or revoked by the T5.2 Secure Authentication/Authorisation component
Requirements filled	N/A
T64B002 Message Bus Topic Management	**Priority:** Must
	Who: ZDMP Assets **Where:** Anywhere **When:** Runtime, whenever a ZDMP Asset needs to create a new topic or delete an existing one **What:** Creation and deletion of Message Bus Topics **Why:** to define and delete topics • on which ZDMP Assets can publish information and, • to which ZMDP Assets can subscribe to receive information
Acceptance Criteria	The T6.4 Message Bus provides the functionality to define and delete topics. Each topic must have a unique name, ie no duplicates are allowed. A topic is represented as a text string. This string may contain forward slashes ("/") to indicate different topic levels.

	The topic management functions can only be accessed by authorised ZDMP Assets that provide valid, function-specific (create, delete) access tokens issued by the T5.2 Secure Authentication/Authorisation component as described in function T64B001.
Requirements filled	RQ_0030, RQ_0099, RQ_0100, RQ_0202, RQ_0321, RQ_0322, RQ_0626, RQ_0679, RQ_0738, RQ_0797, RQ_0867
T64B003 Publishing messages on topics and subscribing to topics	**Priority:** Must
	Who: ZDMP Assets **Where:** Anywhere **When:** Runtime, whenever information must be broadcasted to other ZDMP Assets and whenever ZDMP Assets want to listen for information and certain events **What:** Publishing of messages incl. payloads on certain topics and subscribing to specific topics to receive messages and their payload **Why:** to broadcast information to other ZDMP Assets that are subscribed to a specific topic
Acceptance Criteria	The T6.4 Message Bus allows authorised ZDMP Assets to broadcast (publish) messages and data on certain topics and to receive messages and data by subscribing to relevant topics. For authorisation, the ZDMP Assets must provide valid, function- and topic-specific access tokens (publish, subscribe) issued by the T5.2 Secure Authentication/Authorisation component. The T6.4 Message Bus validates this access token by comparing it with the access token stored for the intended action (publish or subscribe) and topic. If the provided token is valid, the intended action is performed. Otherwise, an error message is returned to the ZDMP Asset.
Requirements filled	RQ_0030, RQ_0099, RQ_0100, RQ_0202, RQ_0321, RQ_0322, RQ_0626, RQ_0679, RQ_0738, RQ_0797, RQ_0867
T64C001 Optional integration of Enterprise Tier, Platform Tier, and Edge Tier components	**Priority:** Should
	Who: Platform Developers **Where:** Anywhere **When:** Design time **What:** Integration of ZDMP Assets and other components using Integration Connectors **Why:** To integrate components that could not be connected otherwise to the T6.4 Services API Management and the T6.4 Message Bus, eg because of unsupported service interfaces
Acceptance Criteria	The T6.4 Integration Server provides the functionality to integrate ZDMP components and modules via Integration Connectors. These connectors connect the integrated component to the internal communication bus of the T6.4 Integration Server to: • Expose the integrated components as REST services via the Services API Management • Connect the integrated components to the T6.4 Message Bus
Requirements filled	N/A
T64D001 Complex Event Processing	**Priority:** Should
	Who: Platform Developers and zApp Developers **Where:** Anywhere **When:** Design time **What:** Definition of Complex Event Processing rules and actions **Why:** To perform certain actions when specific events or messages are received via the T6.4 Message Bus
Acceptance Criteria	The T6.4 Complex Event Processing processes and analyses low-level events such as sensor data received on the T6.4 Message Bus and triggers specific actions when certain conditions are met, or rules are fulfilled. These conditions, rules and actions can be defined by Platform Developers and zApp Developers.
Requirements filled	N/A

Figure 151: Service and Message Bus Functions

5.6.3 Workflows

5.6.3.1 API Management

This feature enables ZDMP Platform Developers and zApp Developers to create, delete and configure REST APIs and API functions to expose services provided by their ZDMP Assets to other ZDMP Assets (T64A001).

The general workflow for all interactions covered by feature T64A001 is as follows:

- The ZDMP Platform Developers and zApp Developers interact with the T6.4 Services API Management either via the T6.4 API Management UI or directly via a REST API
- The T6.4 Services API Management performs the API management action and either returns a success or an error message depending on the result of the performed action
- If used, the T6.4 API Management UI shows the updated states of the user's APIs

Additionally, the access to the different APIs is controlled via access tokens issued and maintained by the T5.2 Secure Authentication/Authorisation component (T64A002). See

5.6.3.2 Creation of REST APIs

ZDMP Platform Developers and zApp Developers can create REST APIs at the T6.4 Services API Management. This allows them to expose services provided by their newly developed and deployed ZDMP Assets. APIs can be either created from scratch or imported using the Swagger, RAML, or WSDL format.

Creating an API triggers the following actions (Figure 152 – T64A001/2a):

- The T6.4 Services API Management component instantiates a new API using the parameters provided by the user (API name, API functions, etc). If any API function is specified, it is created according to the workflow described below
- The created API is assigned a unique ID and is stored in the T6.4 Services API Management API registry
- The T6.4 Services API Management requests an access token for the created API from the T5.2 Secure Authentication/Authorisation component. This token is stored in the T6.4 Services API Management registry for authorisation checks

5.6.3.3 Creation of REST API functions

To expose individual services provided by their ZDMP Assets, ZDMP Platform Developers and zApp Developers must define parameterised API functions within an existing API. This includes the definition of parameter types, return value types and the service endpoint of the ZDMP Asset exposed by the API function (Figure 152 – T64A001/2b).

5.6.3.4 Deletion of APIs and API functions

ZDMP Platform Developers and zApp Developers can delete whole APIs or individual API functions at any time. This triggers the following actions (Figure 152 – T64A001/2c):

- Depending on the use case, the T6.4 Services API Management either deletes the specified API from its API registry (including all API functions) or removes the

specified API function from its parent API. As a result, the deleted functions can be no longer accessed
- If an API is deleted, the access token stored for this API is removed from the T6.4 Services API Management registry and marked as obsolete at the T5.2 Secure Authentication/Authorisation component

5.6.3.5 Modification of API functions

ZDMP Platform Developers and zApp Developers can modify existing API functions. Once submitted, the T6.4 Services API Management applies the changes to the specified API function.

5.6.3.6 Re-issuing and revocation of access tokens

The T5.2 Secure Authentication/Authorisation component can re-issue, or revoke issued API access tokens at any time. This triggers the following actions:

- If an access token is re-issued, the existing access token in the T6.4 Services API Management registry is immediately replaced by the new token. Afterwards, ZDMP Assets must provide the new access token to get access to the corresponding API
- If an access token is revoked, the T6.4 Services API Management registry removes the token from its registry. This results in the corresponding API cannot be accessed by any ZDMP Asset until a new access token is issued

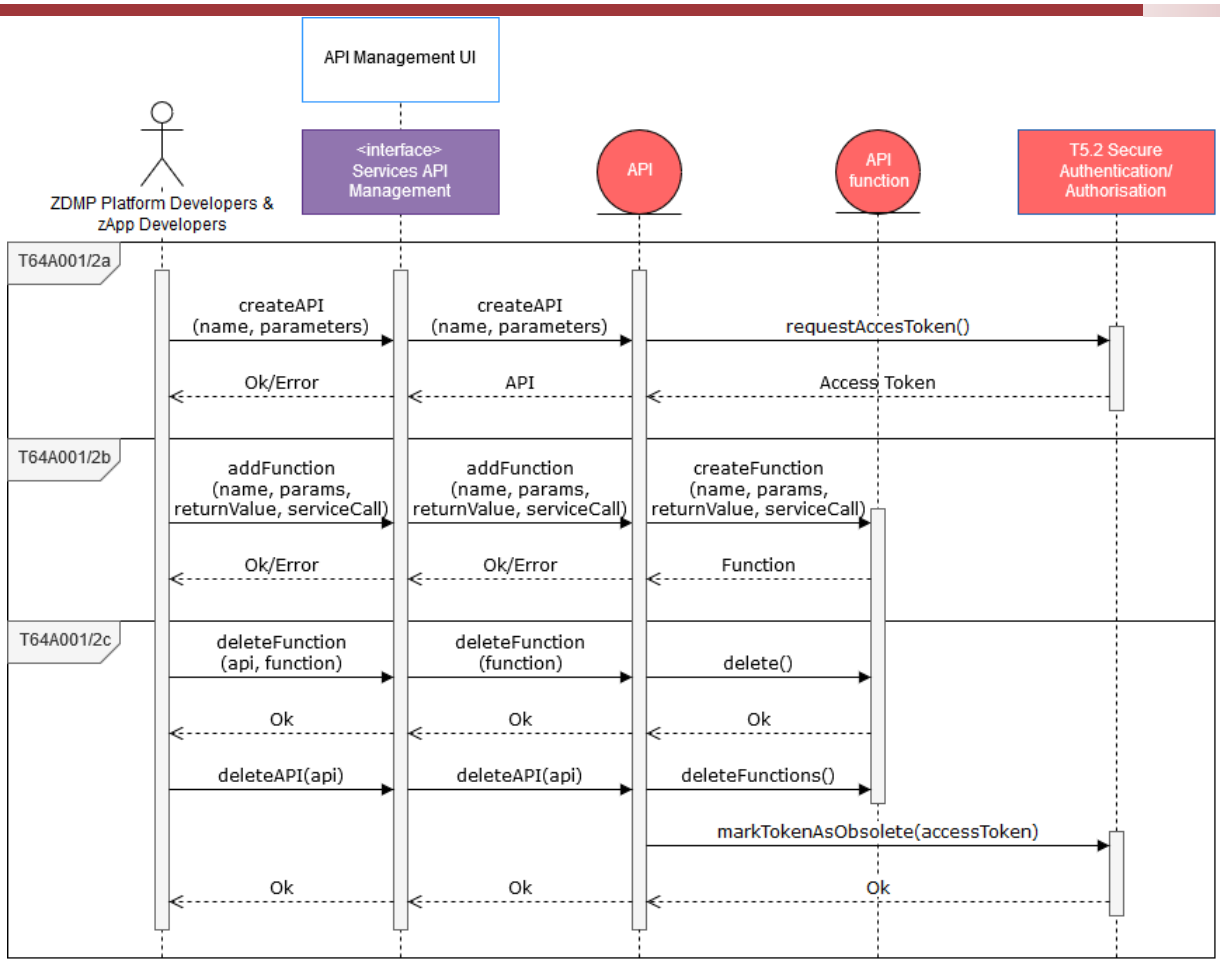


Figure 152: Sequence diagram showing the workflows for creating and deleting APIs and API functions

5.6.3.7 API function calls

This feature enables ZDMP Assets to call API functions provided by the T6.4 Services API Management to access the ZDMP resources exposed by these individual functions (T64A003). To call a specific API function, a ZDMP Asset must provide a valid access token issued by the T5.2 Secure Authentication/Authorisation component for this API.

When a ZDMP Asset calls an API function, the following actions are triggered (see Figure 153):

- The T6.4 Services API Management validates the access token provided by the ZDMP Asset by comparing it to the token stored for the corresponding API. If the access token is invalid, an error message is returned to the calling ZDMP Asset and the further workflow is stopped
- The T6.4 Services API Management executes the API function using the parameters provided by the calling ZDMP Asset. If the parameters are invalid, an error message is returned to the calling ZDMP Asset and the further workflow is stopped. Otherwise, the underlying parameterized service is called as defined in the API function definition
- Once the result of the service call is returned to the T6.4 Services API Management, the result is optionally transformed as defined in the API function. Subsequently, the (transformed) result is sent back to the calling ZDMP Asset

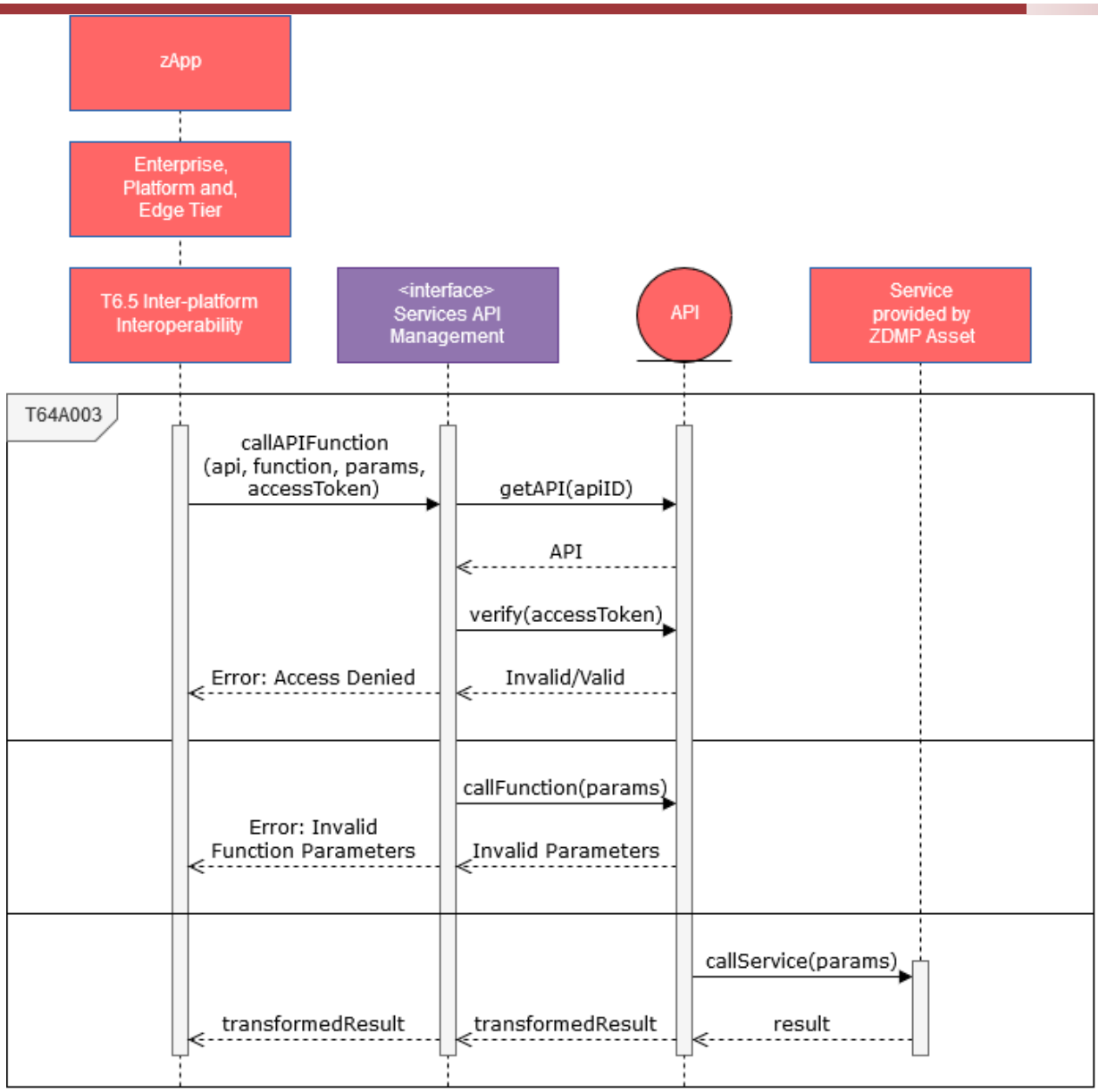


Figure 153: Sequence diagram showing the API function call workflow

5.6.3.8 API logging

This feature enables the T6.4 Services API Management to log the execution of API calls to acquire and derive meta information of APIs such as usage statistics and error reports (T64A004).

Whenever an API function is called with API logging enabled, the T6.4 Services API Management logs the following information:

- Timestamp
- Calling entity
- Accessed function
- Encountered errors

5.6.3.9 Activation and Deactivation of APIs

This feature enables ZDMP Platform Developers and zApp Developers to (temporarily) deactivate and activate their APIs (T64A006), eg for maintenance purposes. This triggers the following actions:

- If an API is deactivated, the T6.4 Services API Management completely disables the access to the deactivated API and its functions. If a function of a deactivated API is called, an error message is sent back to the calling ZDMP Asset stating that the API is (temporarily) not available
- If an API is activated, its API functions can be accessed again as normal (ie, as described in T64A003)

5.6.3.10 API Versioning

This feature enables ZDMP Platform Developers and zApp Developers to create, delete and modify different versions of their APIs (T64A007).

5.6.3.11 Creation of a new API version

When ZDMP Platform Developers and zApp Developers create a new version of one of their APIs, the following actions are triggered:

- The T6.4 Services API Management selects the current version of the specified API from its API registry
- The T6.4 Services API Management creates a latest version of this API by cloning the entire API configuration (API function definitions, documentation, etc) and assigning an increased version number
- The new API version runs in parallel with the other versions of the same API. ZDMP Assets also must specify the API version when calling API functions

5.6.3.12 Deletion of an API version

ZDMP Platform Developers and zApp Developers can delete specific versions of their APIs at any time. This triggers the following actions:

- The Services API Management selects the specified API version
- The Services API Management removes the specified API from its API registry
- If a ZDMP Asset calls an API function of a deleted API version, an error message is return the calling ZDMP Asset stating that the API version is no longer available

5.6.3.13 Modification of an API version

When ZDMP Platform Developers and zApp Developers modify one of their APIs, the following actions are triggered:

- The T6.4 Services API Management selects the specified API version from its registry
- The Services API Management applies the specified changes to the selected API version
- The changes take effect immediately

5.6.3.14 API Mocking

This feature enables ZDMP Platform Developers and zApp Developers to create mocked responses for their API functions (T64A008), eg for testing or debugging purposes. The workflow representing this feature is as follows:

- The Developer defines the mocked response for a specific API function and parameterisation
- The T6.4 Services API Management activates the mocking functionality for the specified API function and stores the mocked response
- When the function is called with the corresponding parameterisation, the stored mocked response is returned to the caller of the API function instead of calling the real service and returning its result

5.6.3.15 Message Bus access tokens

The T6.4 Message Bus maintains a registry of access tokens to authorise the access to its API functions. These tokens are requested from the T5.2 Secure Authentication / Authorisation component at runtime as described below.

5.6.3.16 Access tokens for creating and deleting topics

The T6.4 Message Bus stores two different access tokens to verify if a ZDMP Asset is authorised to create or delete topics. This token pair is requested from the T5.2 Secure Authentication/Authorisation component on system boot.

5.6.3.17 Access tokens for publishing on topics and subscribing to topics

Additionally, the T6.4 Message Bus stores two different access tokens for each individual topic. These tokens are used to verify if a ZDMP Asset is authorised to publish messages on a specific topic or to subscribe to a specific topic. This token pair is requested from the T5.2 Secure Authentication/Authorisation component upon topic creation (see Section Message Bus Interaction).

5.6.3.18 Re-issuing and revocation of access tokens

The T5.2 Secure Authentication/Authorisation component can re-issue, or revoke issued Message Bus access tokens at any time. This triggers the following actions:

- If an access token is re-issued, the existing access token in the T6.4 Message Bus registry is immediately replaced by the new token. Afterwards, ZDMP Assets must provide the new access token to get access to the corresponding API function
- If an access token is revoked, the T6.4 Message Bus registry removes the token from its registry. This results in the corresponding API function cannot be accessed by any ZDMP Asset until a new access token is issued

5.6.3.19 Message Bus interaction

The T6.4 Services and Message Bus component implements a Message Bus module. It enables ZDMP Assets to exchange information and data using the Publish/Subscribe messaging concept (T64B001 and T64B002).

5.6.3.20 Creation of topics

ZDMP Assets can create new topics on-the-fly via the T6.4 Message Bus API. Each topic must be unique, ie no duplicates are allowed. Additionally, the ZDMP Asset must be authorised to create a topic, which is verified by providing an access token issued by the T5.2 Secure Authentication/Authorisation component. The creation of a new topic is represented by the following workflow (see Figure 154 - T64B002a):

- A new topic is defined in the form of a text string. This string can be optionally separated by topic levels indicated by a forward slash (eg production/data/sensor_42).
- The ZDMP Asset calls the API function to create a new topic and passes the topic as well as an access token as function arguments
- The T6.4 Message Bus validates the provided access token. If the token is invalid, an “access denied” error message is returned
- The T6.4 Message Bus checks if the specified topic already exists. In this case, a “duplicate topic” error message is returned
- The T6.4 Message Bus creates the specified topic. Afterwards, it requests two access tokens from the T5.2 Secure Authentication/Authorisation component and stores them in its registry. The first access token is used to authorise ZDMP Assets to publish messages on the new topic, while the second token is required to subscribe to the new topic

5.6.3.21 Deletion of topics

Authorised ZDMP Assets can delete topics at any time via the T6.4 Message Bus API. The authorisation is verified by providing an access token issued by the T5.2 Secure Authentication/Authorisation component. The deletion of a topic triggers the following actions (see Figure 154 - T64B002b):

- The ZDMP Asset calls the API function to delete a specific topic and passes the topic as well as an access token as function arguments
- The T6.4 Message Bus validates the provided access token. If the token is invalid, an “access denied” error message is returned
- The T6.4 Message Bus checks if the specified topic exists. If not, an error message is returned
- The T6.4 Message Bus deletes the specified topic. As a result, publishing messages on the deleted topic is no longer possible and the subscribed ZDMP Assets no longer receive messages on the deleted topic
- The T6.4 Message Bus marks the access tokens stored for the deleted topic as obsolete at the T5.2 Secure Authentication/Authorisation component

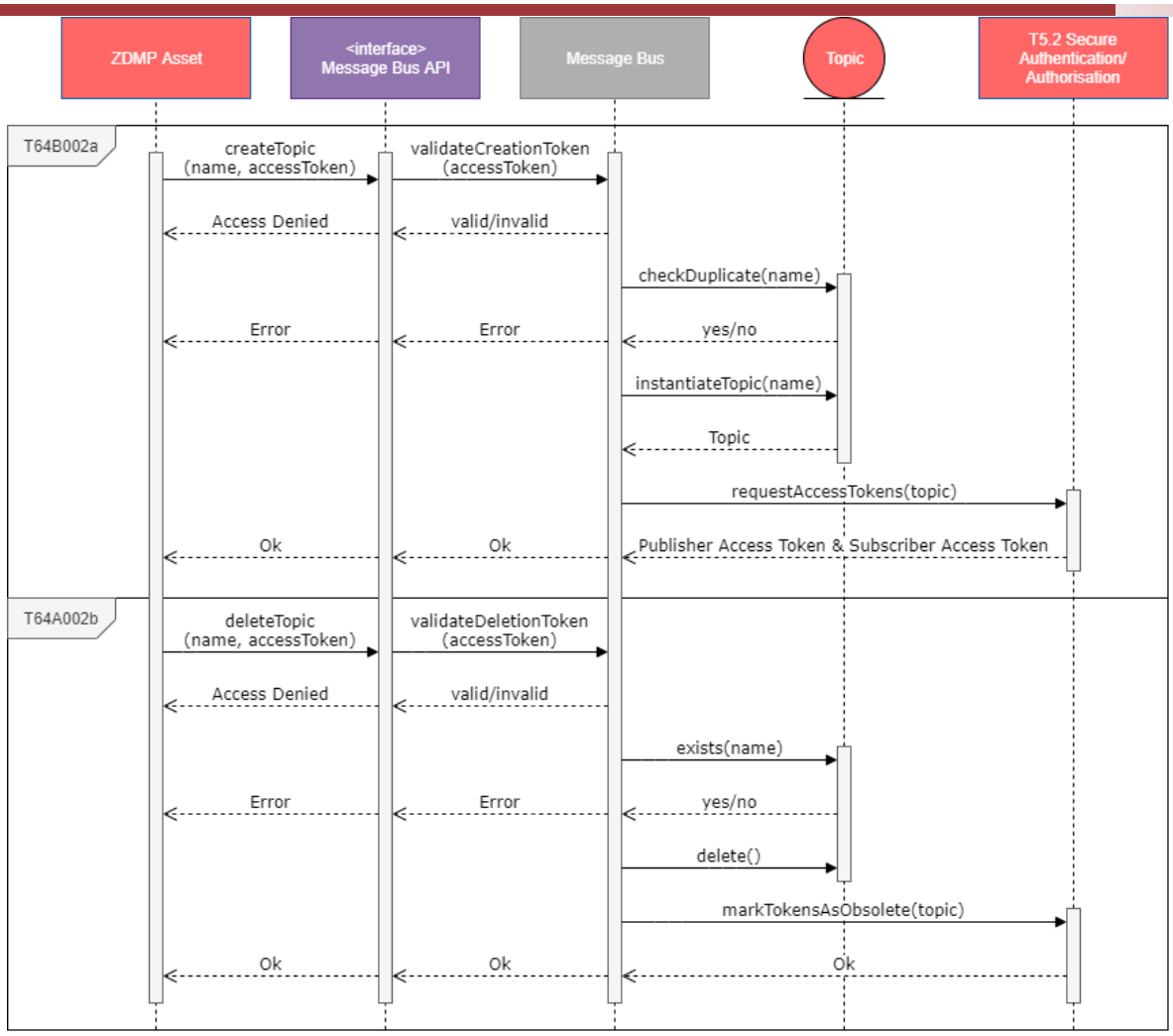


Figure 154: Sequence diagram illustrating the creation and deletion of topics

5.6.3.22 Subscribing to topics

ZDMP Assets can subscribe to existing topics at any time via the T6.4 Message Bus API. Afterwards, they can receive messages that are published on the subscribed topic. Subscribing to a specific topic triggers the following actions (see Figure 154 - T64B003a):

- The ZDMP Asset calls the corresponding T6.4 Message Bus API function and provides the desired topic and the corresponding subscriber access token as function arguments
- The T6.4 Message Bus checks if the specified topic exists. If not, an error message is returned, and the further workflow is stopped
- The T6.4 Message Bus checks if the ZDMP is authorised to subscribe to the specified topic by validating the provided access token. If the token is invalid, an error message is returned, and the further workflow is stopped
- The T6.4 Message Bus subscribes the ZDMP Asset to the specified topic. Subsequently, the ZDMP Asset receives messages published on the specified topic

5.6.3.23 Publishing messages on topics

ZDMP Assets can publish messages on existing topics at any time via the T6.4 Message Bus API. These messages are subsequently broadcasted to all ZDMP Assets that are

subscribed to the specified topic. Publishing messages on a specific topic triggers the following actions (see Figure 154 - T64B003b):

- The ZDMP Asset calls the corresponding T6.4 Message Bus API function and provides the desired topic, the corresponding publisher access token, and the message as function arguments
- The T6.4 Message Bus checks if the specified topic exists. If not, an error message is returned, and the further workflow is stopped
- The T6.4 Message Bus checks if the ZDMP is authorised to publish messages on the specified topic by validating the provided access token. If the token is invalid, an error message is returned, and the further workflow is stopped
- The T6.4 Message Bus publishes the provided message on the specified topic by broadcasting it to all subscribed ZDMP Assets

5.7 Prediction and Optimisation Run-time (WP7)

5.7.1 Overall functional characterization & Context

This component behaves either as a predictor or an optimiser. The choice between "predictor" and "optimiser" is based on the specialization that it is designed for by the component "Prediction and Optimisation (Design Time)". It provides any component (specifically "Quality Assurance") or zApp with predictions or optimised values. The exact algorithm along with the numerical code is defined and bundled into this component by the design time component ("Prediction and Optimisation (Design Time)"). The run-time relevant parameters of the components are configurable. The source of the input to the prediction model or optimisation algorithm is determined by the user of the component.

5.7.2 Functions / Features

- **Configure**: The user can send configuration commands through the user interface, or an external component via the API of the component. The run-time relevant parameters of the components are configurable. Some examples of configuration are: 1- Setup of adjustable parameters of the optimiser or predictor 2- constraining the selected inputs to the optimiser or predictor.
- **Select the source**: The user can select the source type. The source can be a message bus topic or a storage location.
- **Specify the input channels**: Specifies the name of the input channels from the source that determines which sensor or parameter should be used for each input of the model.
- **Perform optimisation**: User/Caller sends the request for optimisation results. This request will produce one output (one set of optimised parameters) per call. If the component is not specialized for optimisation it will return an error code.
- **Perform prediction**: User/Caller sends the request for prediction results.
- **Check for update**: Checks for new versions of this component on Marketplace. In case of existence of new version, it downloads it and replaces it with the previous version. The new version primarily means a new algorithm, a new trained prediction model, a new input pre-processing pipeline, new inputs to the numerical kernel, or in the case of optimiser, a new objective function.

Subtask	Subtask description
T7123A001 **Select the source**	**Priority:** Must
	Who: Process Engineer or an external connected utilizer of the component **Where:** wherever there is access to message bus or storage or the user component can have access to this component **When:** After “configure” (if caller prefers to change the default configurations) **What:** Select the source of input to be whether Message bus or Storage **Why:** To provide the source to the numerical kernel
Acceptance Criteria	The component responds with OK if the source of input is known to the component.
Requirements filled	RQ_0067, RQ_0070, RQ_0118
T7123A002	**Priority:** Must

Specify the input channels	**Who:** Process Engineer or an external connected utilizer of the component **Where:** With access to message bus, storage, or external user with access **When:** After “select the source” **What:** Specifies the name of the sensors, parameters etc, from the source **Why:** To provide the input to the numerical kernel
Acceptance Criteria	The component responds with OK if the input channels are known to the component.
Requirements filled	RQ_0067, RQ_0114
T7123A003 Perform optimisation	**Priority:** Must
	Who: Process Engineer or an external connected utilizer of the component **Where:** With access to message bus, storage, or external user with access **When:** After “specify the input channels” **What:** Calls the “optimiser” with necessary input. **Why:** To have the optimised values and send it to the caller.
Acceptance Criteria	The optimisation result is received caller
Requirements filled	RQ_0054, RQ_0098
T7123A004 Perform prediction	**Priority:** Must
	Who: Process Engineer or an external connected utilizer of the component **Where:** With access to message bus, storage, or external user with access **When:** After “specify the input channels” **What:** Calls the “predictor” with necessary input. **Why:** To generate the predicted values for various uses
Acceptance Criteria	The prediction result is received by the caller
Requirements filled	RQ_0087, RQ_0118
T7123A005 Configure	**Priority:** Must
	Who: User with machine learning experience or an external user **Where:** With access to message bus, storage, or external user with access **When:** The first action when using this component **What:** Configures the run-time relevant parameters of the component **Why:** needed for the component to function properly or as desired
Acceptance Criteria	If the necessary parameters for functioning of the component are valid and configuration is successful, the component replies with OK
Requirements filled	RQ_0067, RQ_0086, RQ_0114
T7123A006 Check for updates	**Priority:** Medium
	Who: Update manager **Where:** With access to message bus, storage, or external user with access **When:** Any time **What:** Checks the status of updates to the component on Marketplace **Why:** to have the latest version of the component (updated functionality, algorithm, or training)
Acceptance Criteria	Receives 1- the information about the latest version on Marketplace 2- latest version of the component
Requirements filled	N/A

Figure 155: Prediction and Optimization Runtime

5.7.3 Workflows

5.7.3.1 Initialization

The following diagram explains the initialization functions and the necessary interactions with other components.

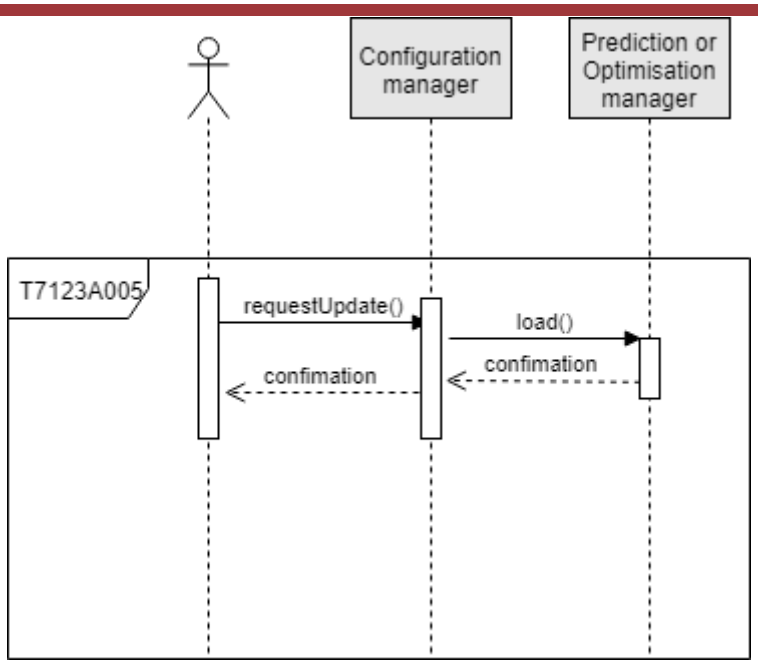


Figure 156: Configuration Sequence Diagram

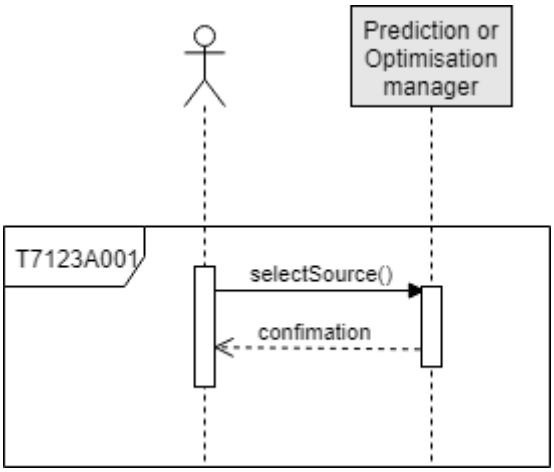


Figure 157: Selecting the source Sequence Diagram

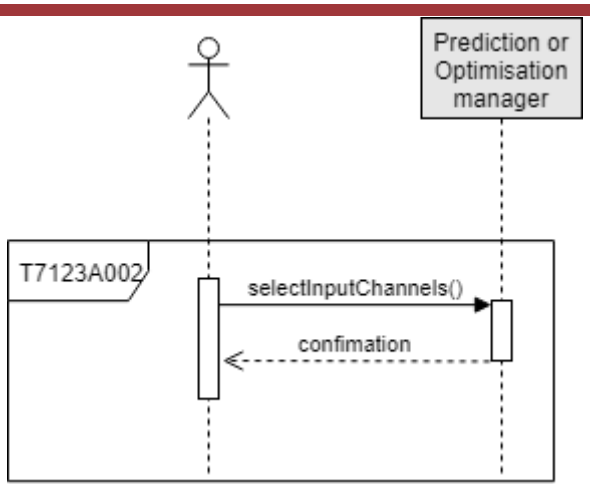


Figure 158: Specifying the input channels Sequence Diagram

5.7.3.2 Perform optimisation

The following two diagrams show the sequence for the component specialized as "Optimiser".

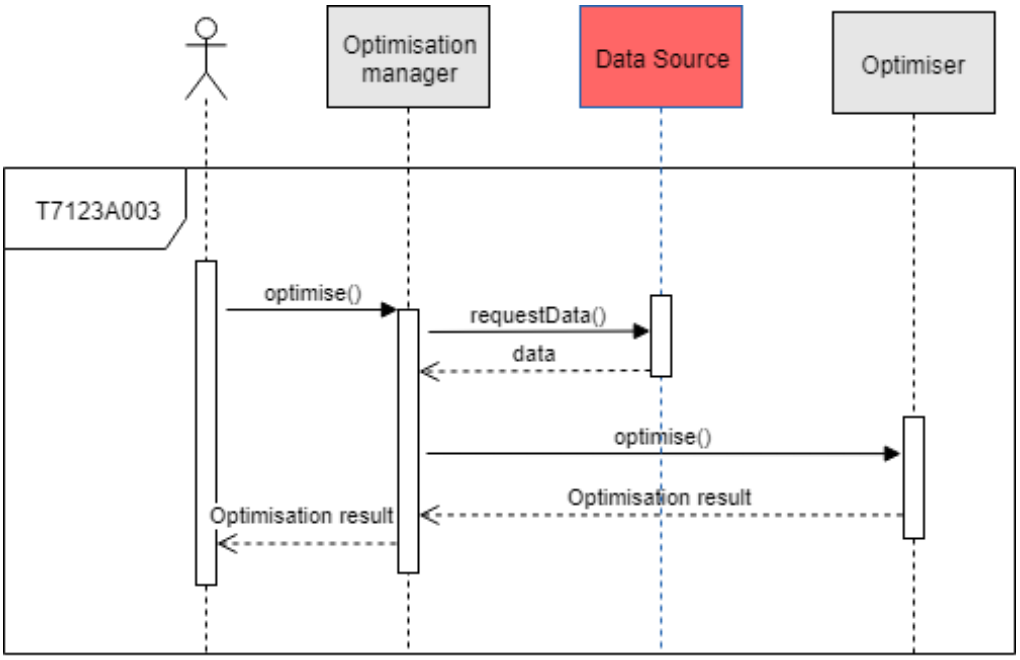


Figure 159: Optimization

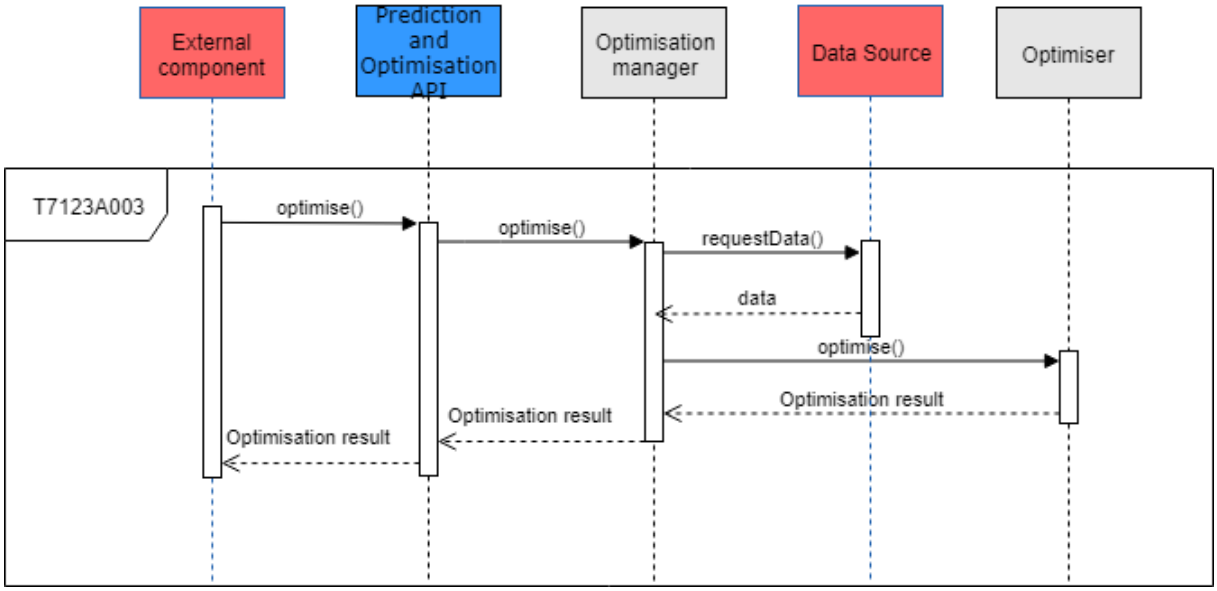


Figure 160: Optimization including the Optimizer

5.7.3.3 Perform prediction

The following two diagrams show the sequence for the component specialized as “Predictor”.

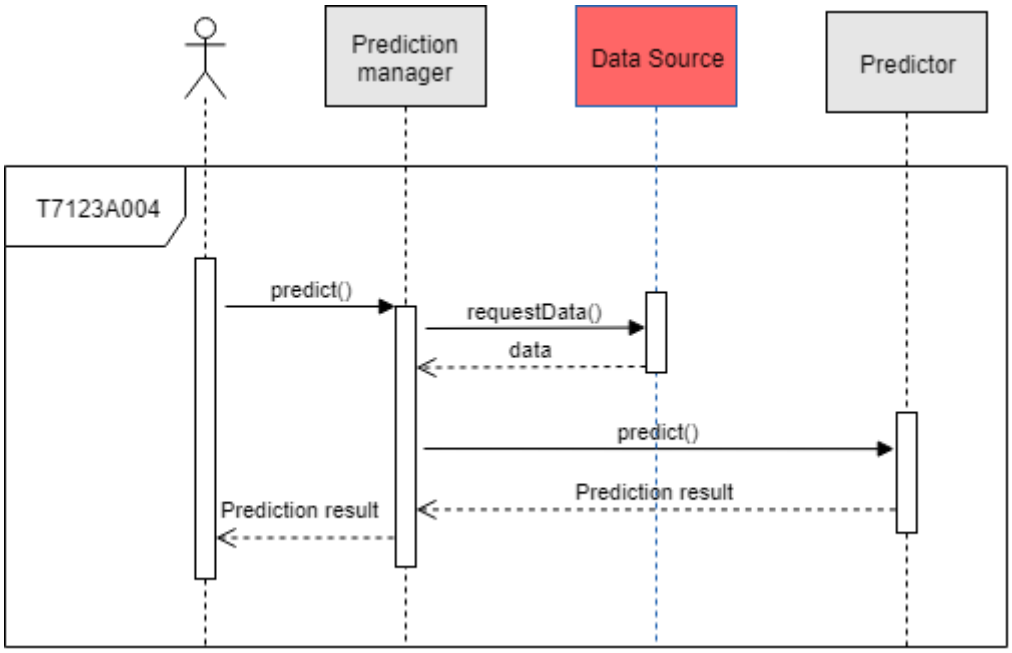


Figure 161: Prediction

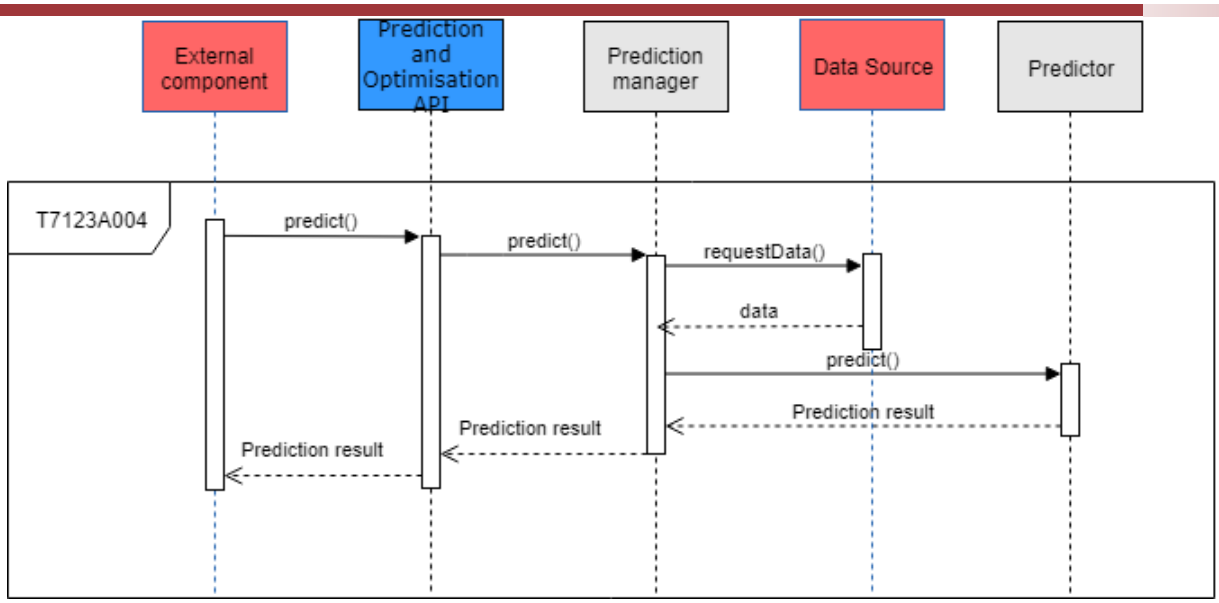


Figure 162: Prediction including Predictor component

5.7.3.4 Check for updates

The following diagram explains this function and the necessary interactions with other components.

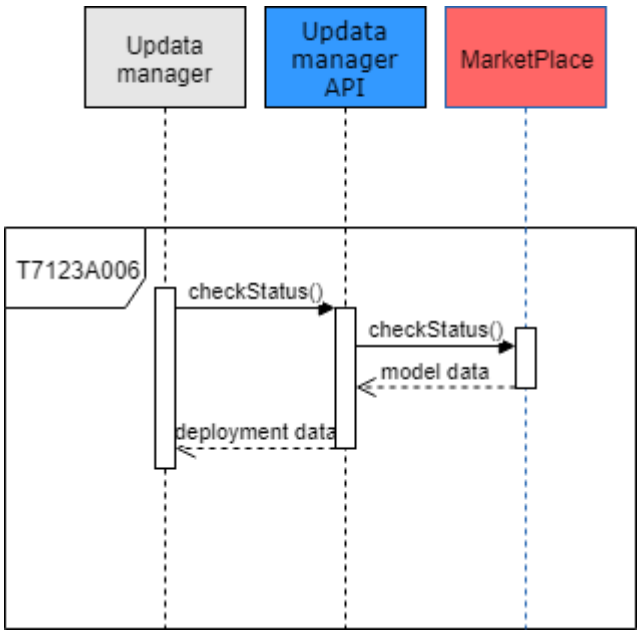


Figure 163: Check for Updates sequence diagram

5.8 Process Assurance Runtime (T7.4)

5.8.1 Overall functional characterization & Context

T7.4 support process setup by suggesting optimised process parameters assuring process quality and making the manufacturing process self-adaptive proposing decisions on the

best actions to optimise overall process quality. This component consists of three main modules: Quality Prediction, Quality Analysis, and Process Self-Adaptation Enabler.

The component is composed of the following modules:

-

5.8.2 Functions / Features

- **Input data management**: data from the analysed processes need to be used to check the quality of the process itself. Therefore, the appropriate data needs to be acquired from different sources and the most appropriate needs to be selected for the three main objectives of this architecture: analyse the quality of the process, make predictions on the quality and obtain values to optimise the processes.
- **Prediction/optimisation requirements specification**: the user can select requirements for the prediction process, eg kind of models to be used, the parameters of the models, the origin of the data, the data to be used and the amount of data used for training, among others.
- **Achievement of results**: To show the predictions / classifications / optimisation results required by the user based on the selected data and the specified requirements
- **Visualisation of the results**: Show the results (analysis, prediction, and optimisation) based on the selection of the signals/indicators desired by the user
- **Models/results management**: Loading and saving the models before or after the training process to be reused in another session

These functions can be further decomposed into the following subtasks that have to be realised:

Subtask	Subtask description
T74A001 **Select data for quality prediction**	**Priority:** Must
	Who: Process Engineer **What:** The user selects which data he wants to use for quality prediction **Where**: Anywhere **When**: During manufacturing **Why:** To specify which data is the desired one from all the available data
Acceptance Criteria	Data sources that have been previously configured in the platform are shown and selected
Requirements filled	N/A
T74A002 **Select data for quality analysis**	**Priority:** Must
	Who: Process Engineer **What:** The user selects which data wants to use for quality analysis **Where**: Anywhere **When**: During manufacturing **Why:** To specify which data is the desired one from all the available data
Acceptance Criteria	Data sources that have been previously configured in the platform are shown and selected
Requirements filled	N/A
T74A003 **Select data for quality process optimisation**	**Priority:** Must
	Who: Process Engineer **What:** The user selects which data wants to use for process quality optimisation **Where**: Anywhere **When**: During manufacturing

	Why: To specify which data is the desired one from all the available data
Acceptance Criteria	Data sources that have been previously configured in the platform are shown and selected
Requirements filled	N/A
T74A004 Visualise data for quality prediction	**Priority:** Must
	Who: Process Engineer **What:** The user selects the kind of graphs to be shown in the interface **Where**: Anywhere **When**: During manufacturing **Why:** To get information on the data to be used in the prediction
Acceptance Criteria	Time series, data in other domains (eg frequency) and/or parameters of the process are shown to the user based on the selection of the data sources in T74A001
Requirements filled	N/A
T74A005 Visualise data for quality analysis	**Priority:** Must
	Who: Process Engineer **What:** The user selects the kind of graphs to be shown in the interface **Where**: Anywhere **When**: During manufacturing **Why:** To get information on the data to be used in the classification
Acceptance Criteria	Time series, data in other domains (eg frequency) and/or parameters of the process are shown to the user based on the selection of the data sources in T74A002
Requirements filled	N/A
T74A006 Visualise data for process quality optimisation	**Priority:** Must
	Who: Process Engineer **What:** The user selects the kind of graphs to be shown in the interface **Where**: Anywhere **When**: During manufacturing **Why:** To get information on the data to be used in the optimisation
Acceptance Criteria	Time series, data in other domains (eg frequency) and/or parameters of the process are shown to the user based on the selection of the data sources in T74A003
Requirements filled	N/A
T74A007 Set Algorithm and its properties	**Priority:** Must
	Who: Process Engineer **What:** The user selects the optimisation algorithm from an available set of options **Where**: Anywhere **When**: During manufacturing **Why:** To specify the algorithm and model to use
Acceptance Criteria	With the kind of model, time performance, and accuracy provided, the user can have the necessary data to launch an optimisation/analysis/optimisation
Requirements filled	N/A
T74A008 Predict	**Priority:** Must
	Who: Process Engineer **What:** The user requests a prediction on the quality of the process **Where**: Anywhere **When**: During manufacturing **Why:** To get information on the future quality of the process
Acceptance Criteria	The user obtains a prediction based on the selected data, algorithms, and properties
Requirements filled	N/A
T74A009	**Priority:** Should

Classify	**Who:** Process Engineer **What:** The user requests a classification on the quality of the process **Where**: Anywhere **When**: During manufacturing **Why:** To analyse the current quality of the process
Acceptance Criteria	The user obtains a classification based on the selected data, algorithms, and properties
Requirements filled	N/A
T74A010 **Optimise**	**Priority:** Must
	Who: Process Engineer **What:** The user requests an optimisation of the parameters of the process **Where**: Anywhere **When**: During manufacturing **Why:** To optimise the process itself
Acceptance Criteria	The user obtains an optimisation based on the selected data, algorithms, and properties
Requirements filled	N/A
T74A011 **Save model**	**Priority:** Must
	Who: Process Engineer **What:** The user saves the created model **Where**: Anywhere **When**: at any moment **Why:** To save a model instance with the provided configuration
Acceptance Criteria	If successful, the user is prompted with a confirmation message. If not, the user is prompted with an error message
Requirements filled	N/A
T74A012 **Load model**	**Priority:** Must
	Who: Process Engineer **What:** The user loads the created model **Where**: Anywhere **When**: at any moment **Why:** To configure the data sources for the model
Acceptance Criteria	If successful, the user is prompted with a confirmation message. If not, the user is prompted with an error message
Requirements filled	N/A
T74A013 **Show quality prediction results**	**Priority:** Must
	Who: Process Engineer **What:** The user obtains the results of the prediction results **Where**: Anywhere **When**: at any moment **Why:** To visualise the prediction results
Acceptance Criteria	The user can select the graphs/indicators from a set of pre-defined options
Requirements filled	RQ_0088, RQ_0123, RQ_0124, RQ_0269, RQ_0270, RQ_0274
T74A014 **Show quality analysis results**	**Priority:** Must
	Who: Process Engineer **What:** The user obtains the results of the analysis results **Where**: Anywhere **When**: at any moment **Why:** To visualise the analysis results
Acceptance Criteria	The user can select the graphs/indicators from a set of pre-defined options
Requirements filled	RQ_0107, RQ_0263, RQ_0272
T74A015 **Show process quality optimisation results**	**Priority:** Must
	Who: Process Engineer **What:** The user obtains the results of the optimisation results **Where**: Anywhere **When**: at any moment **Why:** To visualise the optimisation results

Acceptance Criteria	The user can select the graphs/indicators from a set of pre-defined options
Requirements filled	RQ_0050, RQ_0102

Figure 164: Process Assurance Runtime

5.8.3 Workflows

5.8.3.1 Input data management

This workflow obtains data to be used in the Quality Prediction, Quality Analysis and Process Self-Adaption Enabler modules based on the inputs provided by the user. The main steps are:

- Selection of the data to be loaded
- Loading of the data

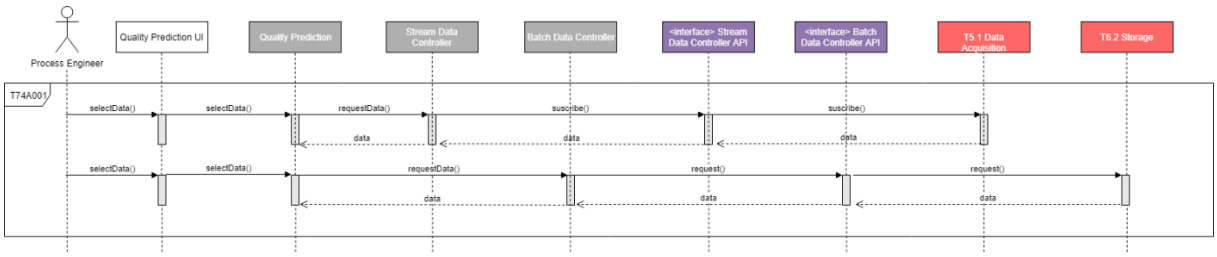


Figure 165. Scheme of subtask T74A001 (input data management)

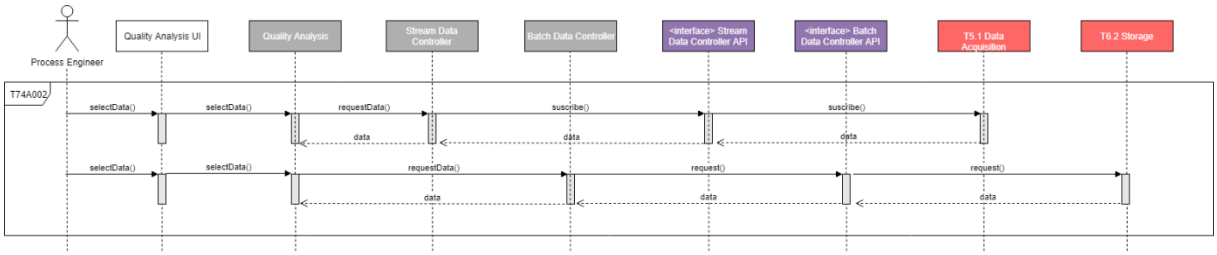


Figure 166. Scheme of subtask T74A002 (input data management)

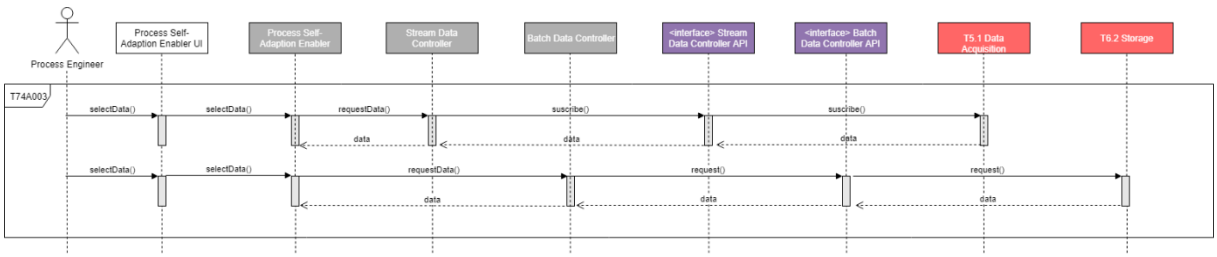


Figure 167. Scheme of subtask T74A003 (input data management)

5.8.3.2 Visualisation

This workflow provides visualisations of the input data and the prediction/classification/optimisation results. The main steps are:

- Selection of the required information (data or results)
- Visualisation of the information (data or results)

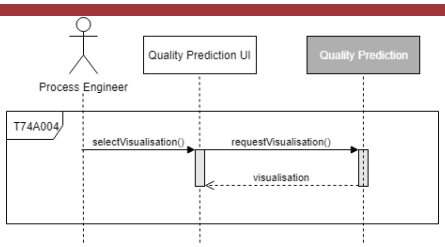


Figure 168: Scheme of subtask T74A004 (visualisation)

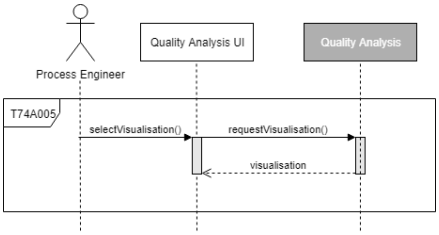


Figure 169: Scheme of subtask T74A005 (visualisation)

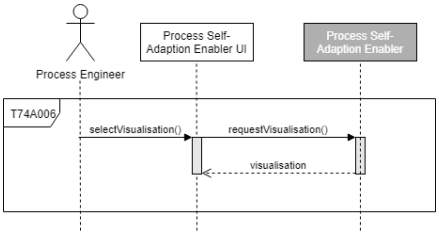


Figure 170: Scheme of subtask T74A006 (visualisation)

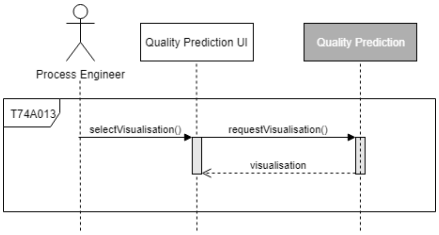


Figure 171: Scheme of subtask T74A007 (visualisation)

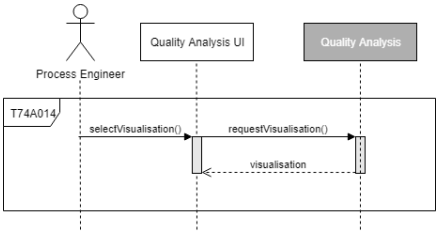


Figure 172: Scheme of subtask T74A008 (visualisation)

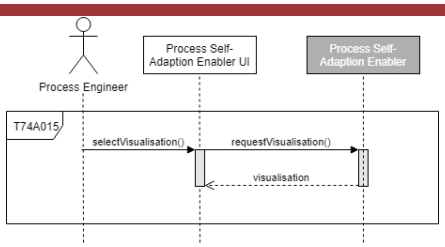


Figure 173: Scheme of subtask T74A009 (visualisation)

5.8.3.3 Prediction/optimisation requirements specification

This workflow sets the details on the prediction/classification/optimisation models to be defined (kind of models, parameters, objectives) based on the inputs provided by the user. The main steps are:

- Definition of the details

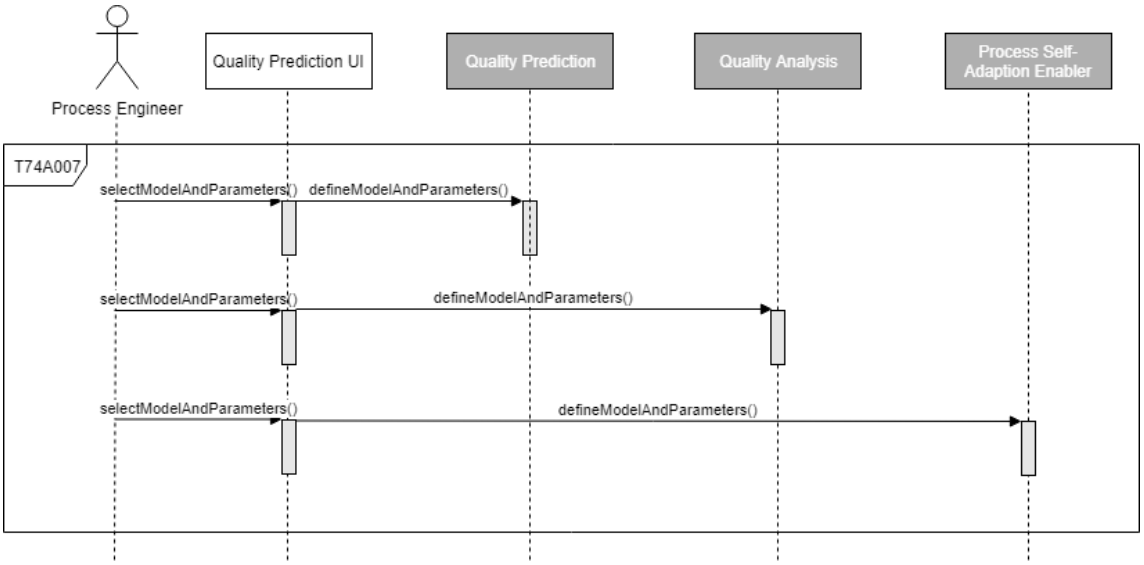


Figure 174: Scheme of subtask T74A010 (prediction/optimisation requirements specification)

5.8.3.4 Achievement of results

This workflow obtains the prediction/classification/optimisation results based on the data and the requirements. The main steps are:

- Creation of a new instance of a prediction/optimisation model
- Calculation of the prediction/classification/optimisation
- Extraction of the results

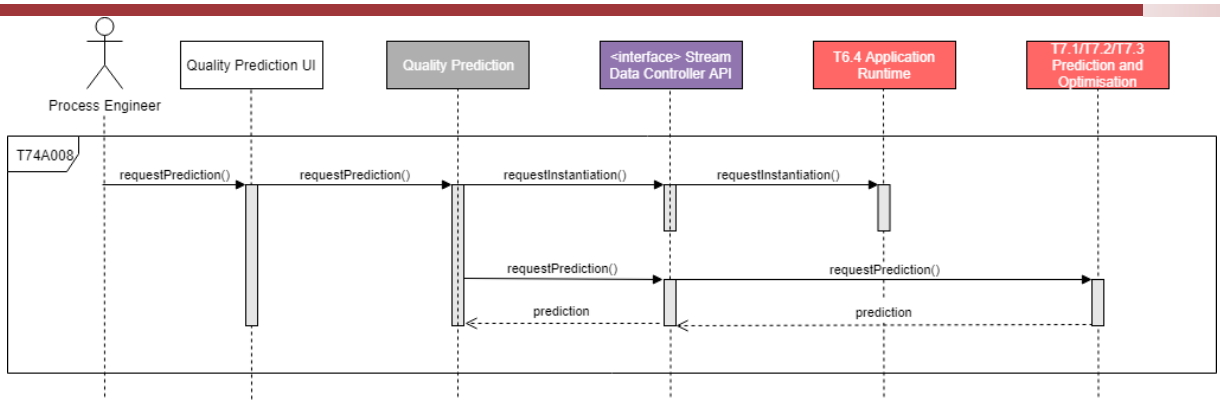


Figure 175: Scheme of subtask T74A011 (achievement of results)

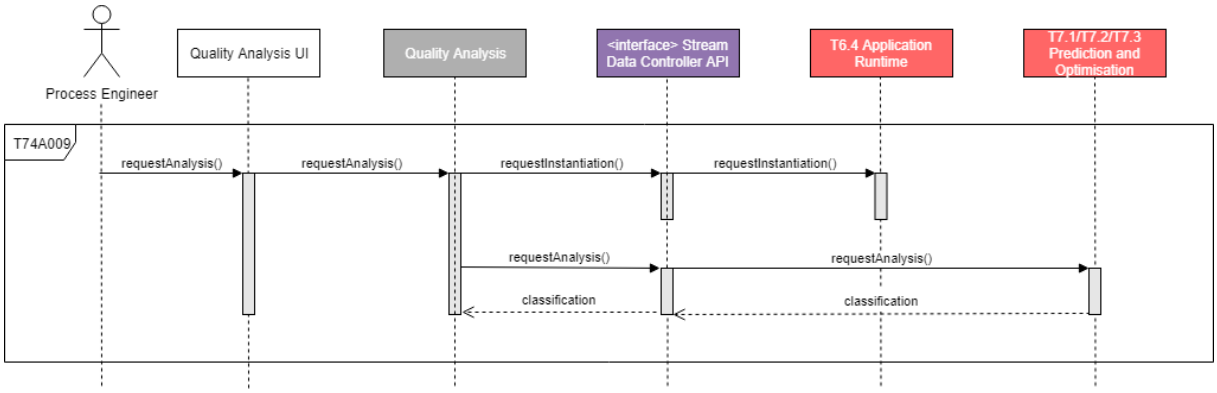


Figure 176: Scheme of subtask T74A012 (achievement of results)

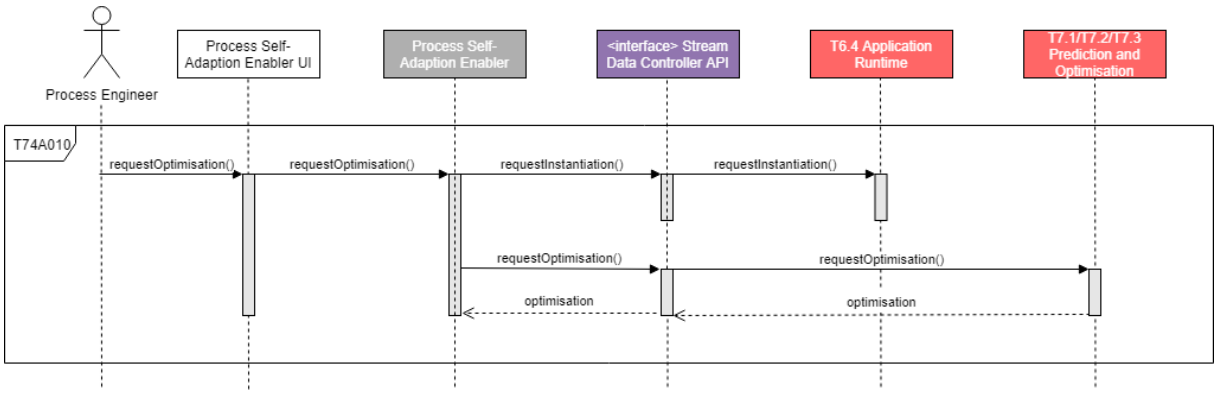


Figure 177: Scheme of subtask T74A013 (achievement of results)

5.8.3.5 Models/results management

This workflow saves and loads the models and the results. The main steps are:

- Selection of the models/results to be saved
- Saving of the models/results
- Loading of the models/results

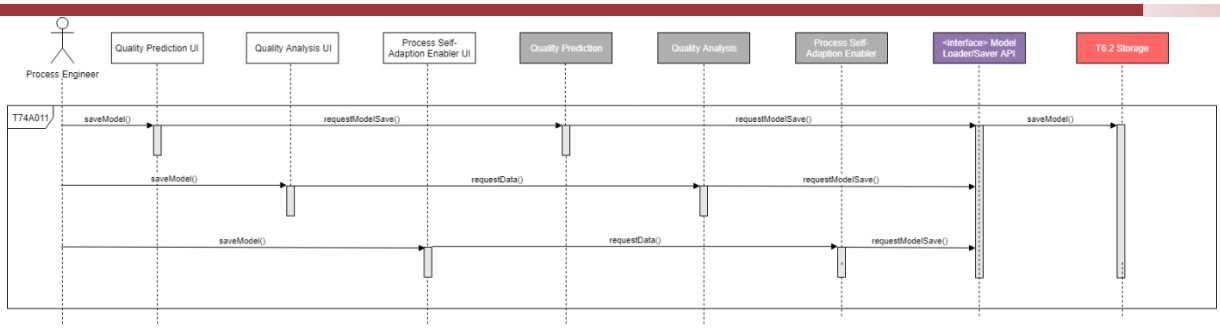


Figure 178: Scheme of subtask T74A014 (model/results management)

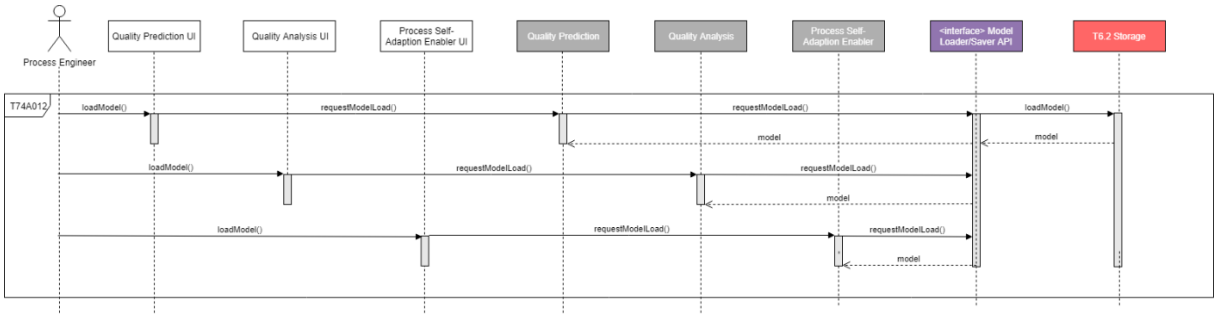


Figure 179: Scheme of subtask T74A015 (model/results management)

5.9 Models Deployment Manager (T8.2, T8.4)

Part of Product Assurance Run-time.

5.9.1 Overall functional characterization & Context

The main goal of this component is to create containers of product quality models or supervision models at run-time. A model is created by three main components: a data processor, a trainer module (Product Quality or Supervision) and a prediction module. In a production environment of the ZDMP platform, each zApps has its specific analytics requirements. This component supports the creation of an analytical model for supporting their specific purposes, selecting the best ML algorithm and data processing configuration to fulfil the task. This component orchestrates the creation of the three modules for each analytical model and manages them a whole runtime instance. Both the data processor, the training module and a prediction module are deployed inside a container, then defining a model container as a single functional unit. So, this component communicates with the Application Services component for supporting the resources assignment, initial configuration, and horizontal scalability for container deployment. This communication for deploying the containers is supported via the Supervision and Product Quality model API.

5.9.2 Functions / Features

- **Data Processor instance creation:** Create a new instance of a data processor to transform the data required for the corresponding trainer
- **Product Quality Model trainer creation:** Create new trainer for generating a Product Quality model to generate predictions regarding quality indicators. The

trained model runs in an isolated container and is initialized using a configuration file, to define properties such as the pushing data rate to a predictive model

- **Supervision Model trainer creation:** this component resembles the previous one, creating a new instance of a supervision model trainer
- **Container deployment:** deploy instances of the previous modules as docker containers using the T6.4/6.5 Application Runtime
- **Data processor – Models interoperability:** connect/route the processed data coming from an instance of the data processor to the trainer and prediction module
- **Historical model instances:** this component logs the different models requested by the zApps and their status (running, stopped, etc.)

These functions can be grouped to the following features, which must be implemented:

Subtask	Subtask description
T82A001 Processing Data Instance Generation	**Priority:** Must
	Who: Models Deployment Manager **Where:** In a container service provider **When:** At design time when a model creation is started from a zApp **What:** Creates an instance for data processing **Why:** To generate the necessary data to feed the Product Quality and Supervision Trainers
Acceptance Criteria	The data processor instance is available
Requirements filled	N/A
T82A002 Create Product Quality Model	**Priority:** Must
	Who: zApp **Where:** In a container service provider **When:** At design time when a Product Quality model creation is requested by a zApp **What:** Create an instance of product quality model trainer in run-time **Why:** To start the process of training the model as data arrives
Acceptance Criteria	The Product Quality trainer is ready to receive data for training
Requirements filled	N/A
T82A003 Create Supervision Model	**Priority:** Must
	Who: zApp **Where:** In a container service provider **When:** At design time when a Product Quality model creation is requested by a zApp **What:** Create an instance of supervision model trainer in run-time **Why:** To start the process of training the model as data arrives
Acceptance Criteria	The Supervision trainer is ready to receive data for training
Requirements filled	N/A
T82A004 Model Container Deployment	**Priority:** Must
	Who: Models Deployment Manager **Where:** In a container service provider **When:** At runtime when a model creation is requested **What:** To deploy instances of the data processor and a Product Quality Model or a Supervision model as docker containers using the Application Runtime **Why:** Models must be executed at runtime
Acceptance Criteria	The Docker container is started and running
Requirements filled	N/A
T82A005 Establish Data Processing Link	**Priority:** Must
	Who: Models Deployment Manager **Where:** Inside a model container **When:** At runtime when a model creation is requested

	What: To connect/route the data processor to the specific trainer and prediction modules **Why:** To send the required processed data to the trainer and prediction modules to be used
Acceptance Criteria	Data is received by both trainer and prediction modules
Requirements filled	N/A
T82A006 **Store Model Instances**	**Priority:** Must
	Who: Models Deployment Manager **Where:** In an external repository (AI-Designer) **When:** At runtime when a new model is generated by the training modules **What:** this component logs the different models created by the zApps and their status (running, stopped, etc) **Why:** Store model instances previously created and made them available to other zApps
Acceptance Criteria	The model instance is available in the AI-Analytics Designer
Requirements filled	N/A

Figure 180: Models Deployment Manager Functions

5.9.3 Workflows

5.9.3.1 Create Product Quality or Supervision Model

This workflow represents both tasks as the sequence of actions are similar.

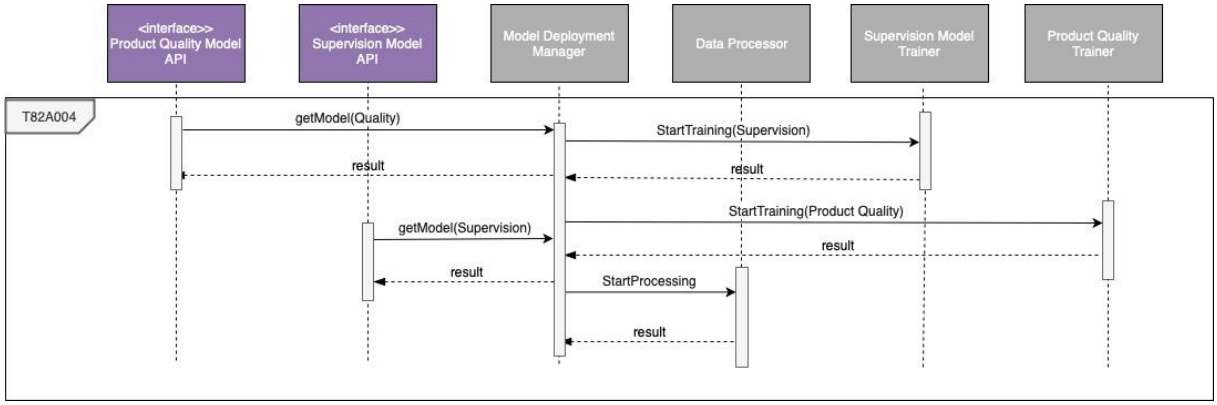


Figure 181: Create Product Quality / Supervision Model

5.9.3.2 Store Model Instances

Suggested sequence:

- Supervision Trainer or Quality Trainer send an updated model to model deployment manager
- Deployment manager redirects the model to the specific API (Product Quality or Supervision)
- Both API connect with the AI-Analytics designer that will function as repository of the models

5.10 Data Processor (T8.2, T8.4)

Part of Product Assurance Run-time.

5.10.1 Overall functional characterization & Context

This component receives the relevant data related with the manufacturing process of a product and processes the data according to a set of predefined and configurable rules. Usually data gathered from an industrial process cannot be used “as-is” for supporting analytical tasks. For instance, data frequency from sensors could range from seconds to days. Using this module, data could be temporally aligned to the same frequency selecting the most suitable aggregation (sum, average, etc) for each sensor. Additionally, data should be sent to the models in specific time window sizes, eg only the values received in the last minute, hour, etc. Therefore, this component adapts the data to be used in the next step of the data analysis pipeline.

To support scalability, this component expects a publish/subscribe broker to get the data in real-time. Following this architectural approach, the Data Processor is subscribed to the topics with the required data, applies a series of rule transformations (temporal alignment, selection, etc.) and then sends a sliding window of the data to the specific models.

5.10.2 Functions / Features

- **Data subscription:** get the information from the publish/subscribe broker (from the T6.4/6.5 Message Bus) on a specific scheduling (every minute, second etc). The component expects that the data follows a predefined schema.
- **Authentication for data subscription: R**equest if the data processor is authorized to get the product data currently available. This a security mechanism to ensure privacy.
- **Processing rules:** the component provides a set of predefined rules for processing the data. Rules are foreseen for supporting temporal alignment if product data is received following different frequencies, ie, applying downsampling or upsampling functions.
- **Processing rules configuration:** the previously defined rules are configurable, for instance defining the aggregation frequency (every minute, hour, etc.), the aggregation operation (sum, min, max, average) and the size of the window (send the last 10 aggregations of data). Configuration is received on model instance creation.
- **Data Windows publishing:** the component periodically sends the processed data to the rest of the related components.
- **Historical data storage:** after the processing step data is stored in a database for further analysis.

Subtask	Subtask description
T82B001 Create Data Subscription	**Priority:** Must
	Who: zApp **Where:** In a model container **When:** Design time when the data processor is created for supporting a model training **What:** Receive product information from the Message Bus on a continuous basis **Why:** To send the data to the prediction and trainer modules

Acceptance Criteria	It should be logged the amount of data received (number of messages)
Requirements filled	N/A
T82B002 Authentication for data subscription	**Priority:** Should
	Who: Data Processor **Where:** Security Run-time **When:** At design time when the data processor is created to support a model training **What:** Product data could be confidential, requiring authenticating the data user **Why:** To ensure security and privacy of the product data
Acceptance Criteria	Data cannot be accessible if the authorization is not granted Only product data authorized for the specific login should be accessible
Requirements filled	N/A
T82B003 Processing Rules Definition	**Priority:** Must
	Who: zApp **Where:** zApp User Interface **When:** at design time from the configuration established by a zApp **What:** Define a processing rule for a variable of the product data **Why:** Data could be received following a different format or temporal alignment than the expected by the models
Acceptance Criteria	Processing rules must be compliant with a previously defined syntax
Requirements filled	RQ_0028
T82B004 Data Processing	**Priority:** Must
	Who: Data Processor **Where:** In a model container **When:** At runtime when a new set of data is received **What:** Processing rules are applied to the received data **Why:** To transform data and aggregate them according to a specific temporal window
Acceptance Criteria	Rules must be applied if the variable is received in the subset of data
Requirements filled	RQ_0028
T82B005 Processed Data Storage	**Priority:** Must
	Who: Data Processor **Where:** In predictions repository **When:** At runtime after processing the product data **What:** Processed data according to the defined rules is stored **Why:** Processed data must be available for generating an historical set or for performing further analysis
Acceptance Criteria	Data must be stored and available for retrieval as soon as it is processed
Requirements filled	N/A
T82B006 Processed Data Publishing	**Priority:** Must
	Who: Data Processor **Where:** In a model container **When:** At runtime after processing the product data **What:** To send the processed data to the subscribed modules **Why:** Different submodules, mainly the trainer and predictor, used the same data
Acceptance Criteria	Submodules must receive the data as it is published Data publishing ratio must be logged (timestamp and number of items)
Requirements filled	RQ_0028

Figure 182: Data Processor Functions

5.10.3 Workflows

5.10.3.1 Data Processing

The following sequence diagram describes the data processing workflow.

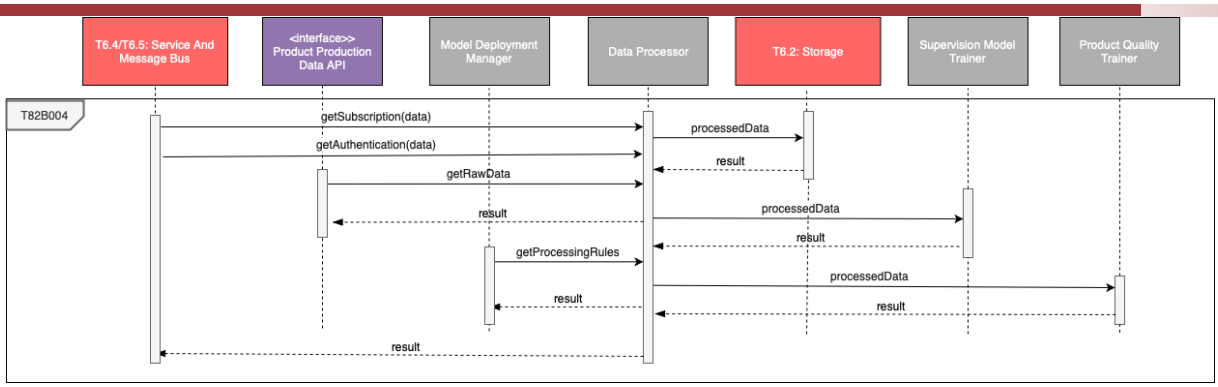


Figure 183: Data Processing Functions Sequence Diagram

5.11 Product Quality Model Trainer (T8.2, T8.4)

Part of Product Assurance Run-time.

5.11.1 Overall functional characterization & Context

This module implements the training for creating predictive models that support T8.2 goals. The available supervised models are meant to correlate production data with related product quality indicators. It is the Data Processor which prepares complete and consistent training datasets to be processed into this module, including values/labels for the product quality indicators. Once the training is terminated with the required accuracy, the trained model can be used inside the Product Quality Predictor to start performing predictions about quality indicators. Loaded models may be continuously trained inside this module to take advantage of larger datasets including more recent data, so to improve prediction accuracy.

5.11.2 Functions / Features

- **Load training parameters:** the module receives an initial configuration file describing different properties to be used to train the model, like data format, accuracy thresholds, hyper-parameters configuration.
- **Initialize model:** this module should be able to load a previous ML model or train a new one according to the previous parameters.
- **Load training data:** the module receives training datasets previously processed by the Data Processor, to start training the model
- **Continuous training:** the module generates a new version of the model every time a new batch of data is received: training model KPIs are exposed during training to allow process monitoring by the data analyst. It also notifies when the new trained model is ready to be deployed

These functions can be further decomposed into the following subtasks that have to be realised:

Subtask	Subtask description
T82C001 Load Model	**Priority:** Must
	Who: Product Quality Trainer **Where:** Anywhere **When:** At the creation of the module instance at configuration time of zApp **What:** Load from Model Deployment Manager the model selected by the end user **Why:** To perform model training once processed data are available
Acceptance Criteria	The correct model is loaded
Requirements filled	N/A
T82C002 Load Training Parameters	**Priority:** Must
	Who: Product Quality Trainer **Where:** Anywhere **When:** At the creation of the module instance **What:** Configure the training according to the specific configuration passed by the Model Deployment Manager **Why:** To perform a model training once processed data are available
Acceptance Criteria	The configuration files follow the expected schema
Requirements filled	*NA*

T82C003 **Load Training Dataset**	**Priority:** Must
	Who: Product Quality Trainer **Where:** Anywhere **When:** Data Processor triggers a new training once a new batch of data is ready and pre-processed **What:** Loads the processed training dataset passed through by the Data Processor **Why:** To perform model re-training once new processed data are available
Acceptance Criteria	All data items conform the expected format
Requirements filled	N/A
T82C004 **Train Model with Dataset**	**Priority:** Must
	Who: Product Quality Trainer **Where:** Anywhere **When:** Data processor supplies new labelled dataset for training **What:** Start training with a dataset supplied by Data Processor **Why:** To improve prediction capabilities using an extended or more recent dataset
Acceptance Criteria	As soon as the new trained model is available, it should be notified
Requirements filled	N/A
T82C005 **Training KPI status**	**Priority:** Must
	Who: Product Quality Trainer **Where:** Anywhere **When**: An external module requests information about these KPIs **What:** Training status (running, target reached/unreached) and other meaningful training KPIs are available to other modules **Why:** To let data analyst monitor the performance and accuracy of the training process
Acceptance Criteria	The most recent values of the KPIs are available when requested
Requirements filled	N/A
T82C006 **Send Trained Model**	**Priority:** Must
	Who: Product Quality Trainer **Where:** Anywhere **When**: Current error indicator, ie requested accuracy, is improved/reached respect actual quality predictor model after a model training on new dataset **What:** Send freshly trained model to Quality Predictor **Why:** To let Quality Predictor update the model under execution and save it in the storage
Acceptance Criteria	The Quality Predictor correctly runs the new updated model
Requirements filled	N/A
T82C007 **Stop/Start Training**	**Priority:** Must
	Who: Product Quality Trainer **Where:** Anywhere **When**: Data analyst from a zApp or an arbitrating external component sends stop/start command **What:** Stop/Start training the model **Why:** To let Data Analyst or an arbitrating external component stop, reconfigure and restart training, for example when a new rule to start/stop training task is to be applied with respect to ones defined on Product quality trainer module
Acceptance Criteria	The training process is stopped/started as expected
Requirements filled	N/A

Figure 184: Quality Model Trainer Functions

5.11.3 Workflows

The remaining subtasks (T82C003 and T82C007) are representable are straightforward request response exchanges between the Product Quality Trainer module and either the

Data Processor (T82C003), Module Deployment Manager (T82C005/7) and the Quality Predictor (T82C006), the more complex Load Model and Load Training Parameter workflows are shown below.

5.11.3.1 Load Model

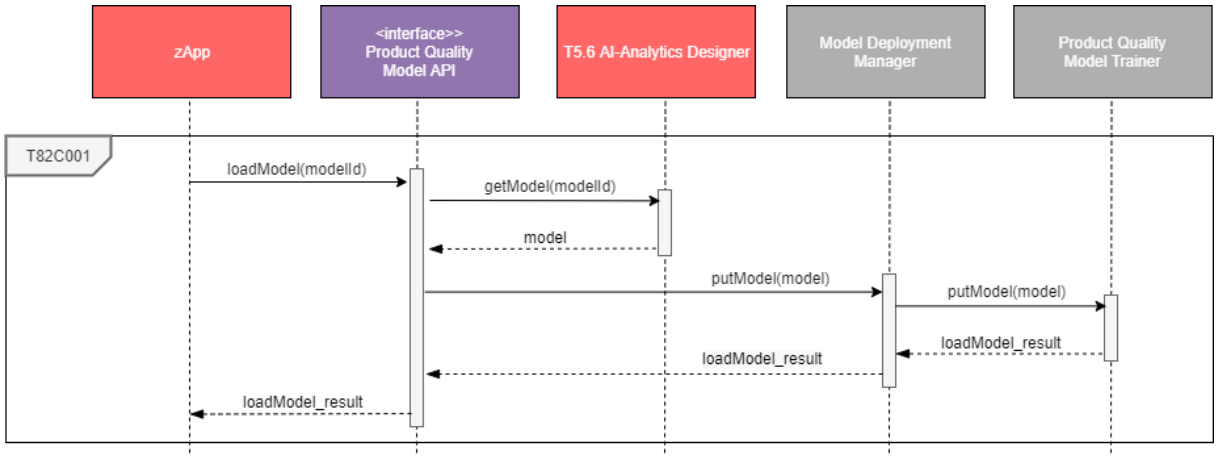


Figure 185: Load Model Sequence Diagram

5.11.3.2 Load Training Parameters

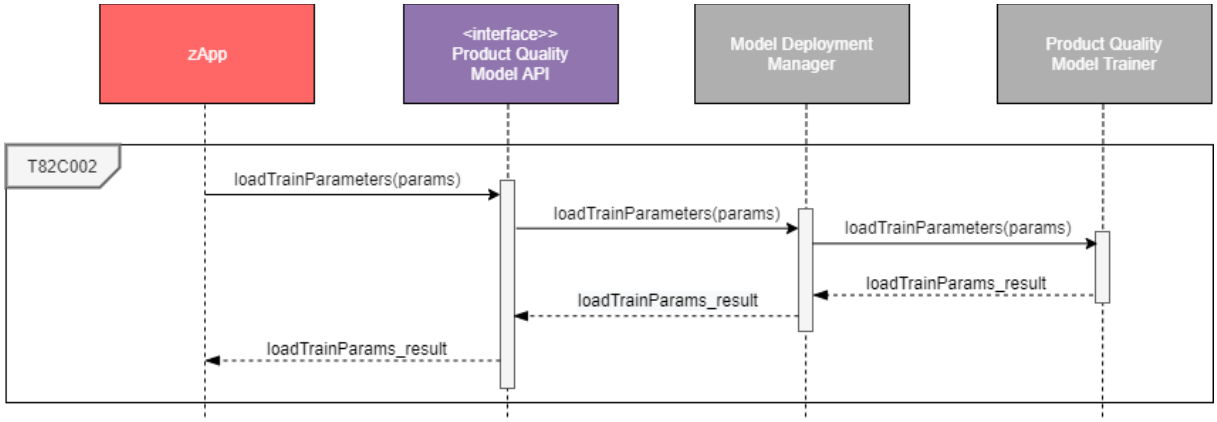


Figure 186: Load Training Parameters

5.12 Predictions Repository (T8.2, T8.4)

Part of Product Assurance Run-time.

5.12.1 Overall functional characterization & Context

This module stores locally the predictions generated by the Quality Predictor for historical purposes and for checking the accuracy achieved by the model.

5.12.2 Functions / Features

- **Predictions storage:** the component supports the storage of the prediction generated by the Quality Model Predictor
- **Query predictions: R**etrieve the predictions generated for a set of variables in a specific period

Subtask	Subtask description
T82D001 Predictions Storage	**Priority:** Must
	Who: Product Quality Predictor **Where:** Local Storage **When:** At runtime as new prediction or set of predictions are generated **What**: This repository stores the predictions generated by the Quality Predictor **Why**: To retrieve historical information in a specific time range and compare the accuracy among
Acceptance Criteria	The predictions repository should store a set of predictions with their specific timestamp and they must be available using the Predictions Query feature
Requirements filled	N/A
T82D002 Predictions query	**Priority:** Must
	Who: zApps or Quality Model Predictor **Where:** From external component (zApp or zAsset) and in the model container **When:** at runtime when a zApp or the predictor request the information to fulfil its functionality **What**: The Predictions Repository retrieves the predictions requested in a specific period **Why**: To get information regarding the predictions generated
Acceptance Criteria	Given a period the repository should return all the predictions generated in such period for a specific quality variable.
Requirements filled	N/A

Figure 187: Predictions Repository Functions

5.12.3 Workflows

Both subtasks are simple request response exchanges between the Product Quality Predictor module and this module, therefore no diagrams are created for simplicity.

5.13 Quality Predictor (T8.2, T8.4)

Part of Product Assurance Run-time.

5.13.1 Overall functional characterization & Context

This component executes a ML model to generate a predictive trend of the quality of the received manufacturing data. Quality is calculated from one or more objective variables from the received data. This component stores the generated prediction, using the prediction repository, to compare the actual values of the product data with the predicted values retrieved form the model. An additional feature of this module is the optimisation of a specific objective variable according to the manufacturing process parameters.

5.13.2 Functions / Features

- **Model execution:** as new processed data is received by this module, predictions are generated and sent to rest of components using an API. For each objective or target variable previously selected in the data processing a, an underlying predictive model provides the values expected for the next period (the next five minutes, the next hours, etc)
- **Optimization: On**ce a model is trained users can search for process conditions that maximize (or minimizes, depending on the type of optimization) one or more quality variables
- **Model updating:** New versions of the resulting model are continuously deployed to provide the requested prediction
- **Predictions storage:** Send the predictions to the prediction storage component for historical analyses

Subtask	Subtask description
T82E001 Execute trained predictor	**Priority:** Must
	Who: Product Assurance Runtime **Where:** In a model container **When:** At runtime after the data processor was started and the training module has generated a model **What:** A previous trained model using the Product Quality Model trainer is executed in runtime with a subset of the product data **Why:** To get predictions regarding product quality as processed data is received form the data processor
Acceptance Criteria	Prediction endpoint of the production quality model accepts requests
Requirements filled	RQ_0045, RQ_0049, RQ_0051, RQ_0103
T82E002 Data batch aggregation	**Priority:** Must
	Who: Product Assurance Runtime **Where:** In a model container **When:** At runtime after the expected amount of processed data is received **What:** To send the processed data to the trainer module using a specific period (every minute, every hour, etc) or an amount of data (for instance, the last 100 items received) **Why:** To get predictions with small subsets of the data and, then, reducing the required time to execute a model
Acceptance Criteria	The generation of batches is logged
Requirements filled	RQ_0045, RQ_0049, RQ_0051, RQ_0103

T82E003 **Prediction publishing configuration**	**Priority:** Must
	Who: zApp or component via Product Quality Model API **Where:** In a model container **When:** At runtime as a response of the model is received **What:** Predictions are published with a predefined time frequency (every 5 seconds, for instance) **Why:** To provide predictions to and external component in a suitable frequency to visualize the data
Acceptance Criteria	Check that predictions are published to the Supervision Model API in the expected frequency
Requirements filled	RQ_0045, RQ_0049, RQ_0051, RQ_0103
T82E004 **Predictions broadcast**	**Priority:** Must
	Who: Product Quality Model API **Where:** Anywhere **When:** At runtime when a set of predictions is received **What:** Predictions must be published and made available to rest of the components as soon as they are generated **Why:** zApps and components will use the predictions to gain insights regarding the quality of the manufacturing data
Acceptance Criteria	Any component or zApp subscribed to the Product Quality Model API must receive the generated predictions.
Requirements filled	RQ_0045, RQ_0049, RQ_0051, RQ_0103
T82E005 **Model update**	**Priority:** Must
	Who: Product Assurance Runtime, Product Quality Model Trainer **Where:** In a model container **When:** At runtime when a condition for updating is met **What**: Use a new product quality model because of a retraining **Why**: To guarantee the accuracy of the generated predictions, models should be updated considering the most recent data
Acceptance Criteria	Model should be updated in real time The predictions generation process must not be stopped until the new model is working
Requirements filled	RQ_0045, RQ_0049, RQ_0051, RQ_0103
T82E007 **Predictions query**	**Priority:** Must
	Who: Quality Predictor, Product Quality Model API **Where:** In a model container **When:** At runtime when an external client or the Quality Predictor request the data **What**: The Quality Predictor or external modules will query the predictions repository in an specific time-range for a specific quality variable **Why**: To get information regarding the predictions generated
Acceptance Criteria	Given a period the repository should return all the predictions generated in such period for a specific quality variable A valid range of start data – end date must be provided
Requirements filled	N/A
T82E008 **Optimization Launch**	**Priority:** Should
	Who: zApp, Product Quality Model API **Where:** zApp configuration interface **When:** at design time in a user interface that configures the optimization (variables, ranges etc) **What**: The module returns the set of process variables values that optimize the outcome of a target variable **Why**: To maximize or minimize the value of a selected target variable
Acceptance Criteria	The user should select at least one target variable for optimization The product data must have at least one process variable or a variable different of the target one

Requirements filled	RQ_0045, RQ_0049, RQ_0051, RQ_0103
T82E009 Optimization	**Priority:** Should
	Who: Quality Predictor **Where:** In a model container **When:** At runtime when an optimization is requested by an external app or component **What**: The Quality Predictor executes several times the trained model with different parameters following an algorithm **Why**: To get the optimal set of values for each of the process variables involved
Acceptance Criteria	Simultaneous call to the trained model must be possible If a solution is not found, it should be notified
Requirements filled	RQ_0045, RQ_0049, RQ_0051, RQ_0103

Figure 188: Quality Predictor Functions

5.13.3 Workflows

5.13.3.1 Execute trained ML model

The following diagram explains this function and the necessary interactions with other components.

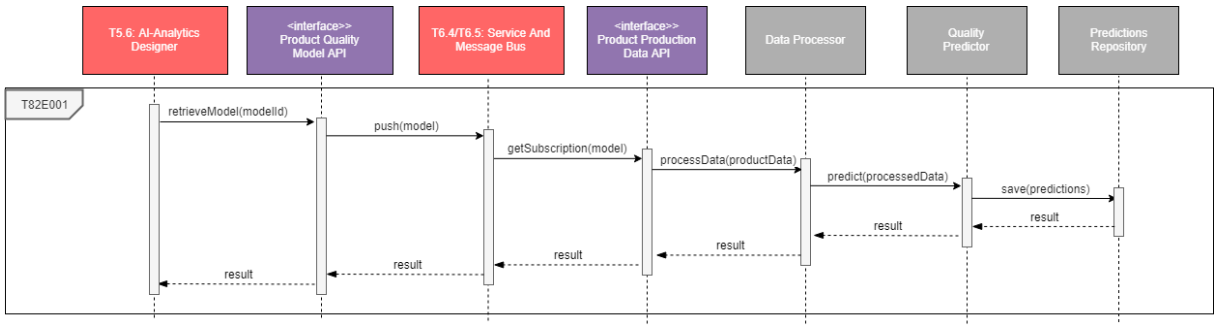


Figure 189: Execute trained machine learning model

5.13.3.2 Predictions broadcast

The following diagram explains this function and the necessary interactions with other components.

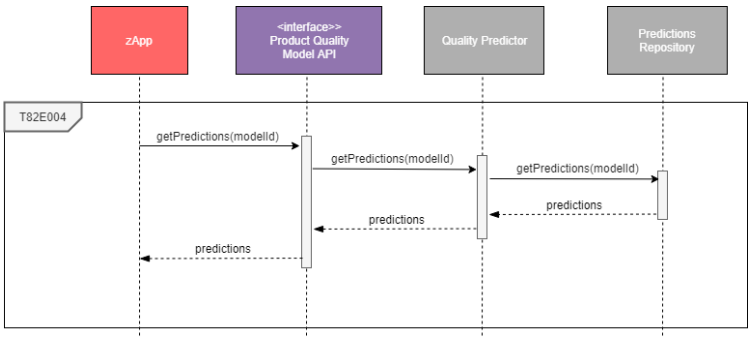


Figure 190: Predictions Broadcast

5.13.3.3 Model update

The following diagram explains this function and the necessary interactions with other components.

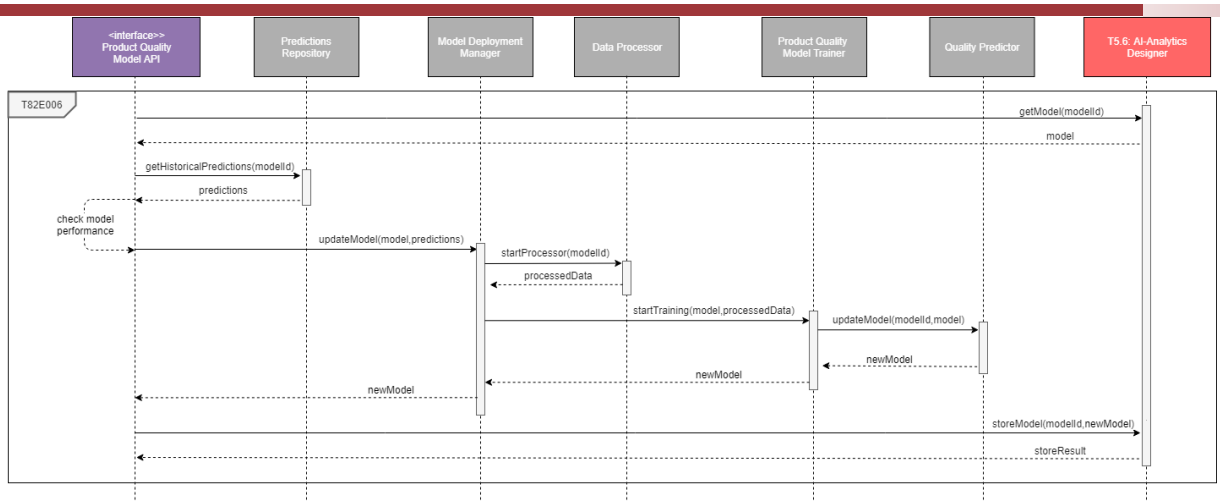


Figure 191: Model Update Sequence Diagram

5.13.3.4 Predictions query

The following diagram explains this function and the necessary interactions with other components.

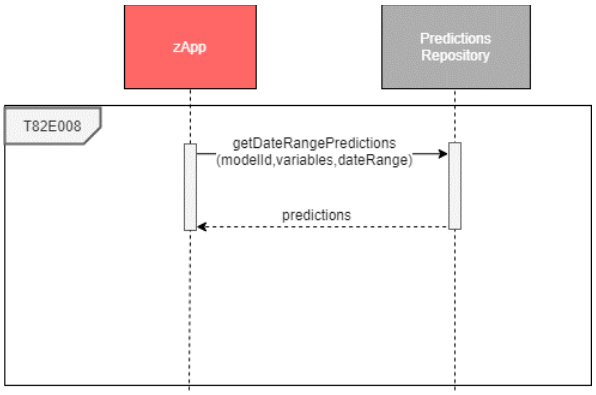


Figure 192: Predictions Query Sequence Diagram

5.14 Supervision Model Trainer (T8.2, T8.4)

Part of Product Assurance Run-time.

5.14.1 Overall functional characterization & Context

This module implements the training and for creating predictive models that support T8.4 goals. These models are based on sound machine learning libraries with the aim of detecting anomalies of production data received at run time. The common approach for building an ML model is to first gather a huge amount of historical data and perform an offline training. However, this approach does not fit for a novel manufacturing process due to the lack of historical data. To address this challenge, this module implements a continuous training of the model as data is received. First, the model is trained with a small subset of the data, and incrementally, the model is improved and updated as new data is received. Anomaly detection is expected to improve over time as more data has been used for training (fitting) the model. This approach also benefits of the fact that, usually, most recent data is more relevant to detect anomalous behaviour.

5.14.2 Functions / Features

- **Load starting model:** the module should be able to load the ML model indicated by the end user
- **Load training parameters:** the module receives an initial configuration file describing different properties to be used to train the model, like data format, accuracy thresholds, hyper-parameters configuration
- **Load training data:** the module receives training datasets previously processed by the Data Processor, to start training the model
- **Continuous training:** the module generates a new version of the model every time a new batch of data is received: training model KPIs are exposed during training to allow process monitoring by the data analyst. It also notifies when the new trained model is ready to be deployed

These functions can be further decomposed into the following subtasks that have to be realised:

Subtask	Subtask description
T82F001 Load Model	**Priority:** Must
	Who: Supervision Model Trainer **Where:** Anywhere **When:** At the creation of the module instance **What:** Load from Model Deployment Manager the model selected by the end user at configuration time of zApp **Why:** To perform model training once processed data are available
Acceptance Criteria	The correct model is loaded
Requirements filled	N/A
T82F002 Load Training Parameters	**Priority:** Must
	Who: Supervision Model Trainer **Where:** Anywhere **When:** At the creation of the module instance **What:** Configure the training according to the specific configuration passed by the Model Deployment Manager

	Why: To perform a model training once processed data are available
Acceptance Criteria	The configuration files follow the expected schema
Requirements filled	N/A
T82F003 **Load Training Dataset**	**Priority:** Must
	Who: Supervision Model Trainer **Where:** Anywhere **When:** Data Processor triggers a new training once a new batch of data is ready and pre-processed **What:** Loads the processed training dataset passed through by the Data Processor **Why:** To perform model re-training once new processed data are available
Acceptance Criteria	All data items conform the expected format
Requirements filled	N/A
T82F004 **Train Model with Dataset**	**Priority:** Must
	Who: Supervision Model Trainer **Where:** Anywhere **When:** Data processor supplies new labelled dataset for training **What:** Start training with a dataset supplied by Data Processor **Why:** To improve anomaly detection capabilities using an extended or more recent dataset
Acceptance Criteria	As soon as the new trained model is available, it should be notified
Requirements filled	N/A
T82F005 **Training KPI status**	**Priority:** Must
	Who: Supervision Model Trainer **Where:** Anywhere **When**: An external module requests information about these KPIs **What:** Training status (running, target reached/unreached) and other meaningful training KPIs are available to other modules **Why:** To let data analyst monitor the performance and accuracy of the training process
Acceptance Criteria	The most recent values of the KPIs are available when requested
Requirements filled	N/A
T82F006 **Send Trained Model**	**Priority:** Must
	Who: Supervision Model Trainer **Where:** Anywhere **When**: Current error indicator, ie requested accuracy, is improved/reached respect actual quality predictor model after a model training on new dataset **What:** Send freshly trained model to Quality Predictor **Why:** To let Quality Predictor update the model under execution and save it in the storage
Acceptance Criteria	The Quality Predictor correctly runs the new updated model
Requirements filled	N/A
T82F007 **Stop/Start Training**	**Priority:** Must
	Who: Supervision Model Trainer **Where:** Anywhere **When**: Data analyst from a zApp or an arbitrating external component sends stop/start command **What:** Stop/Start training the model **Why:** To let Data Analyst or an arbitrating external component stop, reconfigure and restart training, for example when a new rule to start/stop training task is to be applied with respect to ones defined on Supervision Model Trainer module
Acceptance Criteria	The training process is stopped/started as expected
Requirements filled	N/A

Figure 193: Supervision Model Trainer Functions

5.14.3 Workflows

Simple request response exchange workflows for functions T82F003 and T82F007 have been left out for simplicity between the Supervision Model Trainer module and Data Processor (T82F003), Module Deployment Manager (T8FC005/7) and the Anomaly Detector (T82F006).

5.14.3.1 Load Model

The following diagram explains this function and the necessary interactions with other components.

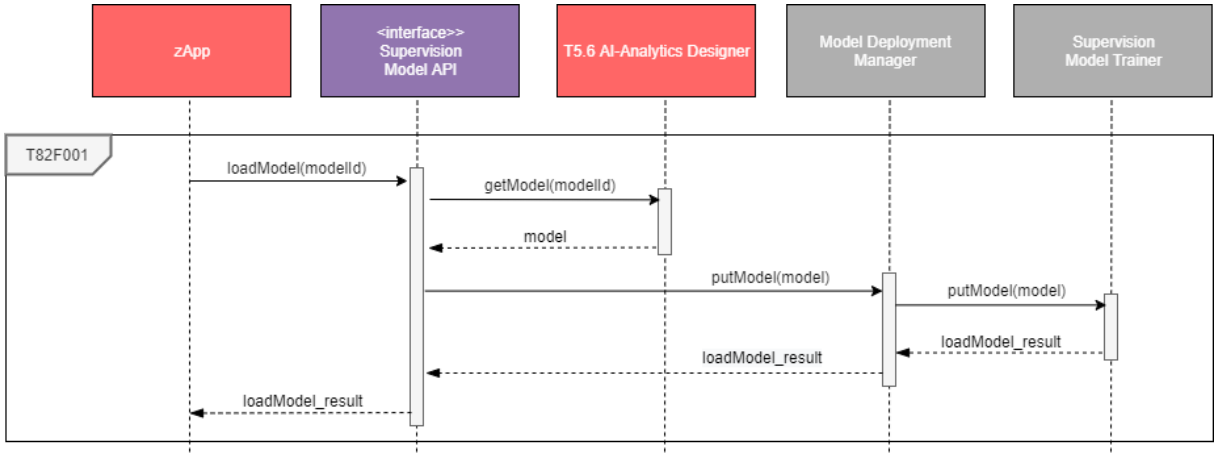


Figure 194: Load Model

5.14.3.2 Load Training Parameters

The following diagram explains this function and the necessary interactions with other components.

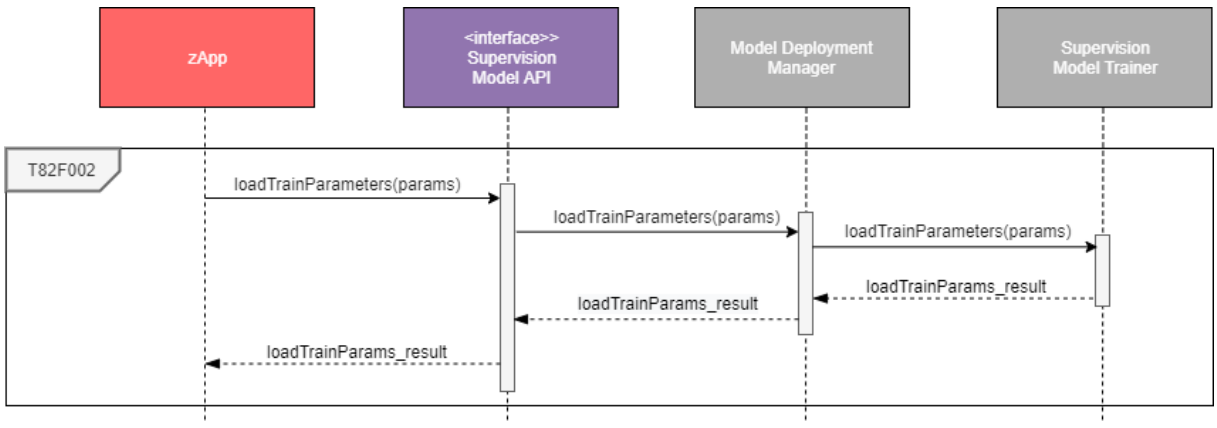


Figure 195: Load Training Parameters Sequence Diagram

5.15 Anomaly Detector (T8.2, T8.4)

Part of Product Assurance Run-time. This component deploys and executes a model to raise alerts when anomalies are detected. Such anomalies are published to be notified by an external app in real time to the user, when one or more statistical indicator computed by the underling ML model are above or below an alert threshold (aka control limit). Thresholds for each variable are also automatically calculated a model created using the Supervision Model trainer. Then, an alert notification will be logged and reported using the Monitoring and Alerting component. This module also records which specific variables are contributing to the incorrect behaviour, to help explaining the anomaly.

5.15.1 Functions / Features

- **Anomaly model specification:** the detector receives a configuration regarding the variables to monitor from the set of product data. These variables are also considered by the Supervision Model trainer
- **Model execution:** as new processed data is received, the model checks for any potential anomaly and the variables involved (also with abnormal values)
- **Anomaly subscription:** External components or zApps subscribe to receive the anomalies as are generated by the continuous execution of the model
- **Alerts notification:** when a potential anomaly is detected by the model, the anomaly is sent and logged. This notification is sent as an alert to other components using the Message bus queues

Subtask	Subtask description
T82G001 **Execute trained anomaly detector**	**Priority:** Must
	Who: Product Assurance Runtime **Where:** In a model container **When:** At runtime as new product data is received **What:** A previous trained anomaly detector model using the Supervision Model trainer is executed in runtime with the received product data **Why:** To get anomalies, which include variable abnormal values, error statistics, possible alerts, and variables responsible of such alerts.
Acceptance Criteria	Anomalies are published to the Supervision Model API as soon as they are generated
Requirements filled	RQ_0013, RQ_0021, RQ_0028, RQ_0034, RQ_0072, RQ_0073
T82G002 **Anomaly notification**	**Priority:** Must
	Who: zApp or component via Supervision Model API **Where:** In a user interface **When:** At runtime as new anomaly is published **What:** Detected anomalies in specific variable must be made available to the rest of the components as soon as they are generated **Why:** zApps and ZDMP components will use the detected anomalies to react accordingly and present notifications to end users
Acceptance Criteria	Every subscribed component must receive the anomaly notification in the next minute If the components were not subscribed, the last anomaly notification is retrieved
Requirements filled	RQ_0013, RQ_0021, RQ_0028, RQ_0034, RQ_0072, RQ_0073
T82G003	**Priority:** Should

Anomaly detector update	**Who**: Product Assurance Runtime, Supervision Model Trainer **Where:** In a model container **When:** At runtime if the anomaly detector model is updated **What**: An anomaly detector model is could be retrained by the Supervision Model Trainer to improve accuracy **Why**: To guarantee the accuracy of the detected anomalies, the new model will replace the previous one
Acceptance Criteria	Model should be updated in real time without stopping the anomaly generation task until the new model is working
Requirements filled	RQ_0013, RQ_0021, RQ_0028, RQ_0034, RQ_0072, RQ_0073
T82G004 Anomaly thresholds reporting	**Priority:** Must
	Who: Anomaly Detector **Where:** In a model container **When:** At runtime when the anomaly detector is trained **What**: For each variable, an upper – bottom normal condition threshold is automatically calculated by the trainer **Why**: These calculated thresholds are updated in real-time and are useful to report what is considered and anomaly
Acceptance Criteria	All variables considered should have a threshold
Requirements filled	RQ_0013, RQ_0021, RQ_0028, RQ_0034, RQ_0072, RQ_0073
T82G005 Anomaly contributing variables	**Priority:** Must
	Who: Anomaly Detector, Supervision Model API **Where:** In a user interface **When:** At runtime when a new anomaly is reported **What**: When an anomaly is detected it should be explained which variables are generating such anomaly **Why**: To report which specific variables must be considered to solve the detected anomaly
Acceptance Criteria	For a given anomaly, it must be calculated the contribution percentage of every variable involved in the anomaly detector.
Requirements filled	RQ_0013, RQ_0021, RQ_0028, RQ_0034, RQ_0072, RQ_0073
T82G006 Recent anomalies	**Priority:** Should
	Who: Anomaly Detector, Supervision Model API **Where:** In a model container **When:** At runtime when the anomalies are requested by an external component **What**: A historical of set of the most recent anomalies **Why**: To receive information of the previous generated anomalies (for instance, if a component was not previously subscribed)
Acceptance Criteria	The last ten generated anomalies must be available
Requirements filled	N/A

Figure 196: Anomaly Detector Functions

5.15.2 Workflows

5.15.2.1 Execute trained ML model

The following diagram explains this function and the necessary interactions with other components.

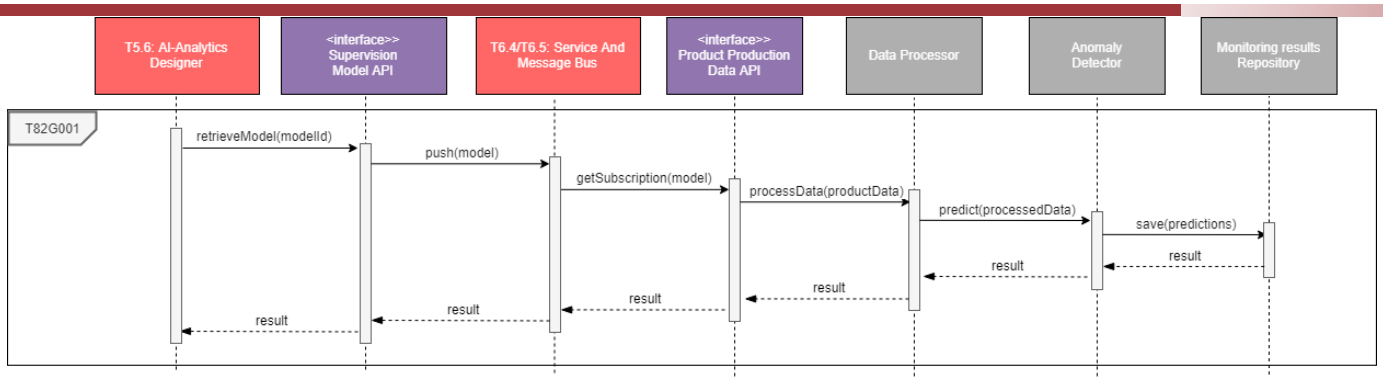


Figure 197: Execute Trained Machine Learning Model Sequence Diagram

5.15.2.2 Alert subscription

The following diagram explains this function and the necessary interactions with other components.

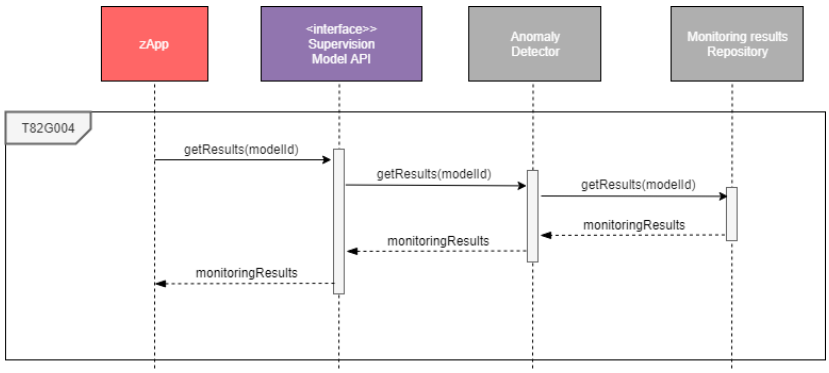


Figure 198: Alert Subscription Sequence Diagram

5.15.2.3 Model updating

The following diagram explains this function and the necessary interactions with other components.

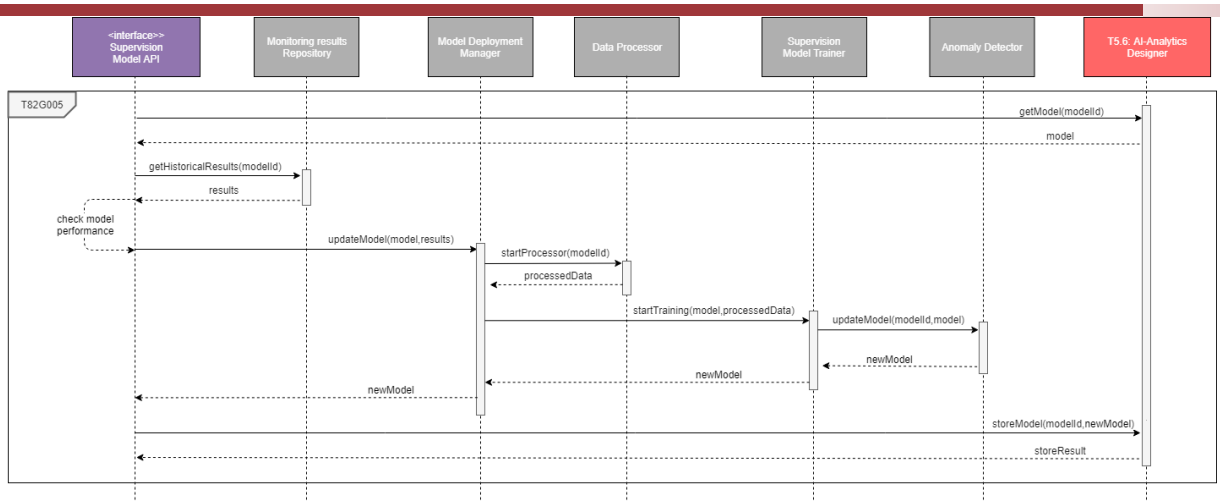


Figure 199: Model Updating Sequence Diagram

5.15.2.4 Monitoring results subscription

The following diagram explains this function and the necessary interactions with other components.

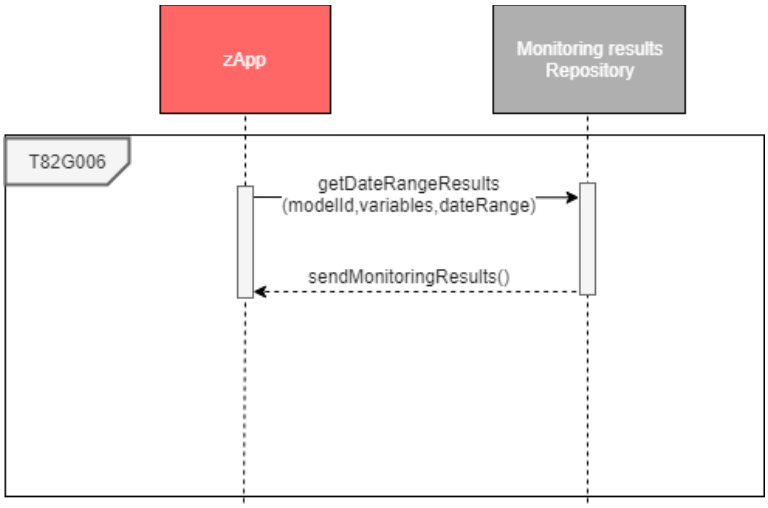


Figure 200: Monitoring Results Sequence Diagram

6 Edge Tier: Run-time

The Edge Tier is a tier for components which need to be located close to the source of data. These components need to be able to integrate with the data quickly due to performance (eg to avoid transferring large amount of image files over the network) or technical reasons (eg image processing that requires a dedicated GPU). They are linked by the T5.5 Distributed Computing Component. It includes aspects such as the T5.1 Data Acquisition and T8.3 Non-Destructive Inspection. Some of these components may be optionally run on the Platform Tier when performance or technical issues do not require them to be run on the Edge Tier.

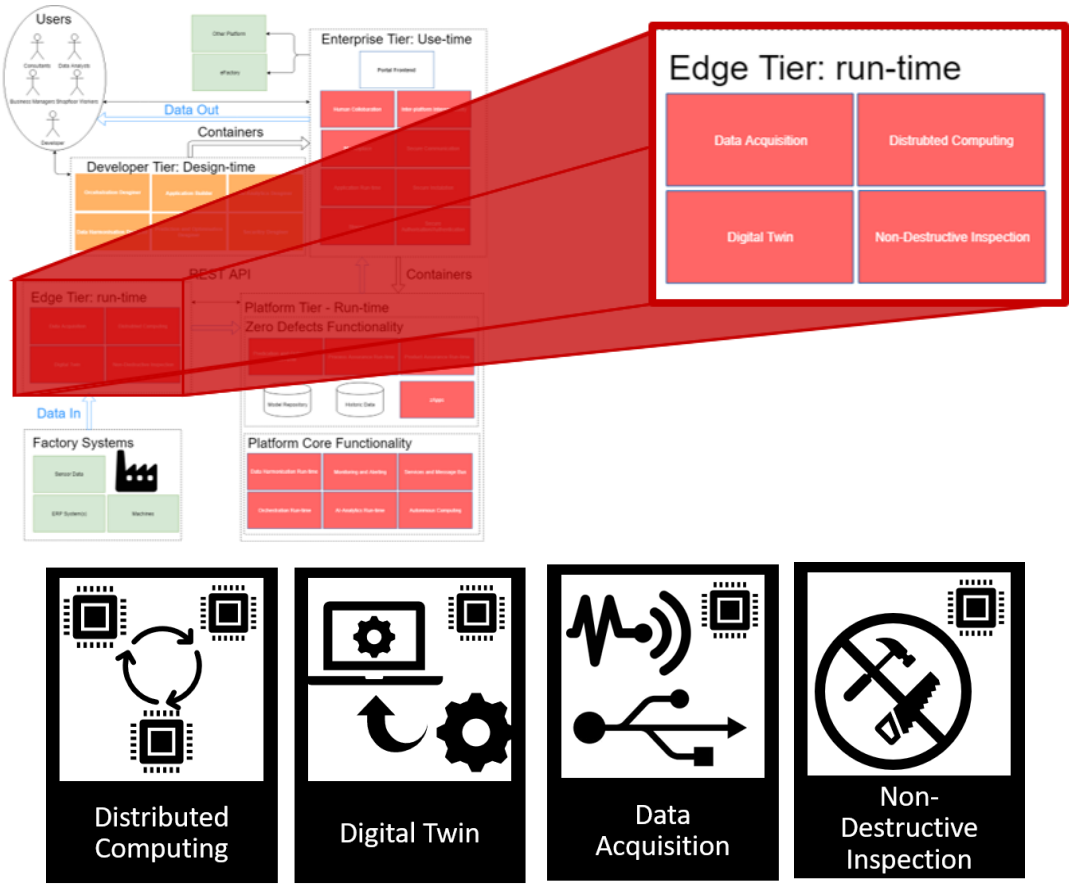


Figure 201: Edge Tier Runtime Components

6.1 Data Acquisition and IIoT (T5.1)

6.1.1 Overall functional characterization & Context

The Data Acquisition component implements a framework for the handling of data from IoT sensors and other sources. It allows the connection to various kinds of data sources: IIoT physical devices such as industrial automation devices (PLCs, smart sensors, RFID readers, etc), ERP systems, SCADA/MES systems/data, and existing databases. It provides a collection of functionalities among which functionalities to collect data from and to send commands to such systems.

6.1.2 Functions / Features

Data Acquisition component provides a set of functionalities that could be grouped on the following features:

- **Data Source Adapters management**: where the installed adapters are shown. The Data Source Adapters are software classes that support the communication of specific data sources through specific open or closed protocols.
- **Data Source Management**: where data sources are registered as sources of information and/or receivers of actuation commands to interact with the physical world. The registration of data sources implies the specification of parameters and the usage of installed adapters.
- **Data Source data reading**: where a range of functionality is provided regarding the different mechanisms to read from the data sources (ie their sensors). The Data Acquisition component can implement synchronous and asynchronous read methods through its Data Source Adapter subcomponent.
- **Data Source controlling**: A set of data sources will provide actuators. In this case, and whenever the data source's associated adapter supports its control, zApps will be able to act on the data sources through the Data Acquisition component.

The functions can be grouped to the following features:

Subtask	Subtask description
T51A001 Receive asynchronous data from Data Source	**Priority:** Must
	Who: Data Acquisition Component **When / Where**: Runtime of the application depending on event's scheduling in the configuration **What:** Receive data pushed by a data source based on an event configured on it **Why:** Provide data to subscribed zApps or other assets
Acceptance Criteria	Make sure that the event is generated according to the configuration parameters provided by the adapter
Requirements filled	RQ_0008, RQ_0009, RQ_0010, RQ_0011, RQ_0012, RQ_0017, RQ_0022, RQ_0024, RQ_0029, RQ_0030, RQ_0040, RQ_0043, RQ_0044, RQ_0068, RQ_0076, RQ_0082, RQ_0093, RQ_0117, RQ_0136, RQ_0197, RQ_0246, RQ_0248, RQ_0291, RQ_0292, RQ_0293, RQ_0295, RQ_0296, RQ_0321, RQ_0322, RQ_0363, RQ_0390, RQ_0391, RQ_0412, RQ_0413, RQ_0448, RQ_0460, RQ_0461, RQ_0467, RQ_0468, RQ_0491, RQ_0515, RQ_0516, RQ_0549, RQ_0552, RQ_0572, RQ_0573, RQ_0575, RQ_0576, RQ_0611, RQ_0612, RQ_0664, RQ_0680, RQ_0723, RQ_0724, RQ_0739, RQ_0783, RQ_0798, RQ_0824, RQ_0825, RQ_0879, RQ_0884, RQ_0960, RQ_0962, RQ_0973, RQ_0975, RQ_0976, RQ_0977, RQ_0988, RQ_0989, RQ_0991, RQ_0992, RQ_1007, RQ_1008, RQ_1047, RQ_1048, RQ_1049, RQ_1050
T51A002 Send data for asynchronous data acquisition	**Priority:** Must
	Who: Data Acquisition Component **When / Where**: Runtime of the application depending on event's scheduling in the configuration

	What: Send data to the zApps or other assets through a push mechanism or to the Services and Message Bus Component through pub/sub mechanism **Why:** Provide asynchronously data to the ZDMP assets
Acceptance Criteria	Make sure that the data is pushed/published according to the zApps or other assets configuration provided
Requirements filled	RQ_0008, RQ_0009, RQ_0010, RQ_0011, RQ_0012, RQ_0017, RQ_0022, RQ_0032, RQ_0033, RQ_0080, RQ_0246, RQ_0248, RQ_0291, RQ_0292, RQ_0293, RQ_0295, RQ_0296, RQ_0321, RQ_0322, RQ_0412, RQ_0413, RQ_0460, RQ_0461, RQ_0467, RQ_0468, RQ_0515, RQ_0516, RQ_0549, RQ_0572, RQ_0573, 0575, RQ_0576, RQ_0680, RQ_0739, RQ_0798, RQ_0824, RQ_0825, RQ_0881, RQ_0886, RQ_1007, RQ_1008, RQ_1047, RQ_1048, RQ_1049, RQ_1050,
T51A003 **Receive data from Data Source (via polling process)**	**Priority:** Must
	Who: Data Acquisition Component **When / Where**: Runtime of the application depending on polling process configuration **What:** Periodically queries data from data sources that do not have a push-based mechanism **Why:** So that zApps or other assets can receive data
Acceptance Criteria	Make sure that the data source configuration stores the time interval between readings. Make sure that a background process retrieves applying the time interval seconds from data sources.
Requirements filled	RQ_0008, RQ_0009, RQ_0010, RQ_0011, RQ_0012, RQ_0017, RQ_0022, RQ_0024, RQ_0029, RQ_0030, RQ_0040, RQ_0043, RQ_0044, RQ_0068, RQ_0076, RQ_0082, RQ_0093, RQ_0117, RQ_0136, RQ_0197, RQ_0246, RQ_0248, RQ_0291, RQ_0292, RQ_0293, RQ_0295, RQ_0296, RQ_0321, RQ_0322, RQ_0363, RQ_0390, RQ_0391, RQ_0412, RQ_0413, RQ_0448, RQ_0460, RQ_0461, RQ_0491, RQ_0515, RQ_0516, RQ_0549, RQ_0552, RQ_0572, RQ_0573, 0575, RQ_0576, RQ_0611, RQ_0612, RQ_0664, RQ_0680, RQ_0723, RQ_0724, RQ_0739, RQ_0783, RQ_0798, RQ_0824, RQ_0825, RQ_0879, RQ_0884, RQ_0960, RQ_0962, RQ_0973, RQ_0975, RQ_0976, RQ_0977, RQ_0988, RQ_0989, RQ_0991, RQ_0992, RQ_1007, RQ_1008, RQ_1047, RQ_1048, RQ_1049, RQ_1050,
T51A004 **Receive synchronous data from Data Source**	**Priority:** Must
	Who: Data Acquisition Component **When / Where**: Runtime of the application whenever a zApp or other asset requests data reading a synchronous mechanism not controlled via polling process **What:** Will query data from data sources that do not have a push-based mechanism **Why:** So that zApps or other assets can receive data
Acceptance Criteria	Make sure that a range of specific data sources can be queried
Requirements filled	RQ_0008, RQ_0009, RQ_0010, RQ_0011, RQ_0012, RQ_0017, RQ_0022, RQ_0024, RQ_0029, RQ_0030, RQ_0040, RQ_0043, RQ_0044, RQ_0068, RQ_0076, RQ_0082, RQ_0093, RQ_0136, RQ_0197, RQ_0246, RQ_0248, RQ_0291, RQ_0292, RQ_0293, RQ_0295, RQ_0296, RQ_0321, RQ_0322, RQ_0363, RQ_0390, RQ_0391, RQ_0412, RQ_0413, RQ_0414, RQ_0415, RQ_0416, RQ_0448, RQ_0451, RQ_0460, RQ_0461, RQ_0462, RQ_0464, RQ_0466, RQ_0491, RQ_0494, RQ_0515, RQ_0516, RQ_0549, RQ_0552, RQ_0572, RQ_0573, 0575, RQ_0576, RQ_0611, RQ_0612, RQ_0664, RQ_0680, RQ_0723, RQ_0724, RQ_0739, RQ_0783, RQ_0798, RQ_0824, RQ_0825, RQ_0879, RQ_0884, RQ_0960, RQ_0962, RQ_0973, RQ_0975, RQ_0976, RQ_0977, RQ_0988, RQ_0989, RQ_0991, RQ_0992, RQ_1007, RQ_1008, RQ_1047, RQ_1048, RQ_1049, RQ_1050
T51A005 **Send data for synchronous data acquisition**	**Priority:** Must
	Who: Data Acquisition Component **When / Where:** Runtime of the application whenever a zApp or other asset requests data reading through a synchronous mechanism not controlled via polling process **What:** Will provide an API to interact with zApps or other component for synchronous data acquisition through pulling operations **Why:** So that zApps can read synchronous data on demand
Acceptance Criteria	Make sure that the API is released and connected with zApps / other components
Requirements filled	RQ_0008, RQ_0009, RQ_0010, RQ_0011, RQ_0012, RQ_0017, RQ_0022, RQ_0032, RQ_0033, RQ_0080, RQ_0246, RQ_0248, RQ_0291, RQ_0292, RQ_0293, RQ_0295, RQ_0296, RQ_0321, RQ_0322, RQ_0412, RQ_0413, RQ_0414, RQ_0415, RQ_0416, RQ_0451, RQ_0460, RQ_0461, RQ_0462, RQ_0464, RQ_0466, RQ_0494, RQ_0515, RQ_0516, RQ_0549, RQ_0555, RQ_0572, RQ_0573, 0575, RQ_0576, RQ_0680, RQ_0739, RQ_0798, RQ_0824, RQ_0825, RQ_0881, RQ_0886, RQ_1007, RQ_1008, RQ_1047, RQ_1048, RQ_1049, RQ_1050

T51A006 **Send command to Data Source**	**Priority:** Should
	Who: Data Acquisition Component **When / Where:** Runtime of the application whenever a zApp or other asset requests to act on a data source **What**: Will be able to send command on specific private protocols to data sources **Why**: So that zApps can be enrich with functionality to act on data sources according to certain criteria
Acceptance Criteria	Data sources will support commands, will be configured as command receivers, adapter implementation will support actions on specific protocols
Requirements filled	RQ_0393, RQ_0404, RQ_0405, RQ_0406, RQ_0407, RQ_0408, RQ_0453, RQ_0454, RQ_0455, RQ_0456, RQ_0457, RQ_0458, RQ_0505, RQ_0506, RQ_0507, RQ_0508, RQ_0509, RQ_0510, RQ_0511, RQ_0512, RQ_0513, RQ_0561, RQ_0562, RQ_0563, RQ_0564, RQ_0565, RQ_0566, RQ_0567, RQ_0568, RQ_0569, RQ_0571, RQ_0642, RQ_0643, RQ_0644, RQ_0645, RQ_0762
T51A007 **Register Data Source on ZDMP**	**Priority:** Must
	Who: User **When / Where:** Design of the application **What**: Set-up an existing data source as a source of information **Why**: So that zApps could receive, query, and possibly send commands to data sources
Acceptance Criteria	Only IT managers/administrators could register data sources on the platform for a specific company
Requirements filled	RQ_0003, RQ_0004, RQ_0005, RQ_0062, RQ_0063, RQ_0064, RQ_0167, RQ_0168, RQ_0389, RQ_0447, RQ_0489, RQ_0551, RQ_0610, RQ_0662, RQ_0722, RQ_0782
T51A008 **Register asynchronous sensors or other IIoT items of Data Source**	**Priority:** Must
	Who: User **When / Where:** Design of the application **What**: Register sensors of a given data source as asynchronous capable **Why**: So that zApps could receive data from data sources under certain events and time intervals
Acceptance Criteria	Only IT managers/administrators could register data sources on the platform for a specific company
Requirements filled	RQ_0003, RQ_0004, RQ_0005, RQ_0062, RQ_0063, RQ_0064, RQ_0389, RQ_0447, RQ_0489, RQ_0551, RQ_0610, RQ_0662, RQ_0722, RQ_0782,
T51A009 **Register synchronous sensors or other IIoT items of Data Source**	**Priority:** Must
	Who: User **When / Where:** Design of the application **What**: Register sensors of a given data source as synchronous capable **Why**: So that zApps could receive data from data sources when requesting proactively
Acceptance Criteria	Only IT managers/administrators could register data sources on the platform for a specific company
Requirements filled	RQ_0003, RQ_0004, RQ_0005, RQ_0062, RQ_0063, RQ_0064, RQ_0389, RQ_0489, RQ_0551, RQ_0610, RQ_0662, RQ_0722, RQ_0782
T51A010 **Configure Data Source as command receiver**	**Priority:** Should
	Who: User **When / Where:** Design of the application **What**: Configure a data source as a command receiver **Why**: So that zApps could send commands according to their needs
Acceptance Criteria	Only IT managers/administrators could configure data sources on the platform for a specific company as command receivers. The implementation of the adapter will offer the list of commands that could be asked for by the Data Acquisition Component under the zApp query
Requirements filled	RQ_0003, RQ_0062, RQ_0393, RQ_0404, RQ_0405, RQ_0406, RQ_0407, RQ_0408, RQ_0453, RQ_0454, RQ_0455, RQ_0456, RQ_0457, RQ_0458, RQ_0505, RQ_0506, RQ_0507, RQ_0508, RQ_0509, RQ_0510, RQ_0511, RQ_0512, RQ_0513, RQ_0561, RQ_0562, RQ_0563, RQ_0564, RQ_0565, RQ_0566, RQ_0567, RQ_0568, RQ_0569, RQ_0571, RQ_0642, RQ_0643, RQ_0644, RQ_0645, RQ_0762
T51A011	**Priority:** Must

List existing Data Sources already configured	**Who**: User (a) and Platform (b) **When / Where:** Design of the application **What**: List existing registered and configured data sources **Why**: So that the User (a) and/or the Platform (b) can acknowledge the sources of information (data source) that could provide data to specific zApps
Acceptance Criteria	Configuration of data sources should be carried about beforehand Data sources will be shown along with some attributes on a table
Requirements filled	RQ_0003, RQ_0062
T51A012 **Filter list of Data Sources according to name and type**	**Priority:** Should
	Who: User **When / Where:** Design of the application **What**: Filter the configured list of data sources by name and type **Why**: So that the User detect what are the sources of information configured
Acceptance Criteria	A search box will allow the filtering of the list of data sources by name and type
Requirements filled	RQ_0003, RQ_0062
T51A013 **Sort list of Data Sources by columns**	**Priority:** Should
	Who: User **When / Where:** Design of the application **What**: Sort the configured list of data sources by any of the columns **Why**: So that a better understanding of configured data sources could be achieved
Acceptance Criteria	Arrows beside each column will allow sorting of the table of data sources in ascending or descending order
Requirements filled	RQ_0003, RQ_0062
T51A014 **Check Data Source status**	Priority: Must
	Who: User (a) and Platform (b) **When / Where:** Runtime of the application **What:** Assess the status of the data sources **Why:** So that problems with data sources could be identified and solved
Acceptance Criteria	Each data source on a list of data sources should add a new column showing the status of the data source
Requirements filled	RQ_0342, RQ_0355, RQ_0627, RQ_0639
T51A015 **Check sensor or other IIoT item status**	Priority: Must
	Who: User (a) and Platform (b) **When / Where:** Runtime of the application **What:** Assess the status of each data source's sensor or other IIoT item. **Why:** So that problems with sensors or other IIoT item could be identified and solved
Acceptance Criteria	Each data source could be queried about its sensors or other IIoT items and a list of those with a new column showing the status of them should be provided
Requirements filled	RQ_0342, RQ_0355, RQ_0627, RQ_0639
T51A016 **List existing Data Source Adapters**	Priority: Must
	Who: User (a) and Platform (b) **When / Where:** Design of the application **What:** List existing installed adapters getting the detail on version, etc **Why:** So that the User (a) and the Platform (b) can acknowledge which data source will be able to be configured according to adapter/version/protocol
Acceptance Criteria	Adapters will be tagged with version and protocol

	List of adapters will be in a table and show name, protocol, version
Requirements filled	RQ_0003, RQ_0062
T51A017 Filter list of Data Source Adapters by Name	Priority: Should
	Who: User **When / Where:** Design of the application **What:** Filter the existing installed adapters by name **Why:** So that it can be detect if a specific adapter is installed properly
Acceptance Criteria	Adapters will be tagged with version and protocol A search box will allow the filtering of the list of adapters
Requirements filled	RQ_0003, RQ_0062
T51A018 Log messages from Data Acquisition component	Priority: Must
	Who: Data Acquisition Component **When / Where:** Runtime of the application **What:** Send logs of information, warnings, and errors to the Services and Message Bus **Why:** so that ZDMP platform can provide a unified log to Users
Acceptance Criteria	Data Acquisition Component should have logs with messages to be shown At least three levels of logs agreed with platform (information, warning error) A list of logs shown according to level of logs
Requirements filled	N/A
T51A019 Providing data model	Priority: Must
	Who: Data Acquisition Component **When / Where:** Runtime of the application **What:** Provide the data model to the Data Harmonization Component **Why:** Browse the existing data model for the purposes of data transformation and harmonization
Acceptance Criteria	Data Acquisition Component should have at least one data model schema stored
Requirements filled	RQ_0007, RQ_0046, RQ_0121, RQ_0135, RQ_0686, RQ_0719

Figure 202: Data Acquisition and IIoT Functions

6.1.3 Workflows

The following sub-sections describe the sequence diagrams describing the workflow of the function and the interactions needed with other components, apps, or users.

6.1.3.1 Data reading from Data Source

This feature allows any ZDMP asset to receive data from the data sources. Reading can be performed in synchronous or asynchronous way.

The asynchronous reading is based on a pub/sub mechanism. The message bus must subscribe to those events related to data receiving, and the adapter configures the data sources, so these transmit asynchronously the requested data whenever the event is triggered. Moreover, the asynchronous data reading can be performed also via a push-based mechanism for those zApps or other assets that provide it (T51A001/T51A002).

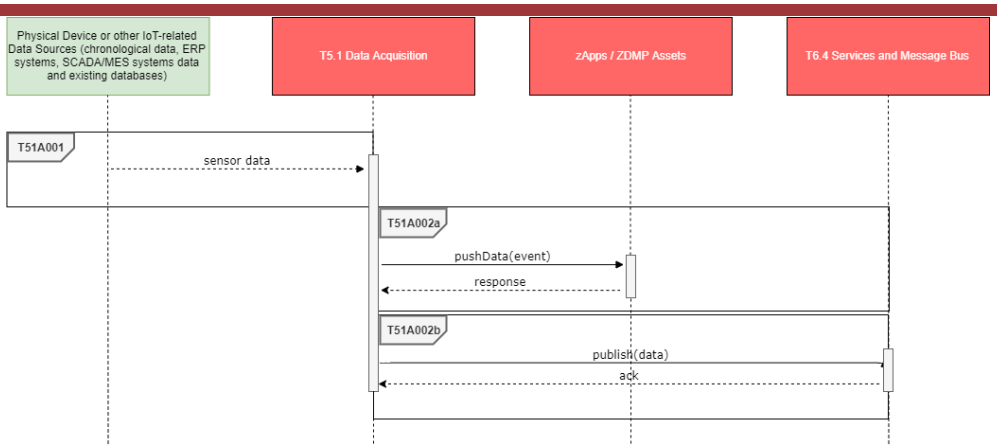


Figure 203: Asynchronous data reading from Data Source sequence diagram

Moreover, it is also possible to query data sources periodically, using a polling process (T51A003/T51A004), and on demand (T51A004/T51A005).

The first reading is enabled by an internal object or process in the Data Acquisition Component, which instructs periodically the Data Source Adapter to perform a query to a designated data source. The result then can be relayed using a pub/sub or push mechanism to the other ZDMP assets.

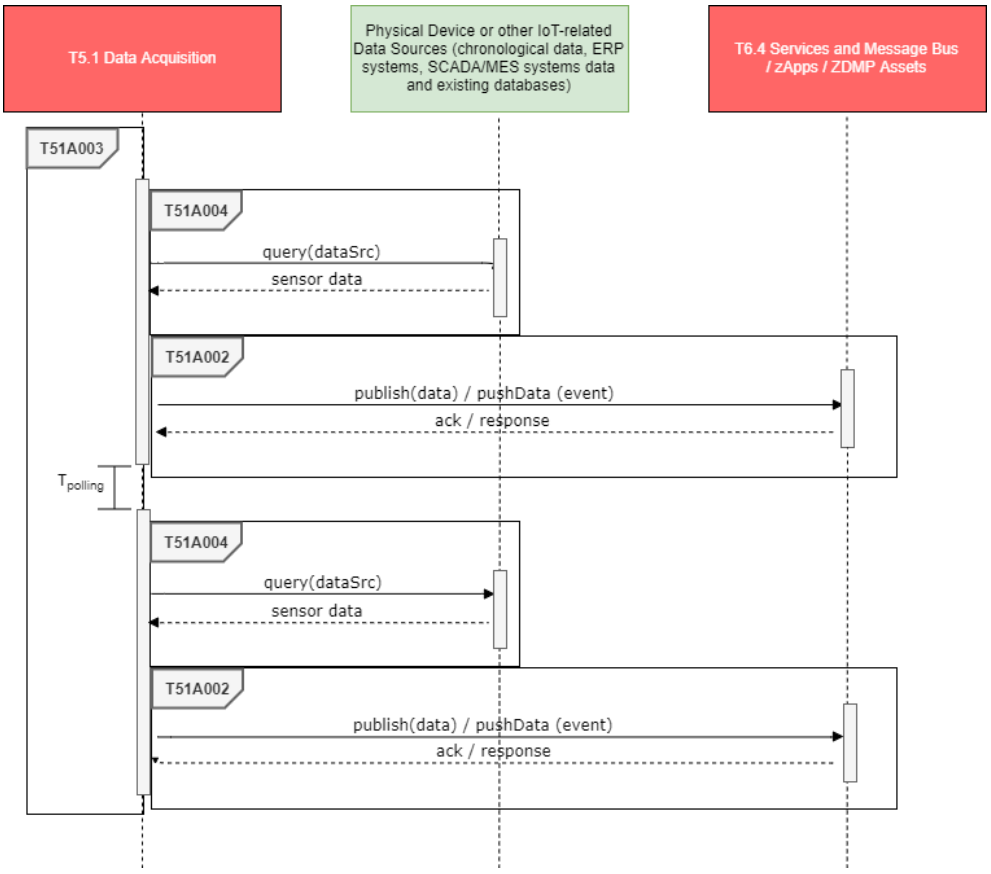


Figure 204: Polling process for reading sensors sequence diagram

The second reading corresponds to a pull mechanism allowing to read on demand data from data sources synchronously.

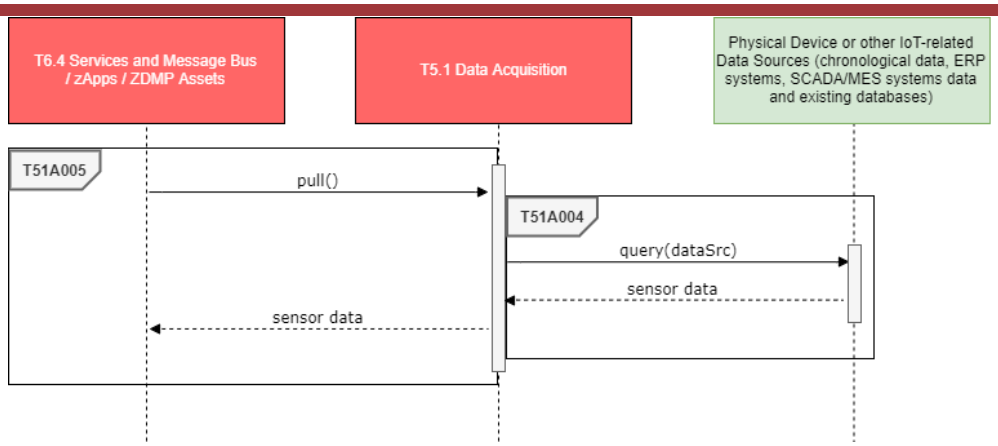


Figure 205: Synchronous data reading sequence diagram

6.1.3.2 Send command to Data Source

This feature allows any zApp to send command to those data sources that allow to be commanded; to do that the Data Source Adapter will use proprietary developments of specific data sources (T51A006).

6.1.3.3 Add/Configure new Data Source

This feature provides the capability to add an existing data source so that zApps can interact with compatible data sources.

Registering a data source implies indicating the type of driver/protocol the data source is using (that should have been previously installed), register the set of sensors or other IIoT items a data source uses to capture data, define for each sensor or other IIoT items the supported modes (synchronous, asynchronous), and / or if the data source can be controlled.

The main functionalities are:

- Register Data Source on ZDMP (T51A007)
- Register asynchronous sensors or other IIoT items of Data Source (T51A008)
- Register synchronous sensors or other IIoT items of Data Source (T51A009)
- Configure Data Source as command receiver (T51A010)

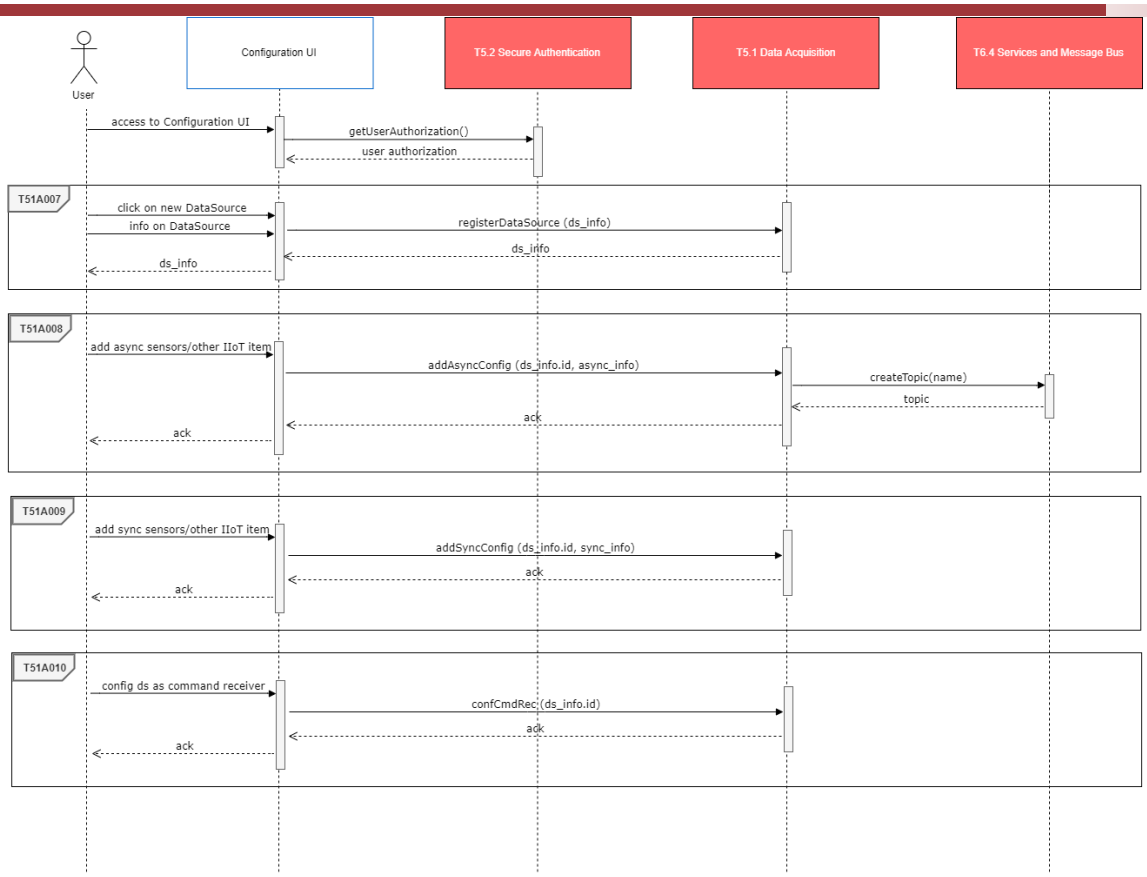


Figure 206: Data Source configuration on ZDMP sequence diagrams

6.1.3.4 Browsing existing Data Sources / Data Source Adapters

The feature provides the capability to browse data sources / data sources adapters that have been registered on ZDMP and filter the list according to their name and/or type. Some of these features are available both from the user side (through the Configuration UI) and from ZDMP platform side (through the Metadata interface).

The main functionalities from the user side are:

- List existing Data Sources already configured (T51A011a)
- Filter list of Data Sources according to name and type (T51A012)
- Sort list of Data Sources by columns (T51A013)
- Retrieve Data Source / Sensor status (T51A014a/T51A015a)
- List existing Data Source Adapters currently installed (T51A016a)
- Filter list of Data Sources Adapters according to name (T51A017)

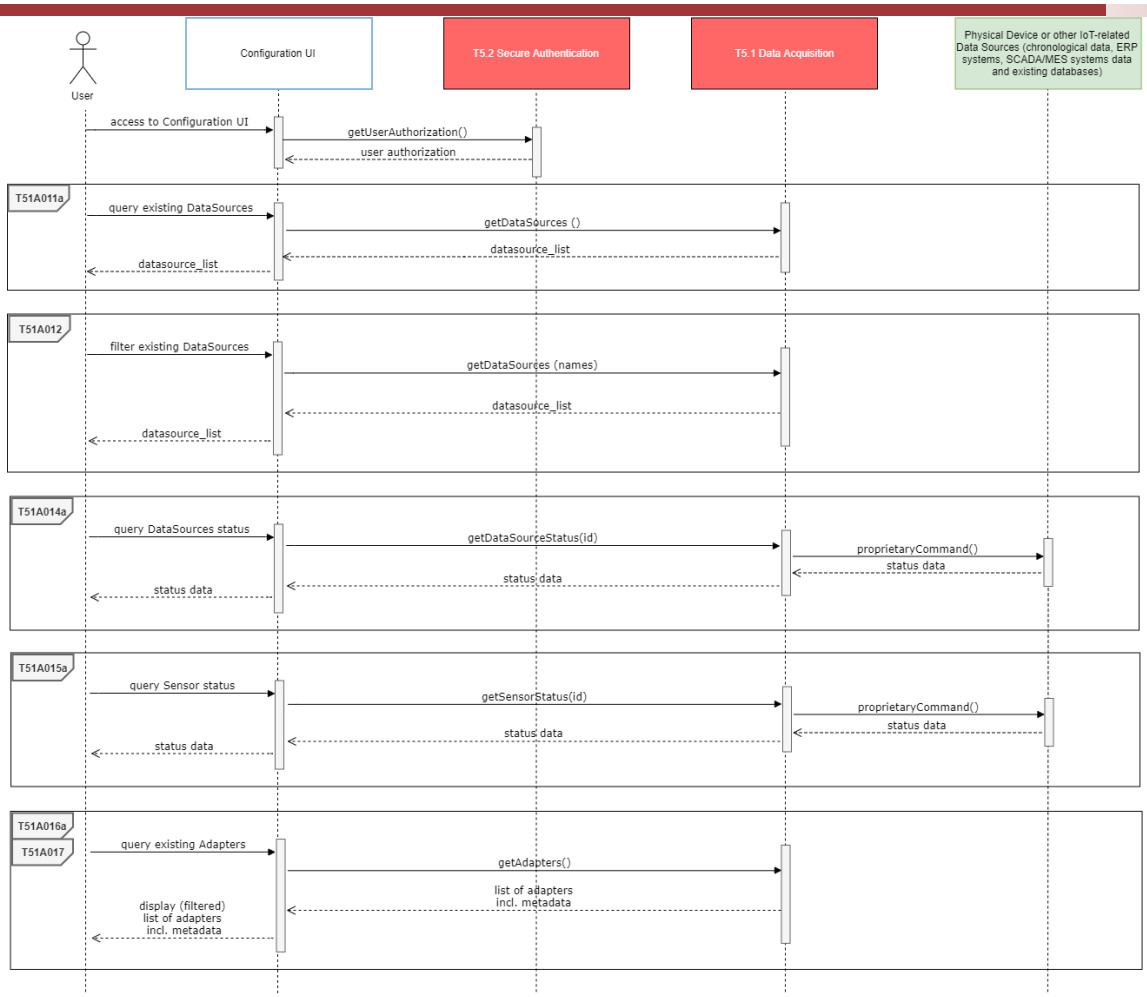


Figure 207: Browsing existing Data Sources / Data Source Adapters sequence diagram – user side

The main functionalities from the platform side are:

- List existing Data Sources already configured (T51A011b)
- Retrieve Data Source / Sensor status (T51A014b/T51A015b)
- List existing Data Source Adapters currently installed (T51A016b)

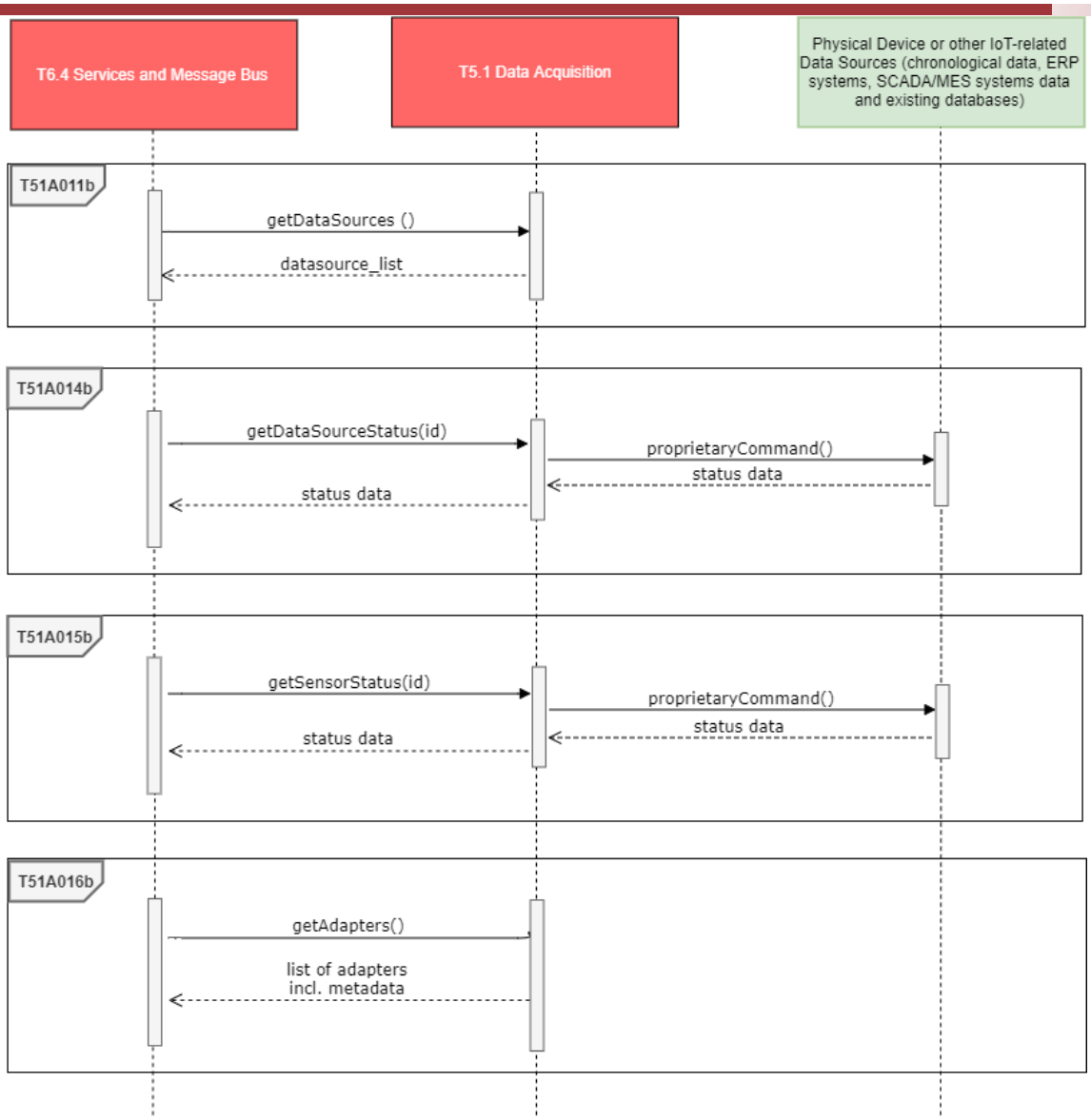


Figure 208: Browsing existing Data Sources / Data Source Adapters sequence diagram – platform side.

6.1.3.5 Log messages from Data Acquisition component

This feature (T51A018) allows the Data Acquisition component to log information, warning, and error messages of the registered data sources via the ZDMP platform.

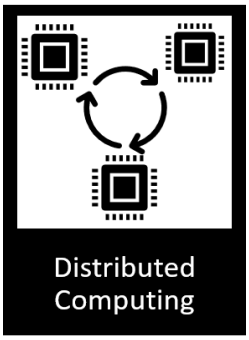
Once a Data Source Adapter receives an information, warning, or error message from its connected device in the device's proprietary format, it will transform the message to the ZDMP-internal format and send it to the Data Source Manager. The Data Source Manager will then log the message to the Services and Message Bus component via the Logging interface.

6.1.3.6 Provide data model to the platform

This feature (T51A019) allows the Data Harmonization Component to retrieve the existing data model for the purposes of data transformation and harmonization. Once the Data Acquisition component receives a request for the data model schema via the Data Source

Gateway interface, it is forwarded to the Data Source Manager that provides a response with the requested schema.

6.2 Distributed Computing (T5.5)



6.2.1 Overall functional characterization & Context

The Distributed Computing component is composed of an API to compute Tasks, an API and a HTML / CSS / JS based web frontend where users can map the location of the computational resources, to better outline the limitations of the execution of distributed tasks.

6.2.2 Functions / Features

The elements and functionalities that can be found in the component are described as follows:

- **Task Computing API**: This feature receives the tasks and their distributed level, and then compute the tasks.
- **Location Mapper:** With this feature the user accesses and defines the location of the computational resources

The function can be grouped to the following features:

Subtask	Subtask description
T55B001 Compute Task	**Priority:** Must
	Who: zApps, Orchestration Run-time, AI-Analytics Run-time **When:** Any time after resources are defined **Where:** Any computational resource located within the Location defined for the task to be computed. **What:** Compute assigned Task **Why:** Optimize performance, reducing overall processing time, using Cloud, Mist, Edge, and Fog computing.
Acceptance Criteria	Caller gets HTTP 202 response and a Json object with the Task Id.
Requirements filled	RQ_0152, RQ_0153, RQ_0154, RQ_0392, RQ_0393, RQ_0394, RQ_0395, RQ_0450, RQ_0451, RQ_0452, RQ_0495, RQ_0554, RQ_0555, RQ_0556, RQ_0614, RQ_0661, RQ_0666, RQ_0706, RQ_0726, RQ_0779, RQ_0785, RQ_0829
T55B002 CRUD Resources Location	**Priority:** Must
	Who: Administrator / User **When:** Any time **Where:** The Distributed Computed API or the Location Mapper UI **What:** Define / Update / Remove resources location in a scope of Cloud, Mist, Edge, and Fog Computing. **Why:** Map resources location, allowing the execution of tasks in the proper environment, considering the privacy and the resources needed.
Acceptance Criteria	Caller gets HTTP response, and a Json object with the Location object.
Requirements filled	RQ_0152, RQ_0153, RQ_0154, RQ_0392, RQ_0393, RQ_0394, RQ_0450, RQ_0451, RQ_0452, RQ_0495, RQ_0554, RQ_0555, RQ_0556, RQ_0614, RQ_0661, RQ_0666, RQ_0706, RQ_0726, RQ_0779, RQ_0785, RQ_0829
T55B003 Get Map of Resources Location	**Priority:** Must
	Who: Users, Application Runtime, Secure Installation **When:** Design-time / Runtime of the application **Where**: Anywhere on the same level the application publishes events to the bus **What:** Get the Map of resources location. **Why:** Adhere to the spatial tagging of computing resources, to execute tasks in the proper environment.

Acceptance Criteria	Caller gets HTTP 200 OK response, and a Json object with a list of the resource's locations.
Requirements filled	RQ_0614, RQ_0661, RQ_0666, RQ_0706, RQ_0726, RQ_0779, RQ_0785, RQ_0829
T55B004 Get Resource Location	**Priority:** Must
	Who: Users, Application Runtime, Secure Installation **When / Where**: Runtime of the application **What:** Get the defined location of a resource. **Why:** Adhere to the spatial tagging of computing resources, to execute tasks in the proper environment.
Acceptance Criteria	Caller gets HTTP 200 OK response, and a Json object with the resource location.
Requirements filled	RQ_0614, RQ_0661, RQ_0666, RQ_0706, RQ_0726, RQ_0779, RQ_0785, RQ_0829

Figure 209: Distributed Computing Features

6.2.3 Workflows

6.2.3.1 Compute Task

The Distributed Computing component receives tasks to be computed from zApps and ZDMP assets to compute, the result of the computation of the tasks are sent back to the component that requested the computation.

The main steps are:

- Receive a task (API calls) to be computed, with the spatial location where it can be executed, and the API call that should be invoked when the computation of the task is completed
- Obtain the list of computational resources within the spatial location defined
- Execute the task in the computational resources previously found
- Send the result of computing the task, invoking the API call defined

6.2.3.2 Define Resource Location

The Distributed Computing component allows users and other ZDMP assets to define, edit and remove the resources spatial location, composed of MachineID (Mist-level), LocationID (Edge-level) and SiteID (Fog-level). Only one of the IDs should be used, as there should always be a clear preference where to run a service. If no IDs are set, the task is expected to also be runnable in the Cloud. Spatial locations are saved in the Storage and used to identify where tasks can be computed, considering the scope of Mist, Edge, Fog and Cloud Computing. The processes of editing and deleting the locations spatial resources are not described in the diagram.

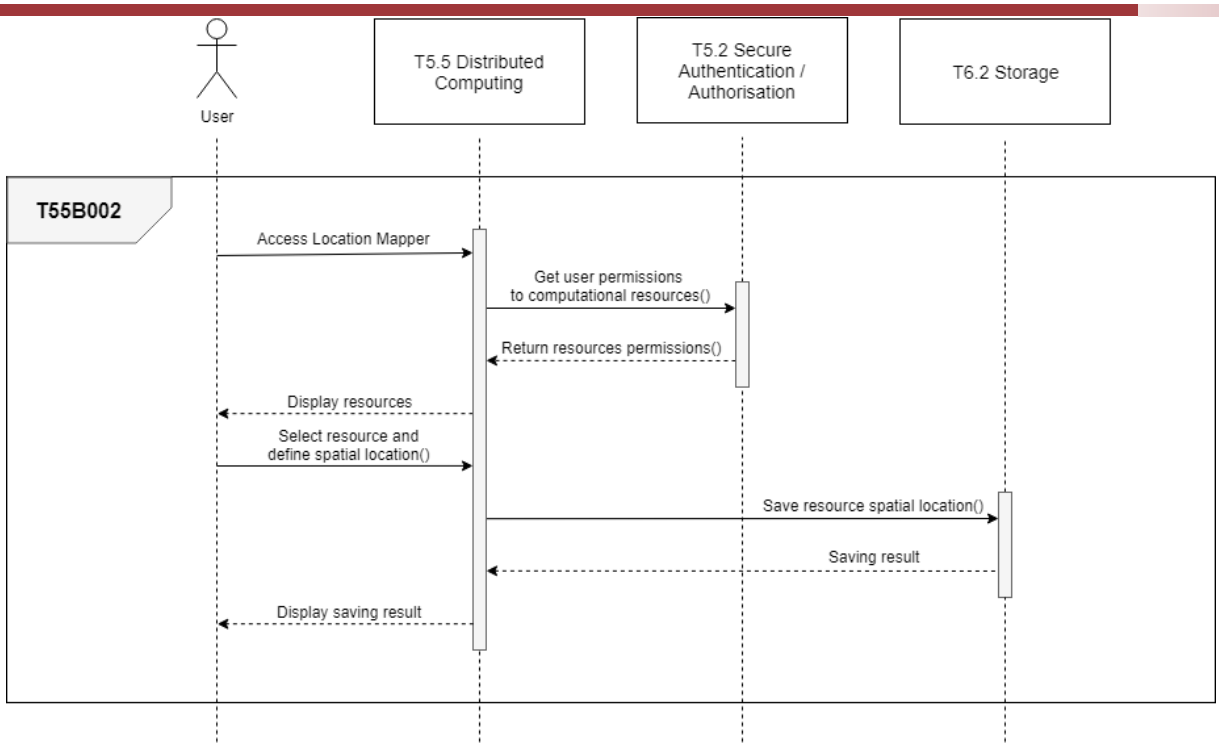


Figure 210: Define resources spatial location

6.2.3.3 Get Map of Resources Location

The Distributed Computing component allows users and other ZDMP assets to obtain the map of resources from a specific location, indicating each resource present in the selected location, its MachineID (Mist), LocationID (Edge) and SiteID (Fog).

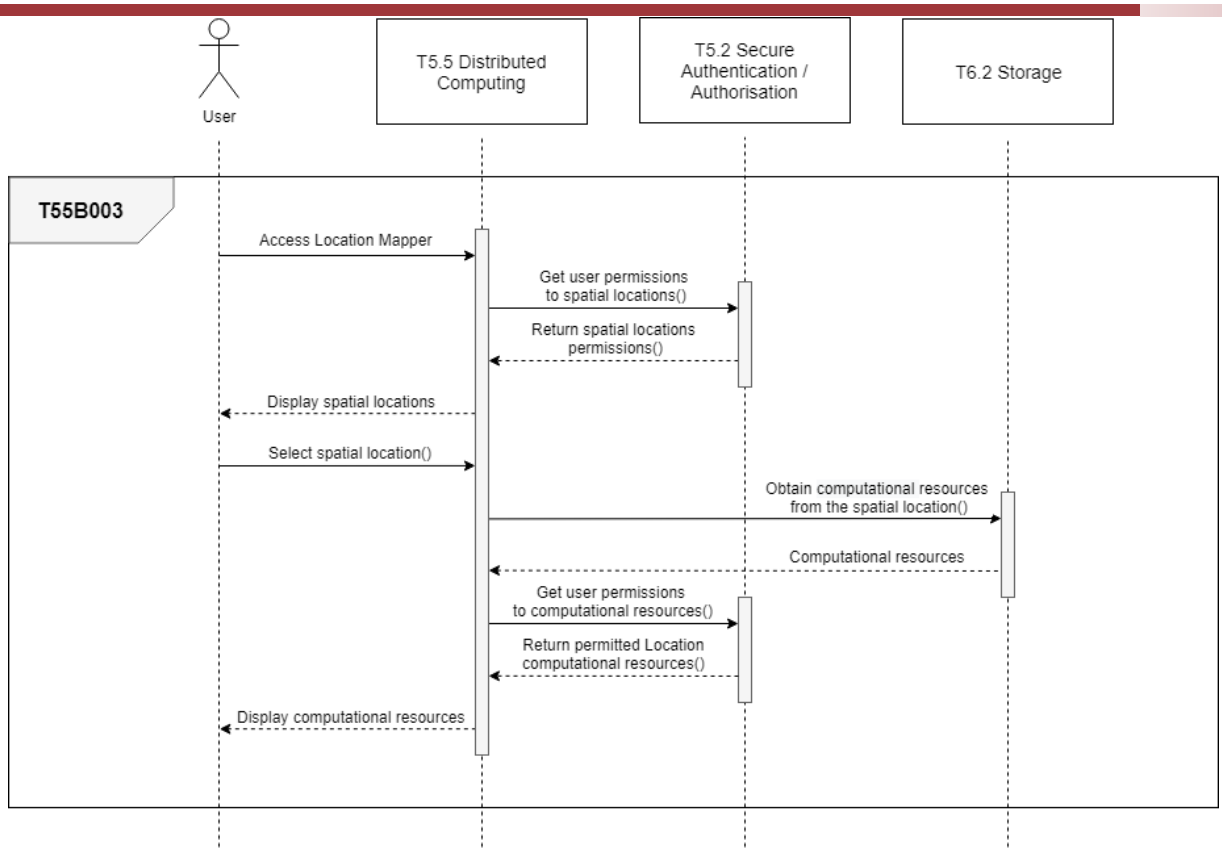


Figure 211: Get resources from spatial location

6.2.3.4 Get Resources Location

The Distributed Computing component allows users and other ZDMP assets to obtain the location of a resource, indicating each its location, composed of MachineID (Mist), LocationID (Edge) and SiteID (Fog).

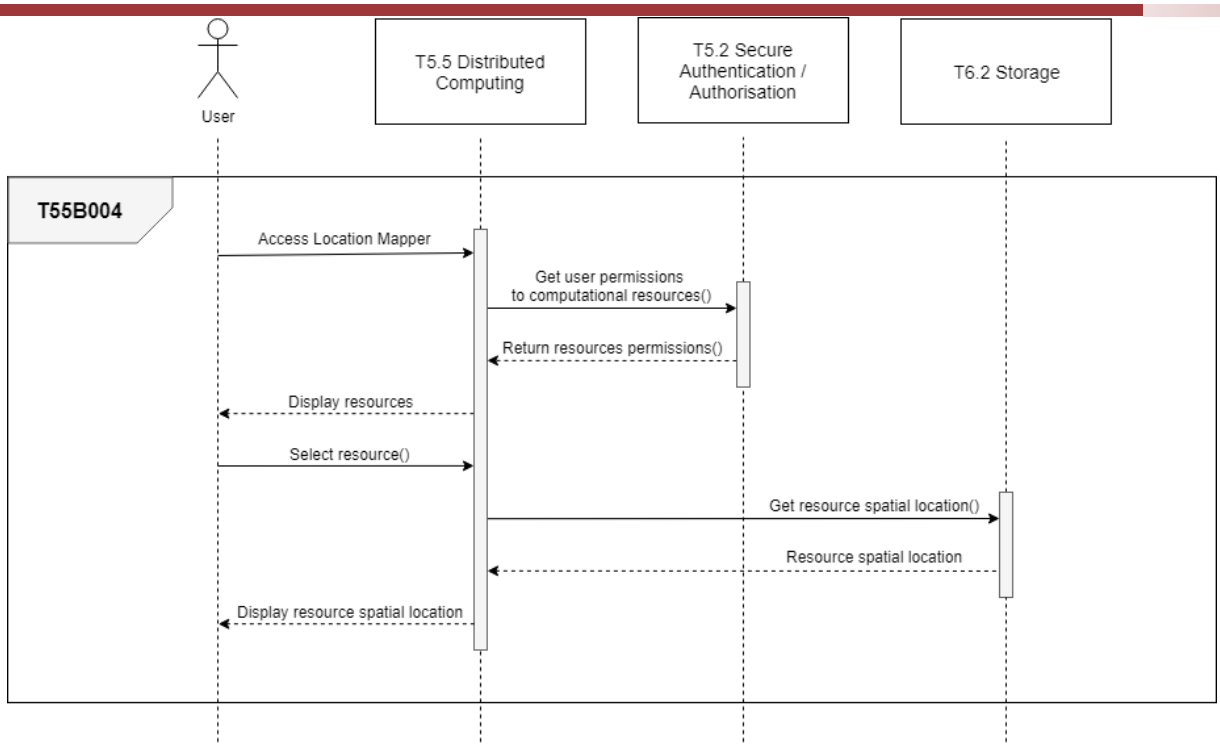
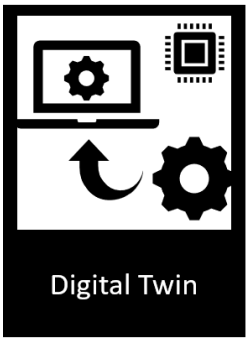


Figure 212: Get spatial location from resource

6.3 Digital Twin (T7.5, T8.1)



This section describes the functional components of the Digital Twin to support T7.3 and T8.1. The main feature provided by this component is a digital representation of the current state of manufacturing assets, processes, and products. Such a model will enable use cases to fulfil some of the objectives described in:

- T7.3 to develop models to detect anomalies in the consumption and infer probable future defects related to them, therefore improving the predictive capabilities of the system
- T8.1 Characterization and Modelling to develop virtual product and process modelling (digital twins)

With these objectives in mind, these sections detail the following functional modules:

- Model Loader
- Model Builder
- Execution Manager
- Simulation Engine

The following figure shows an architecture diagram on how these components relate to the rest of the components of the platform and is represented here due to slight updates from the Architecture deliverable D4.3.1.

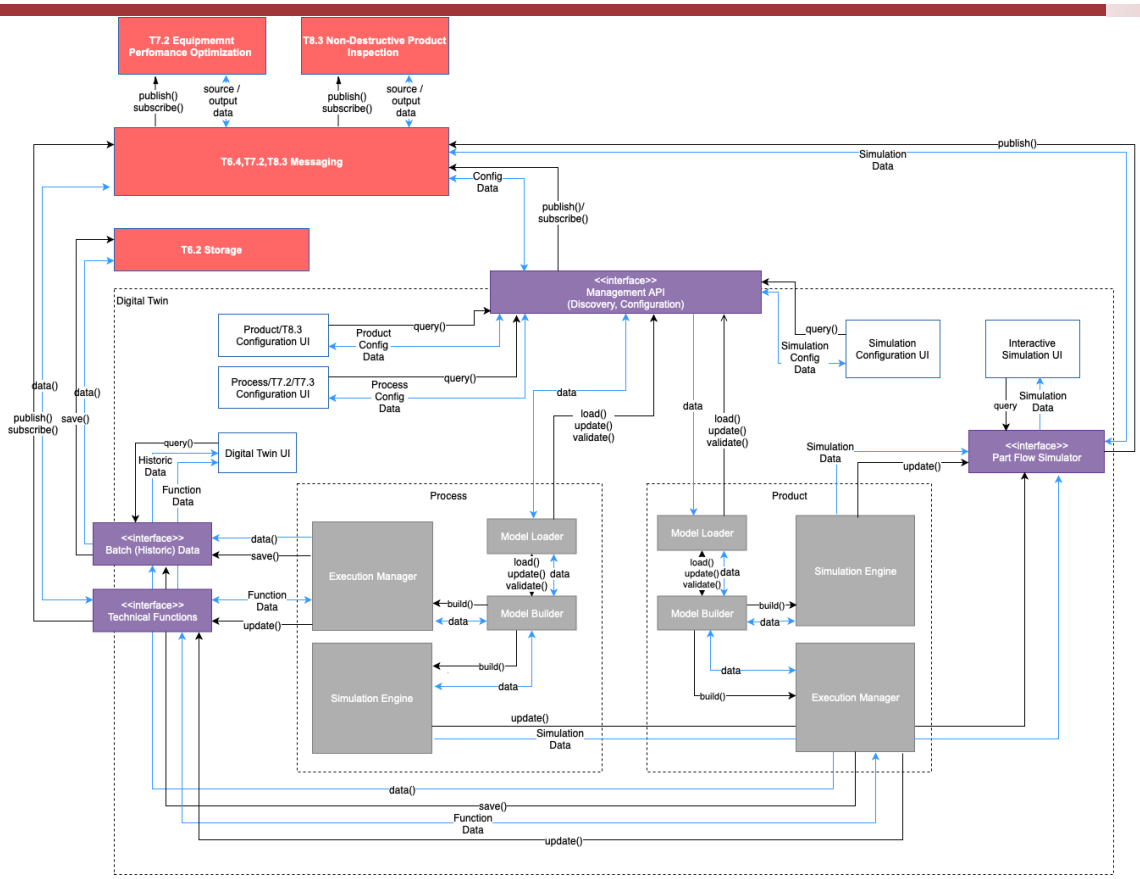


Figure 213: Digital Twin Component interaction diagram

6.2.4 Overall functional characterization & Context

Digital twin refers to a digital replica of potential and actual physical assets (physical twin), containing processes and products that can be used for various purposes. With the digital twin is possible to represent and model processes and products features (ie physical characteristics, bill of materials, tolerances, etc). Moreover, it provides data objects describing various aspects of the physical and logical parts of a manufacturing process. Additionally, it also includes the status of the different (potentially distributed) components of the manufacturing system and product features. A digital twin allows to simulate the future state of the manufacturing process or product production using AI algorithms to perform a dynamic virtual representation.

6.2.5 Functions / Features

The main functions of the digital twin are:

- **Model loader:** this function allows to load an instance of virtual representation of a process or a product. It acts as connector between the Management API interface and the model builder. It also allows to update a previous configuration with new parameters and publish the model that must be simulated. That information is sent the model builder in a compatible format and it forwards configuration updates and validation information from the model builder to the Management API interface.
- **Model Builder:** builds a digital model according to the configuration provided by the model loader into the execution engine implemented in the Process Execution Manager and to the Simulation Engine with the algorithms that simulate using the

selected parameters. The model builder implements functions to update the configuration and validate the model.

- **Process execution manager:** Contains the execution engine that manages the status information of the model. The information is collected from sensors through the Storage or Message Bus components. Information is saved in time data series in the Storage via the Batch (historic) Data interface.
- **Simulation Engine**: Runs the simulation model using a process-based, discrete-event simulation framework, selected when the model is loaded, applying the provided configuration parameters. The results are available via the Part-Flow simulator interface.

These functions can be further decomposed into the following subtasks that have to be realised:

Subtask	Subtask description
DT001 **Product configuration**	**Priority:** Must
	Who: Process Engineer **When:** Design-time **Where:** Runtime **What:** User selects the parameters of the product to be executed o simulated **Why:** To configure the product model according to the provided configuration
Acceptance Criteria	If successful, the user is prompted with a confirmation message. If not, the user is prompted with an error message
Requirements filled	RQ_0055, RQ_0056, RQ_0104, RQ_0125, RQ_0126, RQ_0137, RQ_0144
DT002 **Process configuration**	**Priority:** Must
	Who: Process Engineer **When:** Anywhere **Where:** Design-time **What:** User selects the parameters of the process to be executed o simulated **Why:** To configure the process model according to the provided configuration
Acceptance Criteria	If successful, the user is prompted with a confirmation message. If not, the user is prompted with an error message
Requirements filled	RQ_0055, RQ_0056, RQ_0104, RQ_0125, RQ_0126, RQ_0137, RQ_0144
DT003 **Load Model**	**Priority:** Must
	Who: Process Engineer or from messaging coming from other modules **When:** After DT001 and DT002 **Where:** Runtime **What:** User loads the model **Why:** To load/update a new model instance with the provided configuration
Acceptance Criteria	If successful, the user is prompted with a confirmation message to validate the model. If not, the user is prompted with an error message
Requirements filled	RQ_0055, RQ_0056, RQ_0104, RQ_0125, RQ_0126, RQ_0137, RQ_0144
DT004 **Execute model**	**Priority:** Must
	Who: Process Engineer **When:** After DT003 **Where:** Runtime **What:** User confirms the execution of the model **Why:** To execute the digital representation of the model according to the provided configuration
Acceptance Criteria	If successful, the user is prompted with a confirmation message. If not, the user is prompted with an error message
Requirements filled	RQ_0055, RQ_0056, RQ_0104, RQ_0125, RQ_0126, RQ_0137, RQ_0144
DT005	**Priority:** Must

Simulate model	**Who:** Process Engineer **When:** After DT003 **Where:** Runtime **What:** User confirms the simulation of the model **Why:** To execute the simulation of the model according to the provided configuration
Acceptance Criteria	If successful, the user is prompted with a confirmation message. If not, the user is prompted with an error message
Requirements filled	RQ_0055, RQ_0056, RQ_0104, RQ_0125, RQ_0126, RQ_0137, RQ_0144
DT006 **Visualise Output Data**	**Priority:** Must
	Who: Process Engineer **When:** After DT004 and DT005 **Where:** Runtime **What:** User visualize the model/simulation **Why:** To visualise the data inferred from the real and simulated model
Acceptance Criteria	If successful, the user is prompted with a confirmation message. If not, the user is prompted with an error message
Requirements filled	RQ_0055, RQ_0056, RQ_0104, RQ_0125, RQ_0126, RQ_0137, RQ_0144

Figure 214: Digital Twin Functions

6.2.6 Workflows

This workflow represents the overall function of the digital twin for executing and simulating a process or a product.

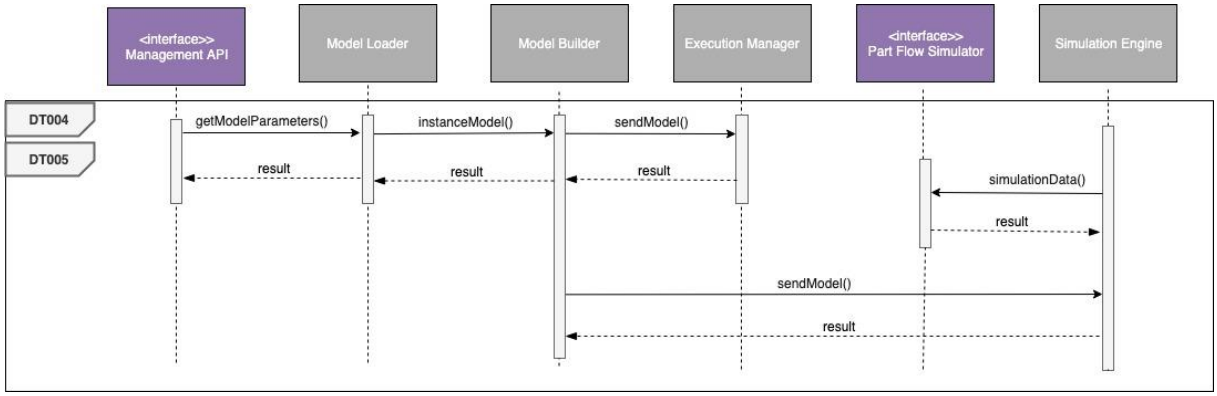


Figure 215: Digital twin process or product sequence diagram

6.3 Quality Inspection Configuration and Execution (T8.3)

This section describes the functional specifications of task T8.3 component. The key features provided by the component are a set of services to build zApps for non-destructive product quality inspection. These services will enable use cases in the project to fulfil some of the objectives described in WP8, mainly:

- **O8.3:** To manage non-destructive inspection of products

Quality Engine Manager is the main component of the Non-Destructive Product Inspection process: it receives data (images, hyperspectral-images, X-RAYs, …) related to product/production that can be used to analyse product quality. Quality Engine Manager elaborates input data using a set of predefined and configurable rules and its working parameters. Usually data gathered from an industrial process cannot used “as-is” and at this stage we consider that pre-processing images phase is part of Quality Engine processes.

Based on the specialization that this component is designed for, a specific processing algorithm for quality inspection purposes is implemented: some of these are listed as distinct modules inside the Quality Engine Manager, as all of them fit the component generic architectural schema. The developer using the platform chooses from a list the most appropriate one(s) to match the product inspection requirements of its own application; the list can be extended when new processing algorithms are needed, implemented and added on the platform, for this reason a generic New Classic Function/New AI Modeller feature is included.

Indeed, the nature of these processing algorithms can be of two types, AI-based and non-AI based (called “Classic” below). The latter includes

- Silhouette (object contour) Extractor
- Silhouette Comparator (typically used in tandem with previous one)
- Features Extraction
- Geometrical Features Analyser
- Object Definition
- Collision Detector

while the former includes

- AI Image Classifier
- AI Image Labeller

forming together a set of analytical tools which helps solving most of the product inspection cases coming from Requirements Analysis D4.1 deliverable.

The processing tools based on AI models are more demanding in designing and preparation due to the training procedure, as showed in the left part of the architecture schema, which only deals with AI modelling. To reflect this distinction, the functional specification description is split in two sections. The first one, described here, is related to the deployment and run-time execution of the component, which embraces all processing algorithms, including the AI-based ones once the AI model is trained and available for execution. The second section is related to the design and training of the AI-based algorithms, due to the peculiarity of the technology.

6.3.1 Overall functional characterization & Context

Based on the specialization that this component has been implemented for, it will provide the end user/zApp using it with product quality related information (defects, quality rate, …) or with any other intermediate output which can be useful to assess product quality KPIs. Run-time relevant parameters of the components are configurable as well the source of the input to inspection algorithm should be determined by the user of the component.

6.3.2 Functions / Features

- **Configure parameters**: Configuration parameters can be set by a user via the user interface or by an external component/zApp via the update of configuration input files. The run-time relevant parameters of the component are configurable.
- **Select data source**: Through the user interface the user can select the source type and the data sources that provide the required data to the algorithm. Input data can come from Message Bus or Storage, which determines the continuous or one-time operation of the component.
- **Select data destination:** Through the user interface the user can select the destination type where to send processed data and results from algorithm. Output data can be sent to Message Bus or saved on Storage.
- **Execute analysis**: The analysis is performed automatically when new data is published for the input source or by request via the user interface. The component produces one output per call.

These functions can be further decomposed into the following subtasks that have to be realised:

Subtask	Subtask description
T83A001 **Select analysis type**	**Priority:** Must
	Who: Product Engineer or zApp Developer **Where:** Anywhere **When:** At design-time, as a first step **What:** Select the processing algorithm to be used to process data for quality inspection **Why:** To select the needed processing algorithm
Acceptance Criteria	The selected algorithm belongs to the list of available ones
Requirements filled	N/A
T83A002 **Select streaming data source**	**Priority:** Must
	Who: Product Engineer or zApp Developer **When:** At design-time, after T83A001 **Where:** Anywhere **What:** Specifies the input source needed for quality inspection, by subscribing a specific topic on the Message Bus **Why:** To provide the source to the processing algorithm
Acceptance Criteria	The user can select the data sources that have been previously made available in the platform
Requirements filled	RQ_0297, RQ_0387, RQ_0445, RQ_0493, RQ_0549, RQ_0627, RQ_0639, RQ_0680, RQ_0739, RQ_0798
T83A003 **Specify historical data source**	**Priority:** Must
	Who: Product Engineer or zApp Developer **Where:** Anywhere **When:** At design-time, after T83A002 **What:** Specifies the input data needed for quality inspection from the Storage **Why:** To provide the proper input to the processing algorithm
Acceptance Criteria	The user can select the data that have been previously saved

Requirements filled	N/A
T83A004 **Select streaming data destination**	**Priority:** Must
	Who: Product Engineer or zApp Developer **Where:** Anywhere **When:** At design-time, after T83A001 **What:** Select the output/results destination by indicating a specific topic to be published on the Message Bus **Why:** To provide output broadcast
Acceptance Criteria	The source of output is known
Requirements filled	RQ_0302, RQ_0387, RQ_0445, RQ_0493, RQ_0549, RQ_0627, RQ_0639, RQ_0680, RQ_0697, RQ_0698, RQ_0699, RQ_0703, RQ_0739, RQ_0798
T83A005 **Specify historical data destination**	**Priority:** Must
	Who: Product Engineer or zApp Developer **Where:** Anywhere **When:** At design-time, after T83A004 **What:** Specifies the output data destination on the Storage **Why:** To store output including results
Acceptance Criteria	The user can select the data destinations that are available
Requirements filled	RQ_0302, RQ_0387, RQ_0445, RQ_0493, RQ_0549, RQ_0627, RQ_0639, RQ_0680, RQ_0697, RQ_0698, RQ_0699, RQ_0703, RQ_0739, RQ_0798
T83A006 **Configure**	**Priority:** Must
	Who: Product Engineer or zApp Developer **Where:** Anywhere **When:** At design-time, after T83A001; at run-time as well **What:** Configures the run-time relevant parameters of the component. When dealing with AI based algorithms, this implies selecting the trained model to be used **Why:** needed for the component to function properly as desired
Acceptance Criteria	The necessary parameters for functioning of the component are known
Requirements filled	*RQ_0309, RQ_0395, RQ_0557, RQ_0558, RQ_0815*
T83A007 **Execute the analysis**	**Priority:** Must
	Who: Quality Engine Manager **Where:** Anywhere **When:** Product Engineer or zApp Developer ask for quality inspection processing, or automatically through message bus publication of new input data from production. At run-time **What:** performs the quality analysis **Why:** To have the quality results as output of the component
Acceptance Criteria	The component produces the expected output
Requirements filled	RQ_0310, RQ_0302, RQ_0387, RQ_0445, RQ_0493, RQ_0549, RQ_0627, RQ_0639, RQ_0680, RQ_0687, RQ_0739, RQ_0798, RQ_0815
T83A008 **Send analysis results**	**Priority:** Must
	Who: Quality Engine Manager **Where:** Anywhere **When:** Product Engineer or zApp Developer ask for processing algorithm results. At run-time **What:** sends the quality analysis results **Why:** To allow prompt access to the quality results as output of the component
Acceptance Criteria	The component produces the expected output
Requirements filled	RQ_0310

Figure 216: Quality Inspection Configuration and Execution Functions

6.3.3 Workflows

6.3.3.1 Select analysis type

The following diagram explains this function and the necessary interactions with other components.

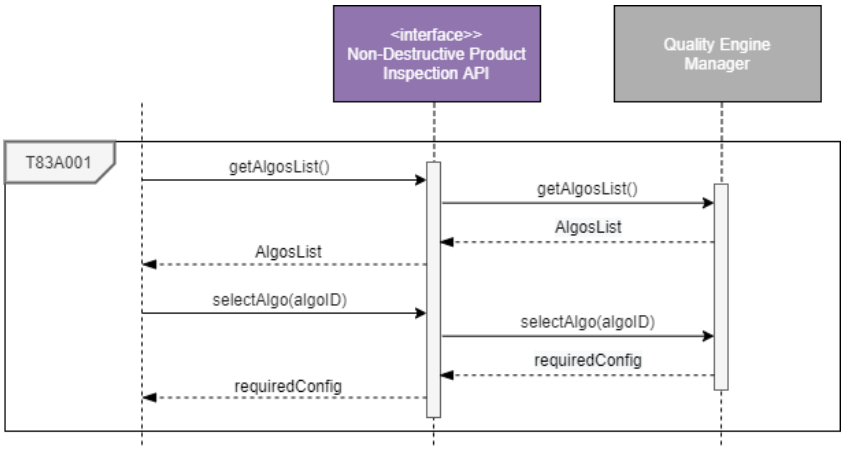


Figure 217: Select Analysis Type

6.3.3.2 Select streaming data source

The following diagram explains this function and the necessary interactions with other components.

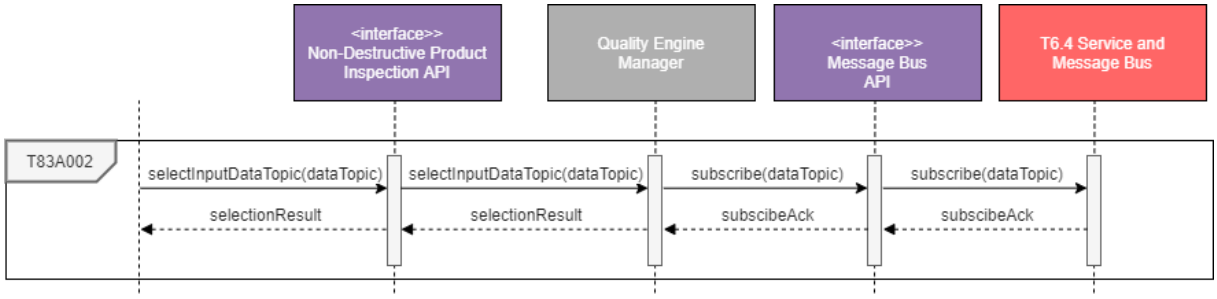


Figure 218: Select Streaming Data Source

6.3.3.3 Specify historical data source

The following diagram explains this function and the necessary interactions with other components.

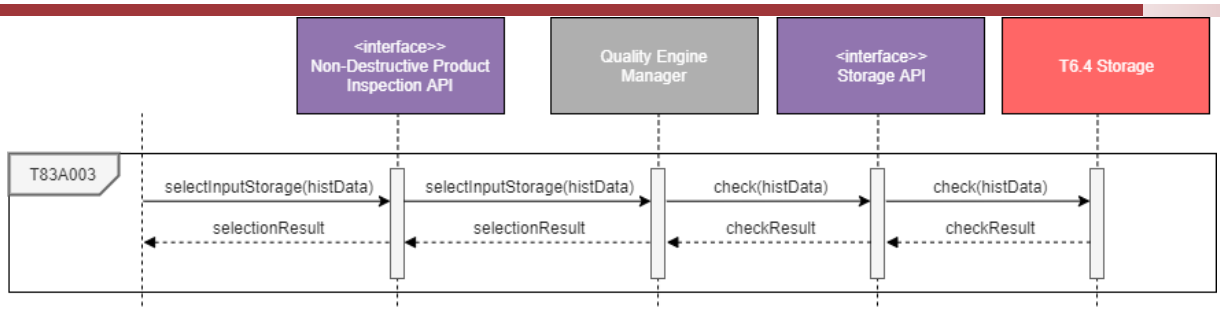


Figure 219: Specify Historical Data Source

6.3.3.4 Select streaming data destination

The following diagram explains this function and the necessary interactions with other components.

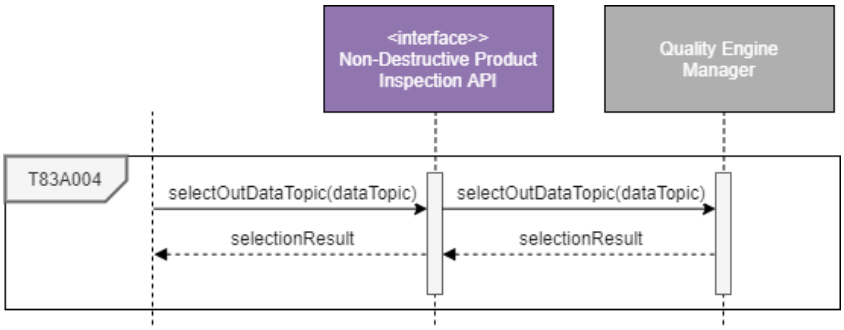


Figure 220: Select Streaming Data Destination

6.3.3.5 Specify historical data destination

The following diagram explains this function and the necessary interactions with other components.

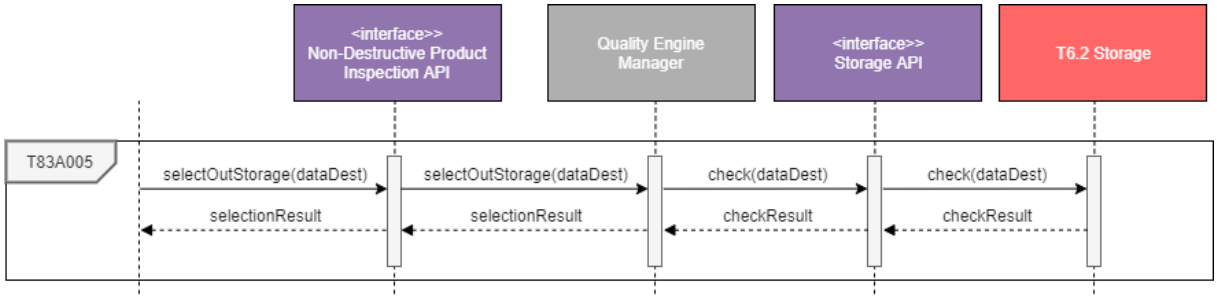


Figure 221: Specify Historical Data Destination

6.3.3.6 Configure

The following diagram explains this function and the necessary interactions with other components.

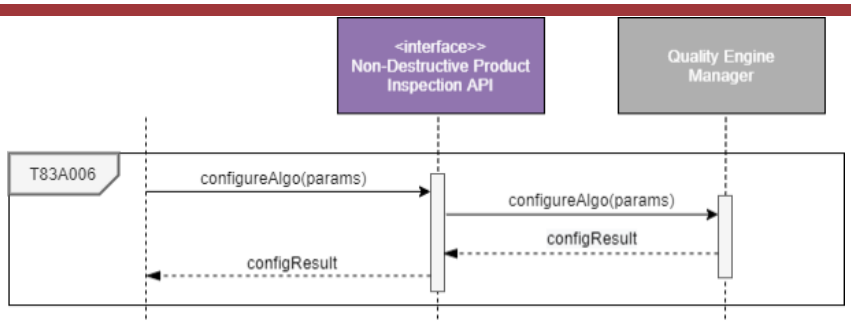


Figure 222: Configure Sequence Diagram

6.3.3.7 Execute analysis

The following diagram explains this function and the necessary interactions with other components.

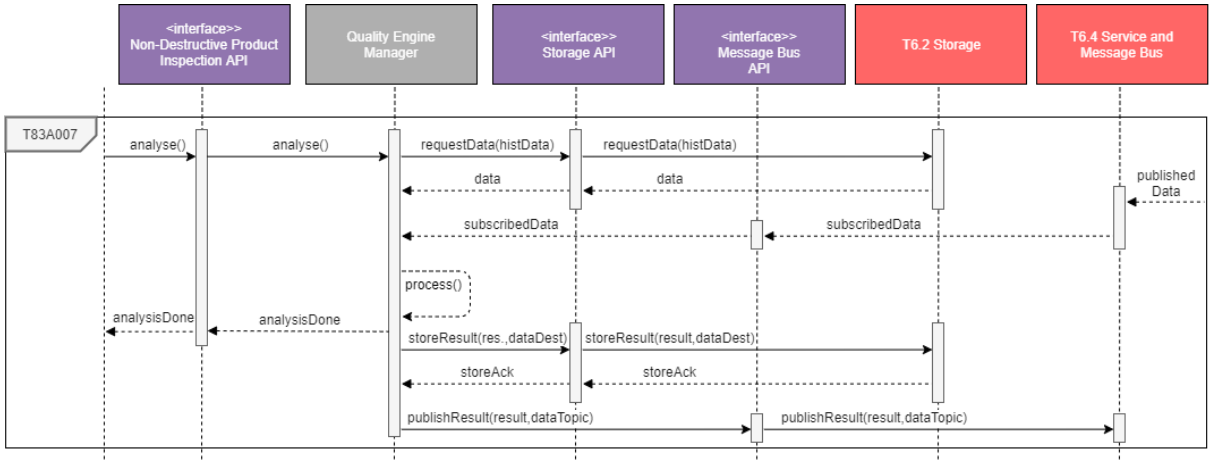


Figure 223: Execute Analysis

6.3.3.8 Send analysis results

The following diagram explains this function and the necessary interactions with other components.

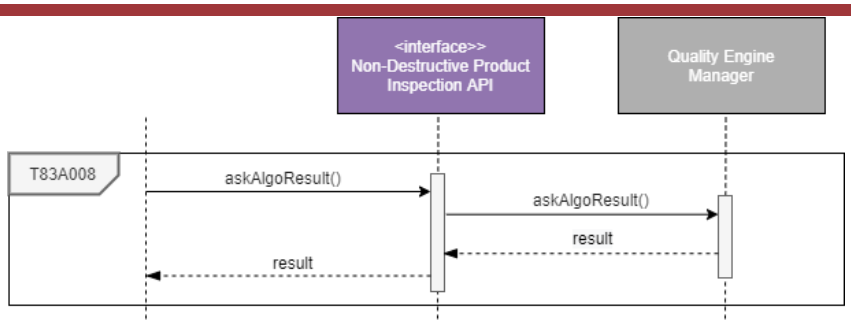


Figure 224: Send Analysis Results

6.4 Quality AI Inspection: Design and Training (T8.3)

6.4.1 Overall functional characterization & Context

This module is part of Non-Destructive Inspection (T8.3). When using supervised AI models to extract quality related information through analysis of production data, a model training process is needed before model deployment and final execution for quality analysis. The training process might be re-run when better or more extendedly labelled datasets are provided or when a new product type is considered.

6.4.2 Functions / Features

- **AI model selection:** To select between available models associated to the specific AI Engine (ie Image Classifier, Image Labeller, etc). This feature allows to select the model from the model storage, to (re)train it with available data sets.
- **Training Data source configuration:** To configure the data sources that provides the required data to the training process. This feature allows to select the data source used to collect the required input to train and evaluate the model. Therefore, the data source should be a valid data source (either external or internal) already configured in the ZDMP Platform.
- **Training parameters configuration:** To configure different properties to be used to train the model, like data format, accuracy thresholds, hyper-parameters configuration.
- **Train AI model:** To train the created model using the provided model, data sequence and configuration.
- **Store trained AI model:** To store the trained model to make it available for run-time deployment

These functions can be further decomposed into the following subtasks that have to be realised:

Subtask	Subtask description
T83B001 **Select AI Engine**	**Priority:** Must
	Who: Product Engineer or zApp Developer **Where:** Anywhere **When:** At design-time when AI models need to be selected for training **What:** End user selects specific AI processing engine (AI Image Classifier, AI Image Labeller, …) to train a model associated to it **Why:** To obtain a list of associated models
Acceptance Criteria	The selected AI processing engine belongs to the list of available ones

Requirements filled	RQ_0306
T83B002 **Select AI model**	**Priority:** Must
	Who: Product Engineer or zApp Developer **Where:** Anywhere **When:** At design-time, after T83B001 **What:** End user selects the AI model to be trained **Why:** To improve quality analysis performance, AI models need to be trained
Acceptance Criteria	The selected AI model belongs to the list of available ones
Requirements filled	RQ_0306
T83B003 **Select training dataset**	**Priority:** Must
	Who: Product Engineer or zApp Developer **Where:** Anywhere **When:** At design-time, after T83B001 **What:** End user selects the training dataset from available historical data **Why:** To train selected AI model with selected dataset
Acceptance Criteria	Data have the required format
Requirements filled	RQ_0307
T83B004 **Select training parameters**	**Priority:** Must
	Who: Product Engineer or zApp Developer **Where:** Anywhere **When:** At design-time, after T83B001 **What:** End user selects training parameters **Why:** To configure training process
Acceptance Criteria	Parameters have the required format
Requirements filled	RQ_0306
T83B005 **Train Model**	**Priority:** Must
	Who: AI Model Training Engine **Where:** Anywhere **When:** End user starts the training procedure **What:** Load selected model, dataset, configuration and start training **Why:** To improve inspection capabilities
Acceptance Criteria	A new trained model is available
Requirements filled	RQ_0306
T83B006 **Training KPI status**	**Priority:** Must
	Who: AI Model Training Engine **Where:** Anywhere **When**: During training **What:** Training status (running, target reached/unreached) and other meaningful training KPIs are available during training **Why:** To let data analyst monitor training performance
Acceptance Criteria	A set of KPIs is available
Requirements filled	RQ_0308
T83B007 **StopTraining**	**Priority:** Must
	Who: Product Engineer or zApp Developer **Where:** Anywhere **When**: Data analyst from end user interface sends stop/start command **What:** Stop training the model **Why:** To let Data Analyst stop, reconfigure, and restart training, for example when a new rule to start/stop training task is to be applied with respect to previously defined ones
Acceptance Criteria	The training process is stopped as expected
Requirements filled	RQ_0306
T83B008	**Priority:** Must

Store Trained Model	**Who:** AI manager **Where:** Anywhere **When**: Current error indicator, ie requested accuracy, is improved/reached after a model training Data Analyst can decide to store the model **What:** Stores trained model to AI Model Storage **Why:** To let Quality Inspection Engine update the model to be executed
Acceptance Criteria	The trained model is stored and available to be deployed
Requirements filled	RQ_0309

Figure 225: Quality AI Inspection: Design and Training Functions

6.4.3 Workflows

6.4.3.1 Select AI model

The following diagram explains this function necessary interactions with other components.

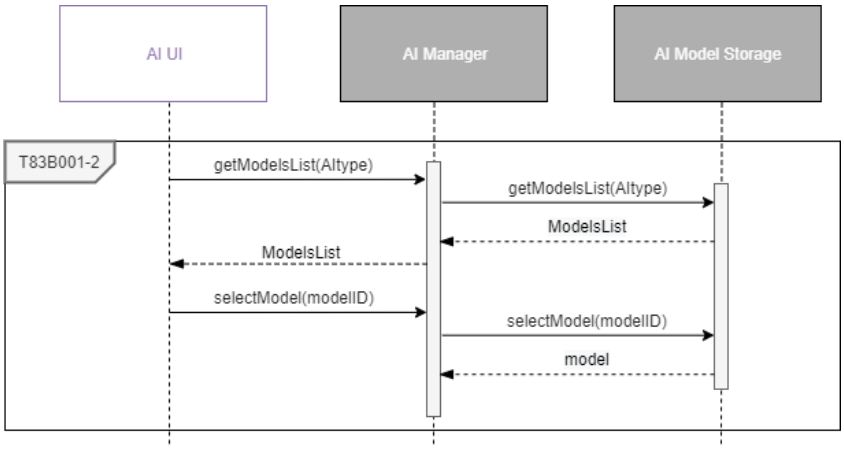


Figure 226: Select AI Model

6.4.3.2 Select training dataset

The following diagram explains this function and the necessary interactions with other components.

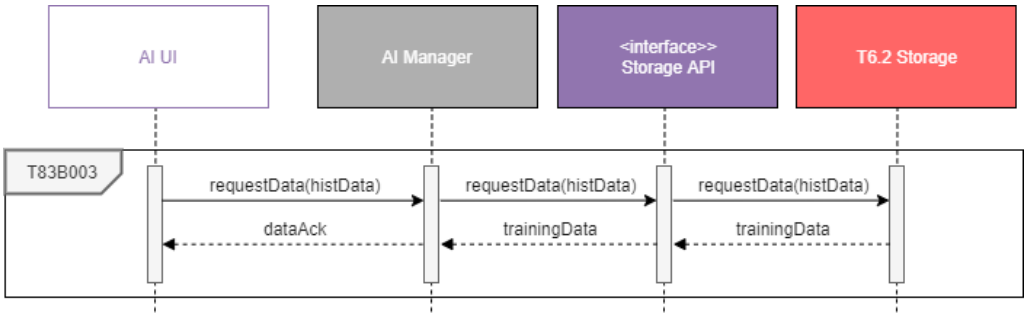


Figure 227: Select Training Dataset

6.4.3.3 Select training parameters

The following diagram explains this function and the necessary interactions with other components.

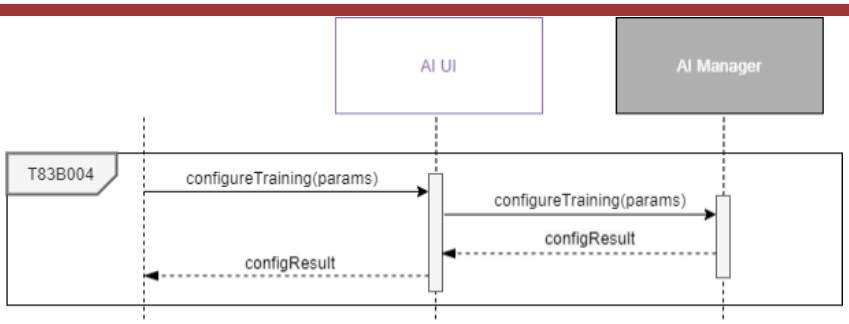


Figure 228: Select Training Parameters

6.4.3.4 Start training and training KPI status

The following diagram explains this function and the necessary interactions with other components.

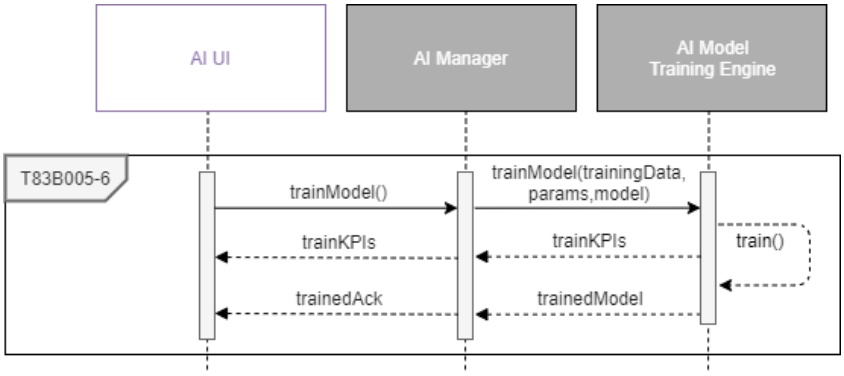


Figure 229: Start Training and Training KPI Status

6.4.3.5 Stop training

The following diagram explains this function and the necessary interactions with other components.

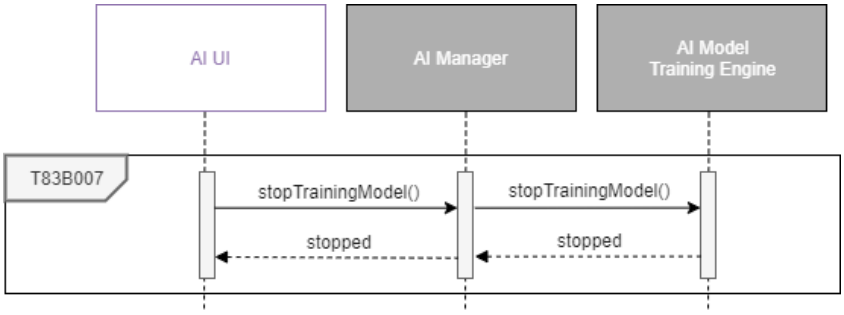


Figure 230: Stop Training

6.4.3.6 Store trained model

The following diagram explains this function and the necessary interactions with other components.

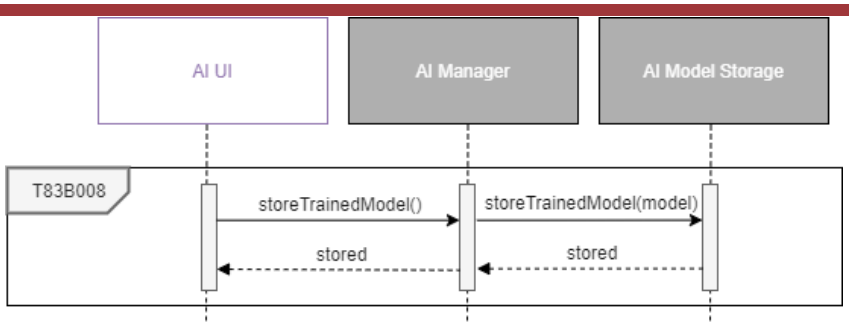


Figure 231: Store Trained Model

6.4.4 Additional Issues

The following table is about remaining issues after the description of the functional specifications.

Issue	Description	Next Steps	Lead (Rationale)
Unfilled Requirements	The following requirements with a “must“-priority were targeted at the task 8.3, but were neither filled by this module, nor by the other modules from T8.3: RQ_0298 ÷ RQ_0301, RQ_0311, RQ_0313, RQ_0690 ÷ RQ_0696. They deal with the functionality of labelling/classifying quality defects on images, which is the starting/improving condition to train supervised ML models for quality inspection on images.	VSYS oversees implementing AI quality analysis on product images and will also take care of providing proper operator interfaces to satisfy these requirements linked to T8.3	T8.3 Task lead VSYS

7 zApps

zApps are referred to as the applications that are developed on top of the platform level infrastructure and utility components (WP5-6) and the specialized components and modules for zero defects processes and products (WP7-8) and are created within WP9-10.

zApps were defined within the use cases as the applications to solve the use cases primary problems and present the stakeholder from the factory / manufacturing with a user interface to help him optimize the process or control the product defects etc. Some of the zApps were defined as separate applications but have been decided to be implemented within the same functional application and are therefore described accordingly within one section.

For the zApps, no concrete architecture is planned or defined, but still describing the functionalities of the applications and the necessary control flow is supposed to lead the development in a more concrete direction and was therefore realized for this deliverable.

7.1 zMachineMonitor & Analytics (zA2.01-zA2.02)

7.1.1 Overall functional characterization & Context

zMachineMonitor is employed to evaluate the health status of a machine tool and its components, aiding the operator to avoid hidden malfunctions. zMachineMonitor automatically gathers, stores and analyse both equipment and machining process data. The application can detect sudden or abrupt changes that can lead to a premature failure and that needs to be taken care of by the operator in a short time, generating notifications to alert the operator or the shop floor manager. The application uses temporal series of values read by the machine, which are is read periodically and stored in different data formats: binary or ASCII delivered in a format readable by a program such Json / Bson or XML. Monitoring data and corresponding machine health values are offered to the operator by means of a user interface. zMachineAnalytics analyses and estimates deviations (faults, non-conformity process parameters, etc) from data gathered by the zMachineMonitor. A dedicated UI delivers status and monitoring information to the Machine Tool User in the zMachineMonitor application, information that will be available to the Machine Tool User, Machine Tool Manufacturer, or the Machine Tool Components Manufacturer. Hence, zMachineAnalytics diagnoses and analyses potential problem(s) as will trigger appropriate mitigation actions.

7.1.2 Functions / Features

- **Data sources configuration:** Backend function to configure the data sources used to integrate master data. Master data includes product type information, and inspection test information
- **Industrial data collection configuration:** Backend function to configure the connections to exchange data with the machine. The user can define different industrial variables
- **Data preparation:** This this function consists in writing the specified configuration values into the specified industrial variables using the configured connections
- **Results analysis**: To make a detailed analysis of the results, comparing new measurements with historic data and analyse trends in available time series to detect and predict possible faults, non-conformity process parameters, etc

These functions can be further decomposed into the following subtasks that have to be realised:

Subtask	Subtask description
ZA2.01.1 Configure master data sources	**Priority:** Should
	Who: ZDMP consultant **When / Where:** During configuration (runtime), on premise **What:** User selects data sources (files, database connections) to integrate master data **Why:** To integrate master data information
Acceptance Criteria	The user can create a new master data configuration The user can create a new data source The user can select data sources that have been previously configured in the platform The user can update a master data configuration with the selected data source Master data provides access to product type information
Requirements filled	RQ_0108, RQ_0111
ZA2.01.2 Configure machine connection	**Priority:** Should
	Who: ZDMP consultant **When / Where:** During configuration (runtime), on premise **What:** User configures a connection to machine **Why:** To configure the connection to the machine and enable data exchange
Acceptance Criteria	The user can create a new unit model The user can edit the connection parameters Optionally, the user could browse the field network to search for available connections
Requirements filled	RQ_0131, RQ_0135
ZA2.01.3 Configure industrial variables	**Priority:** Should
	Who: ZDMP consultant **What:** User configures industrial variables **When / Where:** During configuration (runtime), on premise **Why:** To uniquely identify industrial variable in the machine
Acceptance Criteria	The user can create a new industrial variable The user can specify the necessary parameters to read and write data from the variable The user can edit the model to specify which variables are used to read and write machine data The user can select industrial variables that have been previously configured
Requirements filled	RQ_0131
ZA2.01.4 Edit configuration of machine	**Priority:** Must
	Who: Test Engineer **When / Where:** During configuration (runtime), on premise **What:** User selects a machine **Why:** To edit the configuration of the machine
Acceptance Criteria	The user can edit the configuration The user can select a machine from master data The user can set the machine type of the configuration The machine has a unique identifier The user can select an industrial variable to read the machine identifier For every write variable, the user can set a name For every write parameter, the user can select an industrial variable to write the parameter
Requirements filled	RQ_0138, RQ_0139
ZA2.01.5 Load configuration parameters	**Priority:** Must
	Who: Machine data collection adapter **When / Where:** Before the machine starts the process (runtime)

	What: write the values of the machine **Why:** To configure the machine
Acceptance Criteria	The Machine data collection adapter can get the variables and values to write
Requirements filled	RQ_0138, RQ_0139
ZA2.01.6 Machine operation detection	**Priority:** Must
	Who: Machine data collection adapter **When / Where:** When the machine starts production (runtime) **What:** Read the values performed in the machine **Why:** To know the machine is operating
Acceptance Criteria	The machine adapter can read the machine variables
Requirements filled	RQ_0136
ZA2.01.7 Collect data	**Priority:** Must
	Who: Machine data collection adapter **When / Where:** When the machine is in production (runtime),) **What:** Read data **Why:** To detect existing data
Acceptance Criteria	The machine adapter can read the data in the configured variables
Requirements filled	RQ_0127, RQ_0128, RQ_0129, RQ_0130, RQ_0131, RQ_0135, RQ_0136, RQ_0137
ZA2.01.8 Save data	**Priority:** Must
	Who: Application storage **When / Where:** When the machine starts the operation (runtime), in the machine **What:** Save the machine data **Why:** To provide information and insights to the test engineer
Acceptance Criteria	The machine adapter can record data The data record includes the parameters and values used The data record includes the machine type The data records include the machine unique identifier and are time-stamped
Requirements filled	RQ_0117, RQ_0118, RQ_0119, RQ_0124, RQ_0125, RQ_0126,
ZA2.01.9 Analyse results	**Priority:** Must
	Who: Operator, Machine builder **When / Where:** When the machine is in operation, in the test department **What:** Analyse the results **Why:** To check available results
Acceptance Criteria	The user(s) can set parameters to filter data and configure analytics Process quality prediction can identify trends in health of the machine and components and predict the properties of future batches Process quality prediction can publish the results
Requirements filled	RQ_0109, RQ_0110, RQ_0113, RQ_0114, RQ_0115, RQ_0116, RQ_0120, RQ_0133, RQ_0134
ZA2.01.10 Send notifications	**Priority:** Must
	Who: Machine builder **When / Where:** When the analysis predicts a possible malfunction **What:** Send an email notification **Why:** To warn involved parties
Acceptance Criteria	Monitoring and alerting can subscribe to analytic results Monitoring and alerting can publish notifications when the results match the configured rules The test engineer can check statistical properties of the machine status The test engineer can check the correlation of the parameters and machine status
Requirements filled	RQ_0135
ZA2.02.1 Analyse results	**Priority:** Must
	Who: Operator, Machine builder **When / Where:** When the machine is in operation, in the test department **What:** Analyse the results

	Why: To check available results
Acceptance Criteria	The user(s) can set parameters to filter data and configure analytics Process quality prediction can identify trends in health of the machine and components and predict the properties of future batches Process quality prediction can publish the results
Requirements filled	RQ_0140, RQ_0141, RQ_0142, RQ_0143, RQ_0144, RQ_0145, RQ_0146, RQ_0147, RQ_0148, RQ_0149, RQ_0150, RQ_0151, RQ_0152, RQ_0153, RQ_0154

Figure 232zA2.01-2.02 Features

7.1.3 Workflows

In this section is exposed the sequential diagrams that represents all interactions between components and users in each zApp. The sequential diagrams are divided by zApp and in each of them the interactions between the components and users can be visualized.

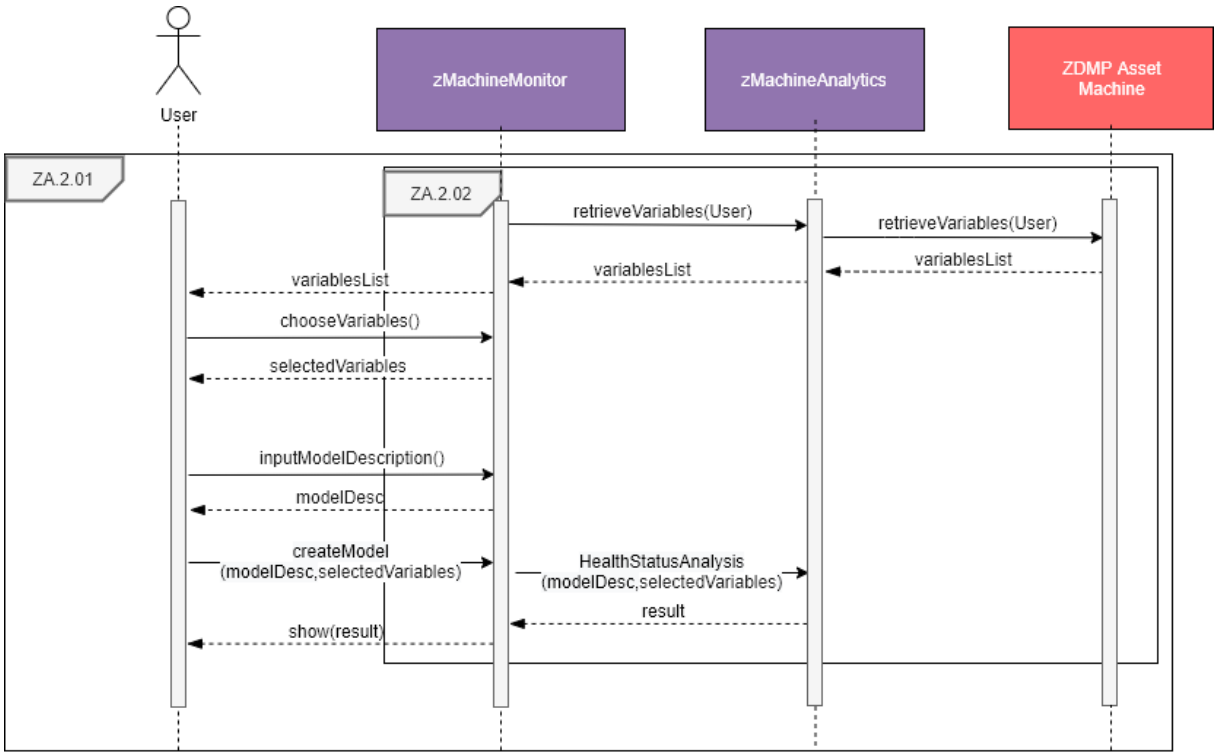


Figure 233: Workflow of zMachineMonitor & Analytics

7.2 zParameterMonitor & Analytics (zA2.03-z2.04)

7.2.1 Overall functional characterization & Context

zParameterMonitor & Analytics is used to define which are the best parameters set for a specific task based on the quality of previous results offered by a defined machine. This application registers general manufacturing conditions, machine parameters, and quality results (feedback regarding the produced part quality). Hence, whenever a similar task shall be performed under similar conditions, the zApp will propose the most similar/higher quality situation and shows the parameters used.
zParameterAnalytics performs the analysis needed to automatically detect which parameter combination provides the best results given a condition (eg type of machine, environmental variables, etc).

7.2.2 Functions / Features

- **Data sources configuration:** Backend function to configure the data sources used to integrate master data. Master data includes product type information, and inspection test information.
- **Industrial data collection configuration:** Backend function to configure the connections to exchange data with the machine. The user can define different industrial variables.
- **Data preparation:** This this function consists in writing the specified configuration values into the specified industrial variables using the configured connections.
- **Feedback:** this function will record the operator feedback on the produced part or component. This information will help the training of the algorithms.
- **Results analysis**: To make a detailed analysis of the results, comparing new measurements with historic data and analyse trends in available time series to detect and predict best set of parameters for a predefined machine operation.

These functions can be further decomposed into the following subtasks that have to be realised:

Subtask	Subtask description
ZA2.03.1 Configure master data sources	**Priority:** Should
	Who: ZDMP consultant **When / Where:** During configuration (runtime), on premise **What:** User selects data sources (files, database connections) to integrate master data **Why:** To integrate master data information
Acceptance Criteria	The user can create a new master data configuration The user can create a new data source The user can select data sources that have been previously configured in the platform The user can update a master data configuration with the selected data source Master data provides access to product type information
Requirements filled	RQ_0156, RQ0157
ZA2.03.2 Configure test unit connection	**Priority:** Should
	Who: ZDMP consultant **When / Where:** During configuration (runtime), on premise **What:** User configures a connection to machine **Why:** To configure the connection to the machine and enable data exchange
Acceptance Criteria	The user can create a new unit model The user can edit the connection parameters Optionally, the user could browse the field network to search for available connections
Requirements filled	RQ_0158
ZA2.03.3 Configure industrial variables	**Priority:** Should
	Who: ZDMP consultant **What:** User configures industrial variables **When / Where:** During configuration (runtime), on premise **Why:** To uniquely identify industrial variable in the machine
Acceptance Criteria	The user can create a new industrial variable The user can specify the necessary parameters to read and write data from the variable The user can edit the model to specify which variables are used to read and write machine data The user can select industrial variables that have been previously configured
Requirements filled	RQ_0131

ZA2.03.4 **Edit test configuration of machine**	**Priority:** Must
	Who: Test Engineer **When / Where:** During configuration (runtime), on premise **What:** User selects a machine **Why:** To edit the configuration of the machine
Acceptance Criteria	The user can edit the configuration The user can select a machine from master data The user can set the machine type of the configuration The machine has a unique identifier The user can select an industrial variable to read the machine identifier For every write variable, the user can set a name For every write parameter, the user can select an industrial variable to write the parameter
Requirements filled	RQ_0138, RQ_0139
ZA2.03.5 **Load configuration parameters**	**Priority:** Must
	Who: Machine data collection adapter **When / Where:** Before the machine starts the process (runtime) **What:** write the values of the machine **Why:** To configure the machine
Acceptance Criteria	The Machine data collection adapter can get the variables and values to write
Requirements filled	RQ_0138, RQ_0139
ZA2.03.6 **Machine operation detection**	**Priority:** Must
	Who: Machine data collection adapter **When / Where:** When the machine starts production (runtime) **What:** Read the values performed in the machine **Why:** To know the machine is operating
Acceptance Criteria	The machine adapter can read the machine variables
Requirements filled	RQ_0136
ZA2.03.7 **Collect data**	**Priority:** Must
	Who: Machine data collection adapter **When / Where:** When the machine is in production (runtime),) **What:** Read data **Why:** To detect existing data
Acceptance Criteria	The machine adapter can read the data in the configured variables
Requirements filled	RQ_0127, RQ_0128, RQ_0129, RQ_0130, RQ_0131, RQ_0135, RQ_0136, RQ_0137
ZA2.03.8 **Save data**	**Priority:** Must
	Who: Application storage **When / Where:** When the machine starts the operation (runtime), in the machine **What:** Save the machine data **Why:** To provide information and insights to the test engineer
Acceptance Criteria	The machine adapter can record data The data record includes the parameters and values used The data record includes the machine type The data records include the machine unique identifier and are time-stamped
Requirements filled	RQ_0117, RQ_0118, RQ_0119, RQ_0124, RQ_0125, RQ_0126
ZA2.03.9 **Send feedback**	**Priority:** Must
	Who: Operator, Machine builder, **When / Where:** When the machine operation is finished **What:** Complete form indicating the performance of the machine related to the offered quality **Why:** To register status
Acceptance Criteria	Monitoring and alerting can subscribe to analytic results Monitoring and alerting can publish notifications when the results match the configured rules The test engineer can check properties of the machine operation data
Requirements filled	RQ_0155

ZA2.04.1 Analyse parameters	**Priority:** Must
	Who: Operator, Machine builder **When / Where:** When the machine is in operation, in the test department **What:** Analyse the results **Why:** To check available results
Acceptance Criteria	The user(s) can set parameters to filter data and configure analytics Process quality prediction can identify trends in health of the machine and components and predict the properties of future batches Process quality prediction can publish the results
Requirements filled	RQ_0140, RQ_0141, RQ_0142, RQ_0143, RQ_0144, RQ_0145, RQ_0146, RQ_0147, RQ_0148, RQ_0149, RQ_0150, RQ_0151, RQ_0152, RQ_0153, RQ_0154

Figure 234: zA2.03-z2.04 Functions

7.2.3 Workflows

In this section is exposed the sequential diagrams that represents all interactions between components and users in each zApp. The sequential diagrams are divided by zApp and in each of them the interactions between the components and users can be visualized.

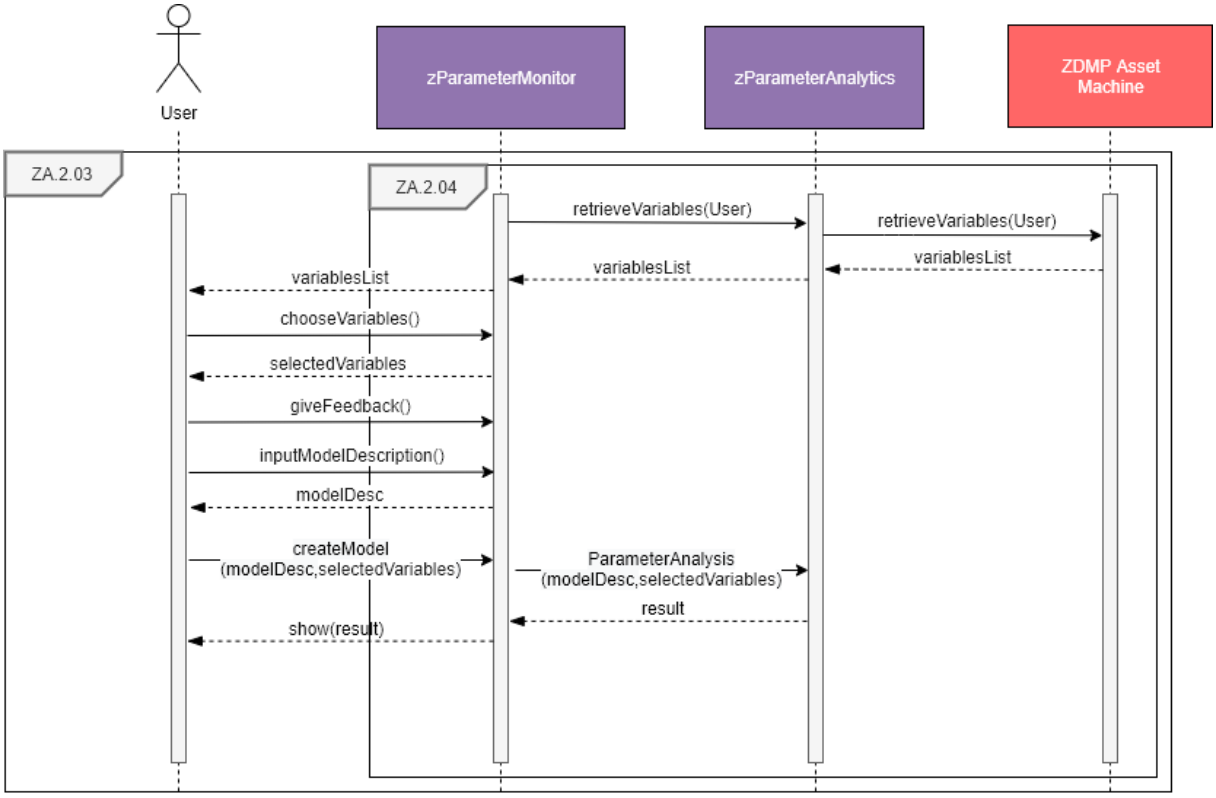


Figure 235. Workflow of zParameterMonitor & Analytics

7.3 z3DScannerDriver & z3DGenerator (zA2.05-zA2.06)

7.3.1 Overall functional characterization & Context

z3DGenerator is employed to generate a 3D file in a model format from a collection of points that is used to detect potential collisions of the machine with the produced part that may damage the surface quality of the produced part.

7.3.2 Functions / Features

- **Data sources configuration:** Backend function to configure the data sources used to integrate master data. Master data includes product type information, and inspection test information.
- **Industrial data collection configuration:** Backend function to configure the connections to exchange data with the machine. The user can define different industrial variables.
- **Data preparation:** This this function consists in writing the specified configuration values into the specified industrial variables using the configured connections.
- **Results analysis**: To make a detailed analysis of the results, comparing new measurements with historic data and analyse trends in available time series to detect and predict possible faults, non-conformity process parameters, etc

These functions can be further decomposed into the following subtasks that have to be realised:

Subtask	Subtask description
ZA2.05.1 Configure master data sources	**Priority:** Should
	Who: ZDMP consultant **When / Where:** During configuration (runtime), on premise **What:** User selects data sources (files, database connections) to integrate master data **Why:** To integrate master data information
Acceptance Criteria	The user can create a new master data configuration The user can create a new data source The user can select data sources that have been previously configured in the platform The user can update a master data configuration with the selected data source Master data provides access to product type information
Requirements filled	N/A
ZA2.05.2 Configure machine connection	**Priority:** Should
	Who: ZDMP consultant **When / Where:** During configuration (runtime), on premise **What:** User configures a connection to machine **Why:** To configure the connection to the machine and enable data exchange
Acceptance Criteria	The user can create a new unit model The user can edit the connection parameters Optionally, the user could browse the field network to search for available connections
Requirements filled	N/A
ZA2.05.3 Configure industrial variables	**Priority:** Should
	Who: ZDMP consultant **What:** User configures industrial variables **When / Where:** During configuration (runtime), on premise **Why:** To uniquely identify industrial variable in the machine
Acceptance Criteria	The user can create a new industrial variable The user can specify the necessary parameters to read and write data from the variable The user can edit the model to specify which variables are used to read and write machine data The user can select industrial variables that have been previously configured
Requirements filled	RQ_0131
ZA2.05.4	**Priority:** Must

Edit configuration of machine	**Who:** Test Engineer **When / Where:** During configuration (runtime), on premise **What:** User selects a machine **Why:** To edit the configuration of the machine
Acceptance Criteria	The user can edit the configuration The user can select a machine from master data The user can set the machine type of the configuration The machine has a unique identifier The user can select an industrial variable to read the machine identifier For every write variable, the user can set a name For every write parameter, the user can select an industrial variable to write the parameter
Requirements filled	RQ_0138, RQ_0139
ZA2.05.5 Load configuration parameters	**Priority:** Must
	Who: Machine data collection adapter **When / Where:** Before the machine starts the process (runtime) **What:** Write the values of the machine **Why:** To configure the machine
Acceptance Criteria	The Machine data collection adapter can get the variables and values to write
Requirements filled	RQ_0138, RQ_0139
ZA2.05.6 Machine operation detection	**Priority:** Must
	Who: Machine data collection adapter **When / Where:** When the machine starts production (runtime) **What:** Read the values performed in the machine **Why:** To know the machine is operating
Acceptance Criteria	The machine adapter can read the machine variables
Requirements filled	RQ_0127, RQ_0128, RQ_0129, RQ_0130, RQ_0131, RQ_0135, RQ_0136, RQ_0137
ZA2.05.7 Collect data	**Priority:** Must
	Who: Machine data collection adapter **When / Where:** When the machine is in production (runtime),) **What:** Read data **Why:** To detect existing data
Acceptance Criteria	The machine adapter can read the data in the configured variables
Requirements filled	RQ_0127, RQ_0128, RQ_0129, RQ_0130, RQ_0131, RQ_0135, RQ_0136, RQ_0137
ZA2.05.8 Save data	**Priority:** Must
	Who: Application storage **When / Where:** When the machine starts the operation (runtime), in the machine **What:** Save the machine data **Why:** To provide information and insights to the test engineer
Acceptance Criteria	The machine adapter can record data The data record includes the parameters and values used The data record includes the machine type The data records include the machine unique identifier and are time-stamped
Requirements filled	RQ_0117, RQ_0118, RQ_0119, RQ_0124, RQ_0125, RQ_0126
ZA2.05.9 Analyse results	**Priority:** Must
	Who: Machine builder **When / Where:** When the machine is in operation, in the test department **What:** Analyse the results **Why:** To check available results
Acceptance Criteria	The user(s) can set parameters to filter data and configure analytics
Requirements filled	N/A

Figure 236: zA2.05-zA2.06 Functions

7.3.3 Workflows

In this section is exposed the sequential diagrams that represents all interactions between components and users in each zApp. The sequential diagrams are divided by zApp and in each of them the interactions between the components and users can be visualized.

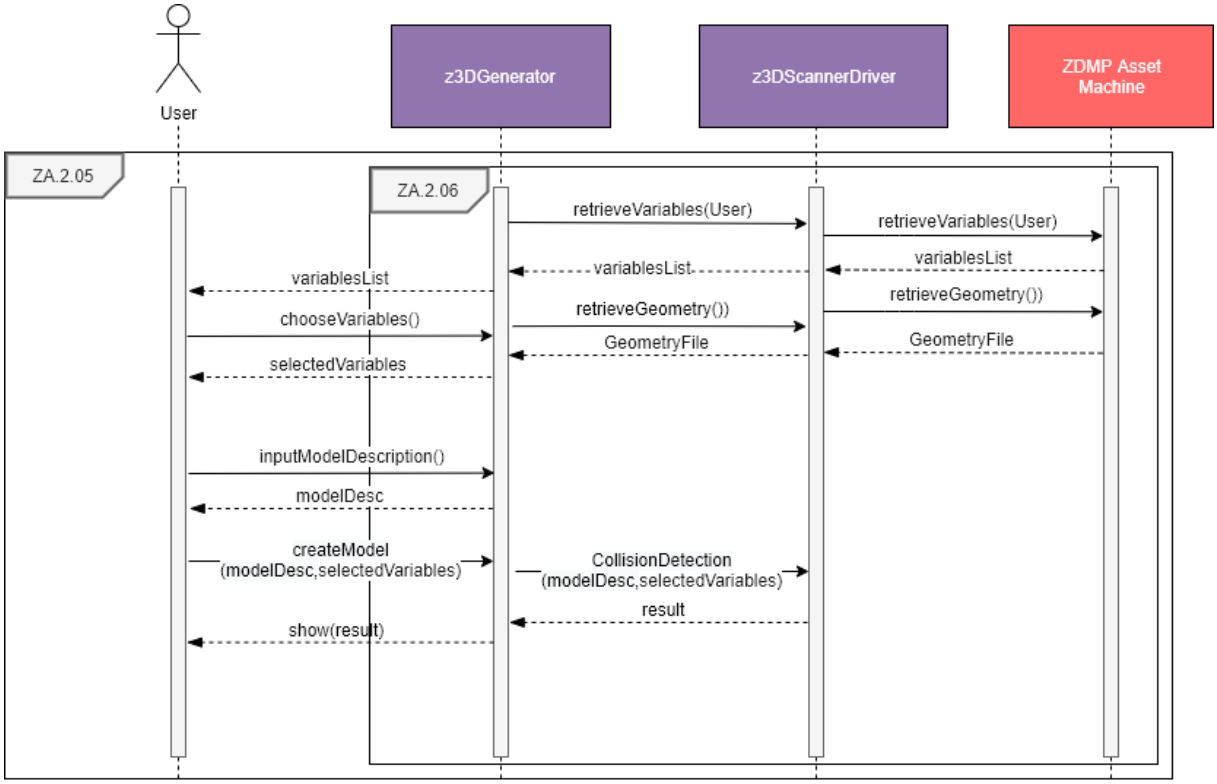


Figure 237: Workflow of z3DScannerDriver & z3DGenerator

7.4 zAnomalyDetector (zA1.01)

7.4.1 Overall functional characterization & Context

The main goal of zAnomalyDetector is to create a model for detecting anomalies and visualize the data variables contributing to such anomaly in a dashboard. This zApp uses the anomaly detector API provided by the Product Assurance Runtime component to generate a ML model for fulfilling this goal.

7.4.2 Functions / Features

The functions of the zAnomalyDetector are the following:

- **Creation of models for anomaly detection:** The Production Supervisor can interact with zAnomalyDetector to choose the data variables to monitor from the available ones. With the selected variables, a specific anomaly detector model is created. The Production Supervisor can subscribe to an existing model to receive notifications about anomalies detected on the product data by the underlying model.
- **Anomaly reporting dashboard:** the zApp includes a dashboard with user-friendly charts to show information regarding anomalous values in the selected variables.

This dashboard also provides statistical indicators about how the different variables are relevant in the context of an anomaly.

- **Anomalies notification:** When an anomaly is detected a visual notification is reported to the Production Supervisor. Notifications are logged and reported using the Monitoring and Alerting component. The notification includes which variables are contributing or "explaining" such anomaly.
- **Real-time monitoring:** This app provides information in real-time regarding the values received of the selected variables and several statistics computed by the model to detect abnormal situations.

The above functions need the implementation of the following tasks:

Subtask	Subtask description
T91ZA001 Model variable initialization	**Priority:** Must
	Who: Production Supervisor **Where:** User interface in the browser **When:** At design time when zAnomalyDetector is started **What:** Choose the set of product data variables from the one currently available for looking anomalies **Why:** To train the model, which performs the anomaly detection, according to the variables specifically required by the Production Supervisor
Acceptance Criteria	The information related to the selected variables will be available in the reporting window At least one variable is selected
Requirements filled	RQ_0028, RQ_0034
T91ZA002 Model creation	**Priority:** Must
	Who: Production Supervisor **Where:** User interface in the browser **When:** At runtime after T91ZA001 **What:** Create an anomaly detector that is continuously looking for anomalies **Why:** To detect anomalies in the product data and notify of such issues
Acceptance Criteria	A model deployed in run-time with the expected configuration
Requirements filled	RQ_0028, RQ_0034
T91ZA003 Model visualization	**Priority:** Must
	Who: Production Supervisor **Where:** User interface in the browser **When:** At design time when zAnomalyDetector is started **What:** Select a previously generated model to report for anomalies **Why:** Several anomaly detectors, with different configuration could be executed simultaneously
Acceptance Criteria	List all available anomaly detectors to the supervisor The model is retrieved, and the information of the anomaly detector dashboard is changed accordingly The information related to the selected variables will be available in the reporting window
Requirements filled	N/A
T91ZA004 Time period selection	**Priority:** Should
	Who: Production Supervisor **Where:** User interface in the browser **When:** At runtime when an anomaly detector model is already loaded in the dashboard **What:** Select a start date and an end date for filtering the data **Why:** To help the Production Supervisor in the understanding of the anomalies generated in a specific time

Acceptance Criteria	Information of the dashboard must be restricted to the specific period selected for all the graphical components
Requirements filled	N/A
T91ZA005 Anomaly visualization with Statistical control charts	**Priority:** Must
	Who: Production Supervisor **Where:** User interface in the browser **When:** At runtime as new anomalies are retrieved **What:** A control chart showing statistical indicators, such as T^2 or Squared Prediction Error, to check whether the current values of the product data variables (historical or predicted) are in a normal scenario or not **Why:** To help in the supervision tasks regarding the status of the production data
Acceptance Criteria	Charts must show in real-time the statistical indicator calculation provided by the model If no data is received, the error should be notified to the user
Requirements filled	RQ_0013
T91ZA006 Variable contribution plot	**Priority:** Must
	Who: Production Supervisor **Where:** User interface in the browser **When:** At runtime when an anomaly is selected **What:** A bar chart informing the contribution/weighing of the variables in the anomaly. By default, the contribution plot shows the last anomaly, but the Production Supervisor can select another variable in the control chart to inspect its contribution plot. **Why:** To inform the Production Supervisor which variables are responsible of the status of the production data
Acceptance Criteria	The contribution of all the variables selected by the Production Supervisor must be shown
Requirements filled	RQ_0021
T91ZA007 Anomalies History	**Priority:** Should
	Who: Production Supervisor **Where:** User interface in the browser **When:** At runtime **What:** A list of the last notifications generated by the current loaded anomaly detector **Why:** To inform the Production Supervisor about anomalies generated when the application was running
Acceptance Criteria	The last ten notifications must be recorded and must be accessible from the dashboard
Requirements filled	N/A
T91ZA008 Time Series plot	**Priority:** Should
	Who: Production Supervisor **Where:** User interface in the browser **When:** At runtime **What:** A chart showing the values received of a variable over time **Why:** To report visually the temporal behaviour of a variable
Acceptance Criteria	A time series plot could be generated for any of the variables selected by the Production Supervisor
Requirements filled	N/A

Figure 238: zA1.01 Functions

7.4.3 Workflows

7.4.3.1 Model creation

The createModel() subtask is explained in Supervision Model API functional specifications.

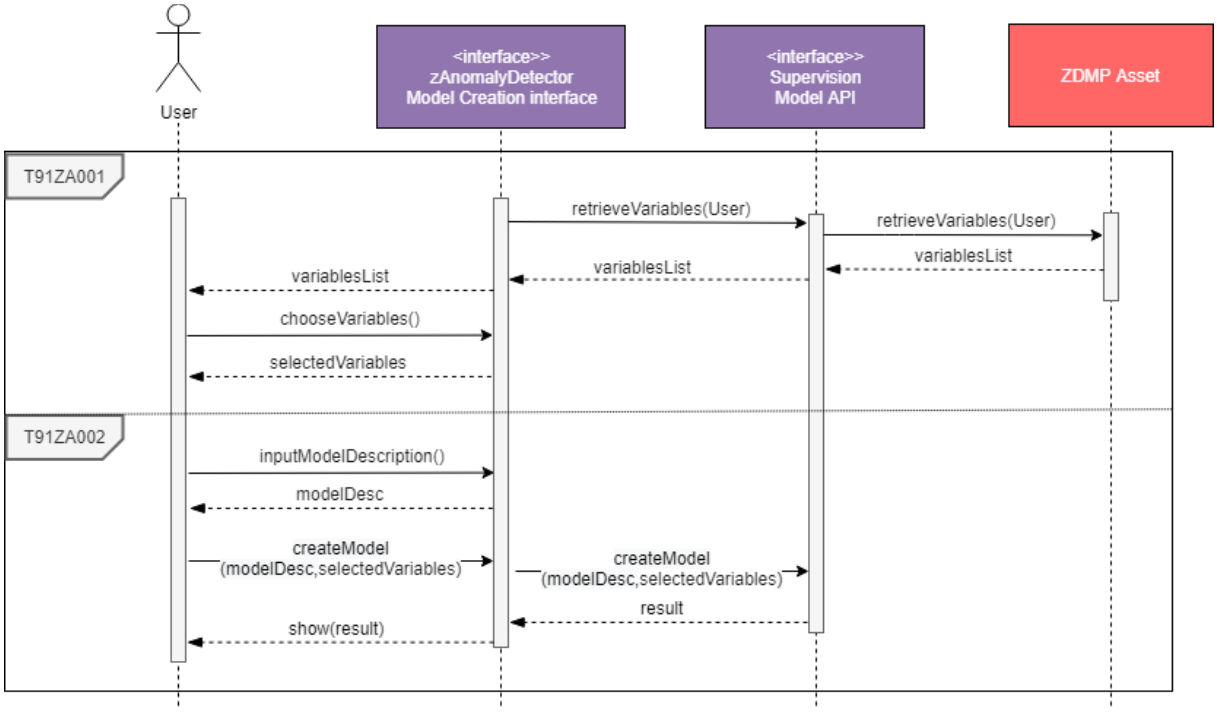


Figure 239: Model Creation Sequence Diagram

7.4.3.2 Model visualization

The following diagram explains this function and the necessary interactions with other components.

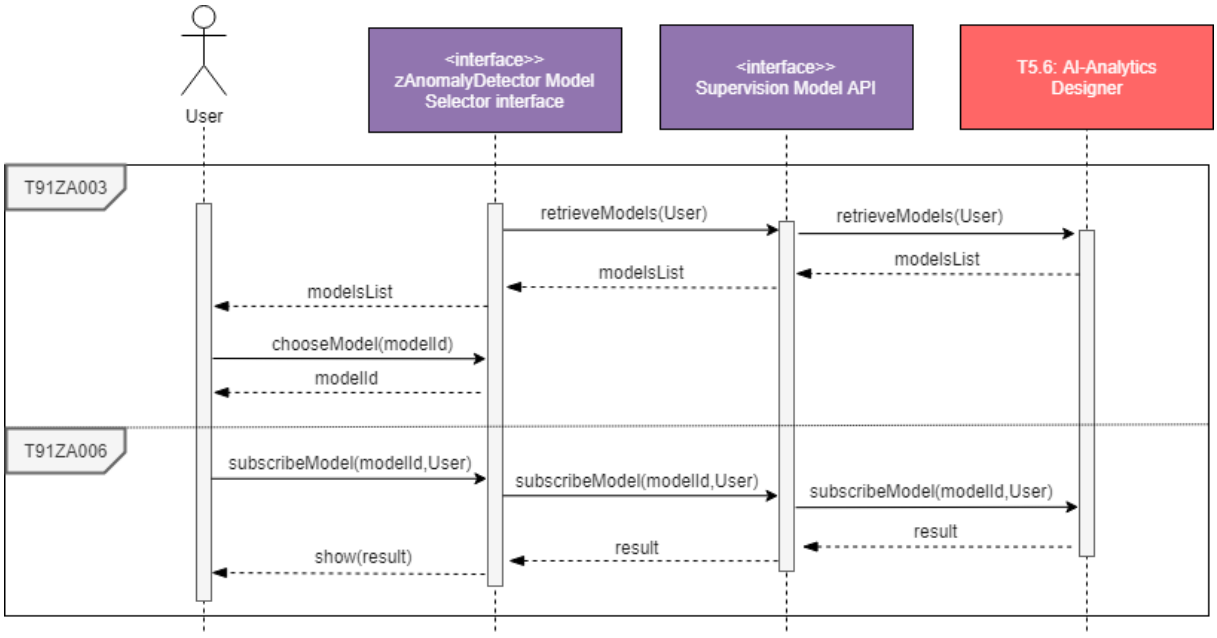


Figure 240: Model Visualization Sequence Diagram

7.4.3.3 Time period selection

For this function, predictions include the following:

- Statistical indicators, such as T2 or Squared Prediction Error, to update the corresponding control charts (subtask T91ZA005)
- Variable contribution plot (subtask T91ZA007)

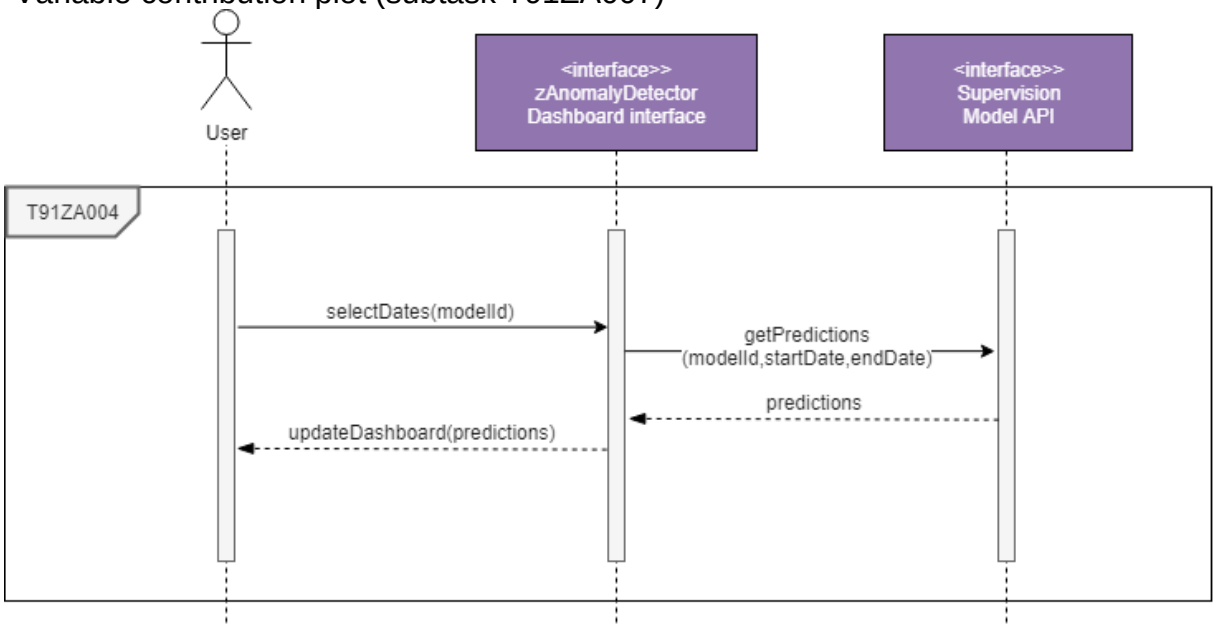


Figure 241: Time Period Selection Sequence Diagram

7.4.3.3.1 Notification history

The following diagram explains this function and the necessary interactions with other components.

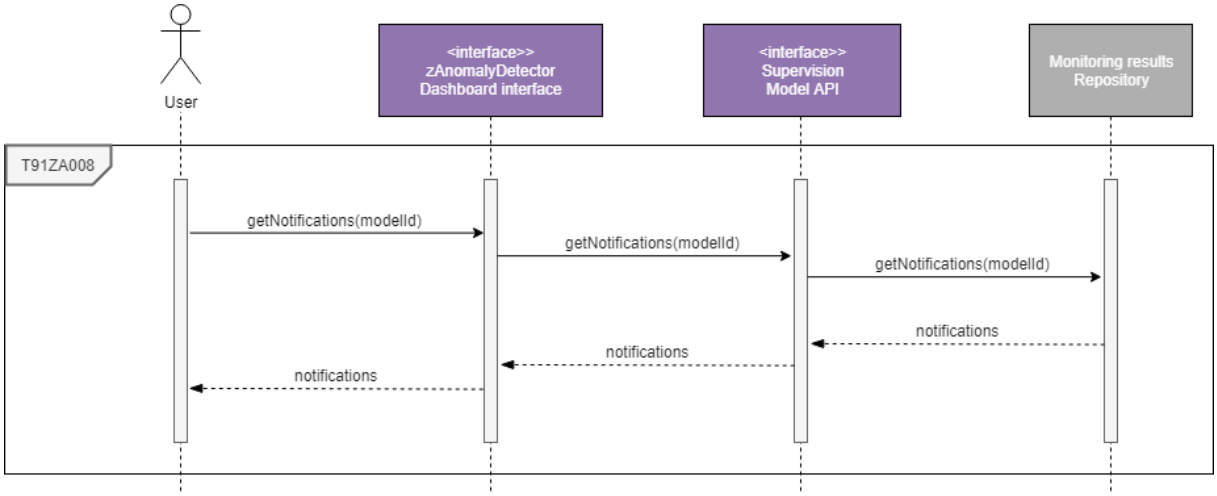


Figure 242: Notification History Sequence Diagram

7.4.3.3.2 Time series plot

The following diagram explains this function and the necessary interactions with other components.

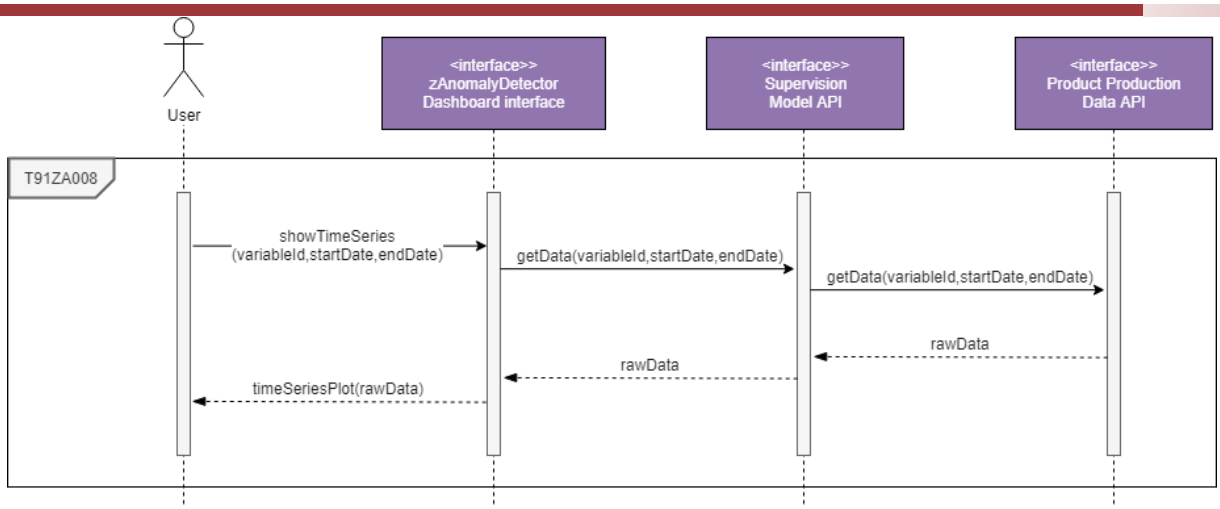


Figure 243: Time Series Plot Sequence Diagram

7.5 zDigitalTwin (z1.02)

7.5.1 Overall functional characterization & Context

The main goal of the zDigitalTwin app is to process real-time data from a manufacturing process and report a simulation of the observed system. One application in the context of ZDMP is to simulate objective variables, ie manufacturing data related with quality or performance indicators, and then to produce a prediction about how the manufacturing system will behave. To achieve this goal this zApp creates a model using the AI components provided by the ZDMP platform. As new data is processed, this model generates predictions regarding the values of the objective variables selected by the end-user, and, then a web-based dashboard reports such predictions using user-friendly charts. Additionally, this zApp includes the functionality to optimize process variables (or parameters), ie variables from the manufacturing process not explicitly related to an indicator, but with a clear influence. An optimization produces recommendations regarding such process variable to maximize (improve quality) or minimize (reduce faulting parts) a certain objective established by the Quality inspector. For evaluating the improvement of using the optimized process variables, the end-user could use them in real-time to generate a simulation on how the process will behave.

7.5.2 Functions / Features

The main functions of the zDigitalTwin are the following:

- **Digital Twin Model creation:** The Quality Inspector can interact with the zDigitalTwin interface to choose the set of objective variables to be simulated, from the set of process data received. Additionally, the end-user can select the process variables, variables whose values will be used to predict the objective variables. From the set of selected variables, a digital twin model is trained and deployed using the ZDMP AI components.
- **Predictions subscription:** zDigitalTwin is subscribed to an existing digital twin model to receive the values of the objective variables in real-time
- **Reporting Dashboard:** the zDigitalTwin includes a web-based interface to show the predicted and historical values for the objective variables received from the digital twin model. Using this dashboard, the Quality inspector can define the period for visualizing historical data and configuring the process variables.
- **Optimization: On**ce a digital twin model is trained it is possible to search for the best values of the process variables that maximize or minimizes, one or more objective variables.

These functions need the implementation of the following tasks:

Subtask	Subtask description
T91ZD001 Digital Twin model creation	**Priority:** Must
	Who: Quality inspector **Where:** In the User Interface of the zDigitalTwin **When:** At design time when the app is started **What:** To select variables from the manufacturing data which define a set of objective variables and a set of process variables, and then, create a digital twin model **Why:** The digital twin model must be trained to predict objective variables considering process variables

Acceptance Criteria	Predictions related to the selected variables will be available in the reporting dashboard
Requirements filled	RQ_0045, RQ_0049
T91ZD002 Model subscription	**Priority:** Must
	Who: Quality inspector **Where:** In the User Interface of the zDigitalTwin **When:** At runtime after T91ZD001 **What:** Subscribe to an existing digital twin model to get information of the predicted objective variables **Why:** As there could be several digital twin models available, the dashboard only reports information of the model selected by the Quality inspector
	Acceptance Criteria
	List all models currently available A confirmation is sent to the Quality inspector whether the model is sending predictions to the zApp
Acceptance Criteria	Predictions related to the selected variables will be available in the reporting dashboard
Requirements filled	RQ_0103
T91ZD003 Reporting dashboard	**Priority:** Must
	Who: Quality inspector **Where:** In the User Interface of the zDigitalTwin **When:** At runtime after T91ZD002 **What:** Show the predicted and historical values for the objective variables or process variables using a line chart (in a time series fashion) that its updated in real time **Why:** To report the prediction generated by the digital twin model and provide a visual hint of the accuracy or error of the simulation
Acceptance Criteria	Predictions related to the selected variables will be available in the reporting dashboard Predictions and historical data must be update in real-time as new predictions are received from the subscribed model
Requirements filled	N/A
T91ZD004 Variable chart selection	**Priority:** Should
	Who: Quality inspector **Where:** In the User Interface of the zDigitalTwin **When:** At runtime after T91ZD002 **What:** Hide or unhide a time-series chart of a specific objective or process variable **Why:** As a digital twin model could involve hundreds of variables, to avoid creating many charts in the dashboard and improve usability
Acceptance Criteria	Predictions related to the selected variables will be available in the reporting dashboard Initially, only the objective quality variables will be show and the rest of the variables will be presented as the Quality inspector selects them
Requirements filled	RQ_0045
T91ZD005 Process variables selection	**Priority:** Must
	Who: Quality inspector **Where:** In the User Interface of the zDigitalTwin **When:** In design time when the digital twin model to train is specified **What:** Define a constant value for a process variable that would be used by the model for generating the predictions **Why:** To change in real time process conditions and check the expected behaviour of the simulation
Acceptance Criteria	A set of initial values will be established by the digital twin model Variables values must be consistent in terms of data type and ranges
Requirements filled	RQ_0045

T91ZD006 Dashboard time period selection	**Priority:** Should
	Who: Quality inspector **Where:** In the User Interface of the zDigitalTwin **When:** At runtime after T91ZD002 **What:** Change the period of the information shown in the dashboard **Why:** To check how the models have performed in the past or reduce the amount of information show in the dashboard
Acceptance Criteria	Predictions related to the selected variables will be available in the reporting dashboard The period filtering is applied simultaneously to every chart shown in the dashboard By default, the period is the current working day
Requirements filled	N/A
T91ZD007 Optimization Configuration	**Priority:** Should
	Who: Quality inspector **Where:** In the User Interface of the zDigitalTwin **When:** At design time after a model is created using T91ZD001 **What:** Configure the optimization of a digital twin model to minimize or maximize the value of an objective variable **Why:** To find the best set of values of the involved process variables to achieve a minimization or maximization of an objective variable
Acceptance Criteria	Predictions related to the selected variables will be available in the reporting dashboard Digital Twin models and variables previously defined by the Quality inspector must be available for optimization purposes
Requirements filled	N/A
T91ZD008 Optimization Constraints	**Priority:** Should
	Who: Quality inspector **Where:** In the User Interface of the zDigitalTwin **When:** At design time after T91ZD007 **What:** Configure the optimization process defining a value range for each of the process variables involved **Why:** To avoid that the optimization process generates as result values not feasible in the manufacturing environment
Acceptance Criteria	Constraints can have minimum and maximum values, so it must be checked that minimum is lesser than maximum. Null values are allowed, indicating that there is no minimum (or maximum) Predictions related to the selected variables will be available in the reporting dashboard
Requirements filled	N/A
T91ZD009 Optimization Reporting	**Priority:** Should
	Who: Quality inspector **Where:** In the User Interface of the zDigitalTwin **When:** At runtime after T91ZD008 **What:** Show the result of the optimization process, ie, the best set of values according to the quality variable to optimize. **Why:** To report the variable values that maximize (or minimizes, depending on the type of optimization) one or more quality variables.
Acceptance Criteria	If a feasible solution is found, then show the values of each of the variables. If there is no feasible solution, then show an informative message Predictions related to the selected variables will be available in the reporting dashboard
Requirements filled	N/A
T91ZD010	**Priority:** Should

Optimization Loading	**Who:** Quality inspector **Where:** In the User Interface of the zDigitalTwin **When:** At runtime after T91ZD007 **What:** Load the result of the optimization process, ie the values obtained in the process variable configuration (if this functionality is available) **Why:** To evaluate the optimization in the dashboard using the real-time data
Acceptance Criteria	All the optimized variable values must be loaded in the process variable configuration Predictions related to the selected variables will be available in the reporting dashboard
Requirements filled	RQ_0103

Figure 244: z1.02 Functions

7.5.3 Workflows

7.5.3.1 Digital Twin Model creation

The createModel() subtask is explained in Product Quality Model API functional specifications.

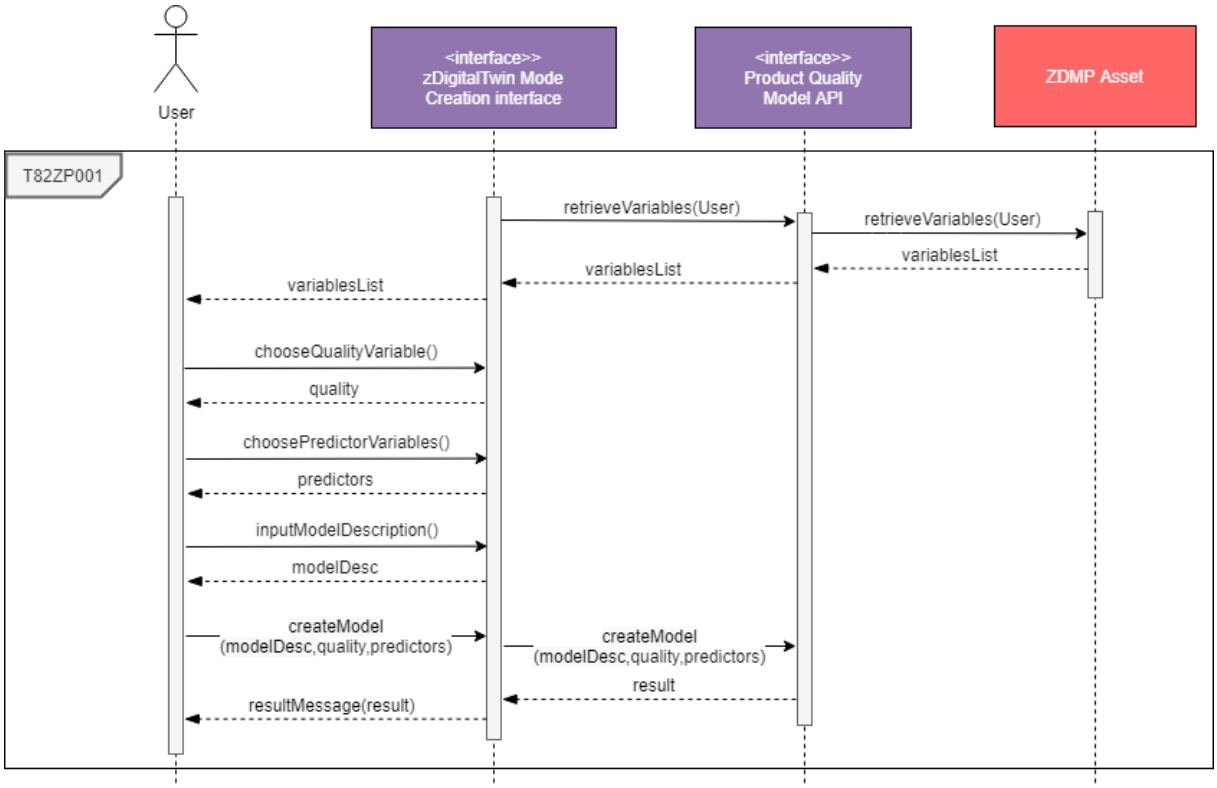


Figure 245: Digital Twin Model creation Sequence Diagram

7.5.3.2 Model subscription

The following diagram explains this function and the necessary interactions with other components.

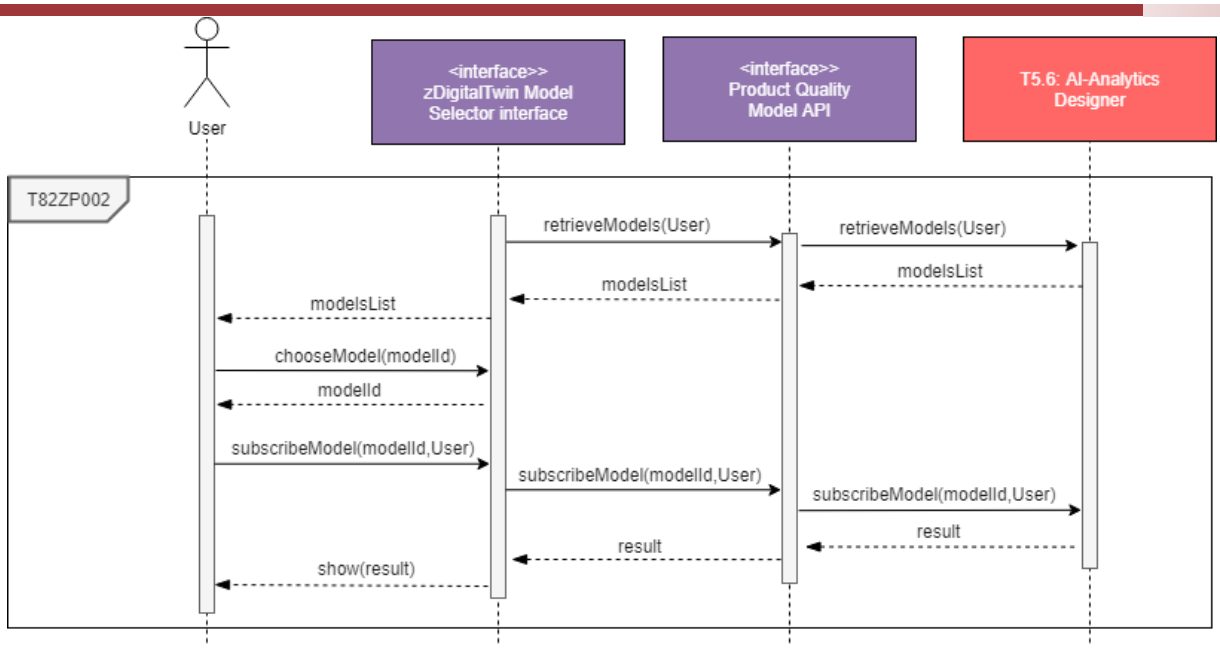


Figure 246: Model Subscription Sequence Diagram

7.5.3.3 Results dashboard

The dashboard shows a chart of the predicted and actual values for the quality variables of the model selected in a combo box. The combo shows the list of models the Quality inspector is subscripted to. The chart also shows the last historically predicted and actual values from a determined date range. It also shows the error of those predictions.

This subtask is related to the “T82E008 – Predictions query” subtask of “Quality Predictor” component.

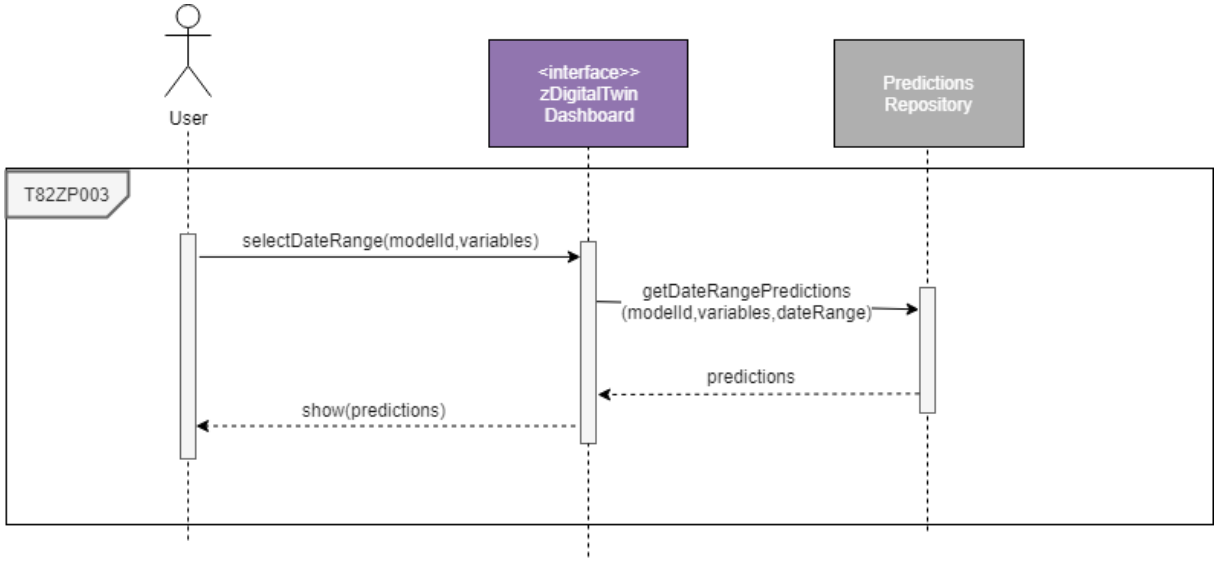


Figure 247: Update Results Dashboard Sequence Diagram

7.5.3.4 Results dashboard date range selection

The dashboard has a date range selector to change the date range shown in the chart.

7.5.3.5 Optimization (configuration)

See deliverable D4.2.1 from task “T4.2 User Mock-ups”. In the optimization configuration interface, the user must select a model, a variable/s to optimize (specifying the type of optimization: maximization/minimization) and the constraints for the optimization.

7.5.3.6 Optimization

Once the optimization is configured, the user can start the optimization process. During optimization, the values of the decision variables (predictors) are changed following some optimization criteria (that depends on the optimization technique used by the app). Those values must be inside the range specified by the constraints. Then, this new data is sent to the “Product Quality Prediction” components to get the predictions, that is, the value of the quality variable/s to optimize. The “check optimization” step compares the last optimized values with the new ones to check whether the new values are optimum than the last ones, until a stop criterion is met. When the optimization process is finished, the following results are shown to the user:

- If the optimization has converged, then the interface shows:
- The value of the optimized variable/s
- The value of the decision variables (predictors) at the optimization point
- If the optimization has not converged this message must be shown: “No feasible solution found”

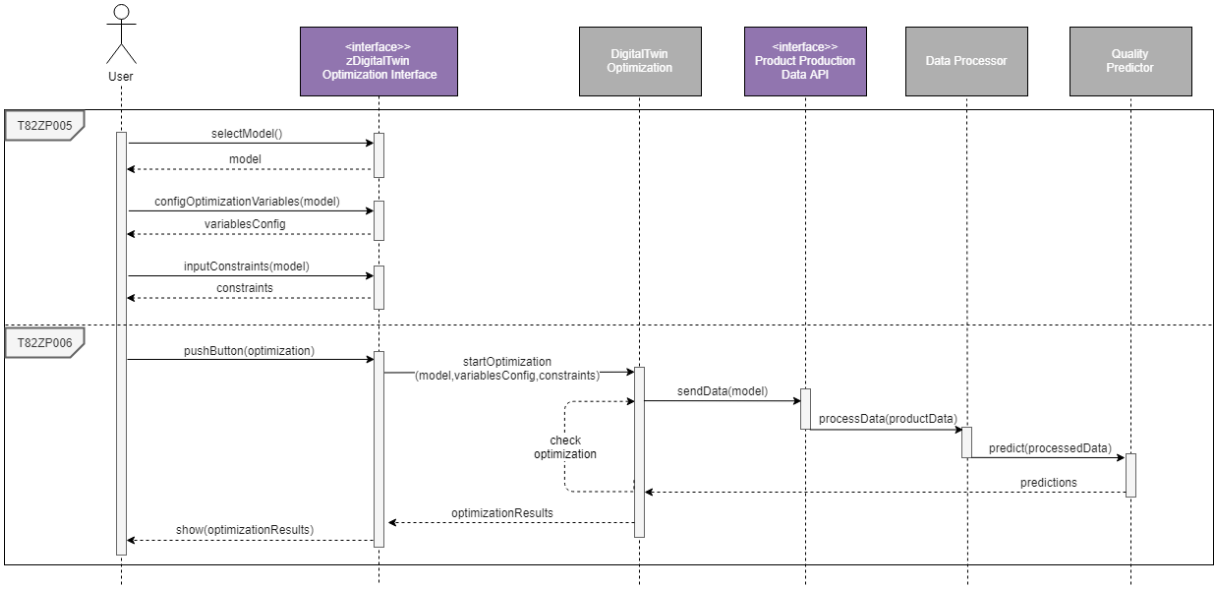


Figure 248: Optimization Sequence Diagram

7.6 zAlarm (zA1.03)

7.6.1 Overall functional characterization & Context

The objective of the zAlarm application is to provide important event information in real time, obtained as input from the zAnomalyDetector application (or in the T5.4 Monitoring & Alerting in case it is needed), to users through specialized wearables. User may use the wearables interface (buttons) to send replies, like such as acknowledging, accepting, or declining tasks related to the events/alarms.

These wearables need to include sub GHz RF communications interfaces, such as LoRA or other available technologies, to by-pass floor plant RF bands blacklisting and company regulations. The technology and development chosen may be used in other devices apart from wearables, such as PLCs, based on use case requirements, with a subset or restricted functionalities (display or user interaction).

7.6.2 Functions / Features

The functions and features of the zAlarm application are the following:

- **Receive Alarm**: The zAlarm application enables special zAssets, such as wearables, to receive alarm events with associated tasks
- **Accept alarm task**: The device that receives the alarm event can acknowledge the reception of the message and accept the associated task.
- **Decline Alarm task**: The device that receives the alarm event can acknowledge the reception of the message but decline or reject the associated task, therefore triggering the zAlarm to send the event to another candidate destination.
- **Automatic expiration and resend of alarm**: given the need to reduce downtime and speed-up the response to alarms, if the destination of the alarm and associated task does not respond (either accepting or declining), zAlarm must react cancelling the alarms with pending response within a given time, and reopen (resend) the alarm to another potential candidate. It must also inform the previous recipient that the task has been cancelled, to avoid duplicity of tasks.

Subtask	Subtask description
ZA001 Distribution of Alarm events	**Priority:** Must
	Who: zAlarm app in runtime platform **When**: On detection of alarm situation or maintenance event **Where**: Anywhere **What:** Send alarm with associated task details to a user/device **Why:** Improve response time to alarms and events
Acceptance Criteria	A wearable physical device (or any other ZDMP asset) has received an alarm addressed to it.
Requirements fulfilled	RQ_0035, RQ_0036, RQ_0037, RQ_0038, RQ_0039, RQ_0047, RQ_0089, RQ_0105, RQ_0106, RQ_0107, RQ_0108, RQ_0109, RQ_0110, RQ_0145, RQ_0147, RQ_0148, RQ_0149, RQ_0150
ZA002 Acceptance of alarm task event	**Priority:** Must
	Who: Physical devices, wearables **When:** Availability of user to respond to the alarm is positive **Where**: Anywhere **What:** Send positive acknowledgement with associated task details to a the zAlarm app in runtime platform **Why:** Improve response time to alarms and events
Acceptance Criteria	zAlarm app receives a confirmation of acceptance of alarm task
Requirements fulfilled	N/A

ZA003 **Task Acceptance management**	**Priority:** Must
	Who: zAlarm app **When:** On receiving acceptance response from one recipient for a given task **Where**: Runtime platform, anywhere **What:** Mark task as assigned and inform all other recipients to stand by. **Why:** Improve response time to alarms and events
Acceptance Criteria	All other recipients (if any) receive a task cancellation message, and zAlarm has marked task as assigned.
Requirements fulfilled	N/A
ZA004 **Rejection of alarm task event**	**Priority:** Must
	Who: Physical devices, wearables **When:** Availability of user to respond to the alarm is positive **Where**: Anywhere **What:** Send negative acknowledgement with associated task details to a the zAlarm app in runtime platform **Why:** Improve response time to alarms and events
Acceptance Criteria	zAlarm app receives a confirmation of rejection of alarm task
Requirements fulfilled	N/A
ZA005 **Task Rejection management**	**Priority:** Must
	Who: zAlarm app in runtime platform **When:** On receiving rejection response from all recipients for a given task **Where**: Runtime platform, anywhere **What:** Request details of other available users' candidates for receiving the task and resend alarm **Why:** Improve response time to alarms and events
Acceptance Criteria	The zAlarm app has detected that rejection messages have been received from all (may be one or many) users for a given task and triggers a new distribution of alarm event.
Requirements fulfilled	RQ_0035, RQ_0036, RQ_0037, RQ_0038, RQ_0039, RQ_0047, RQ_0089, RQ_0105, RQ_0106, RQ_0107, RQ_0108, RQ_0109, RQ_0110, RQ_0145, RQ_0147, RQ_0148, RQ_0149, RQ_0150
ZA006 **Alarm event timeout**	**Priority:** Must
	Who: zAlarm app in runtime platform **When:** On expired timer after sending an alarm event to a wearable **Where**: Runtime platform, anywhere **What:** Request details of other available users' candidates for receiving the task and resend alarm **Why:** Improve response time to alarms and events
Acceptance Criteria	The zAlarm app has detected that no confirmation (positive or negative) has been received within the configured response time window and triggers a new distribution of alarm event.
Requirements fulfilled	RQ_0035, RQ_0036, RQ_0037, RQ_0038, RQ_0039, RQ_0047, RQ_0089, RQ_0105, RQ_0106, RQ_0107, RQ_0108, RQ_0109, RQ_0110, RQ_0145, RQ_0147, RQ_0148, RQ_0149, RQ_0150

Figure 249: zA1.03 Functions

7.6.3 Workflows

7.6.3.1 Distribution of Alarm events and acceptance task organization

The zAlarm provides ZDMP Assets with the possibility to receive alarm events with associated task details. These events and task are generated in the ZDMP zAnomalyDetector application, and then sent to physical devices such as wearables to ensure users wandering around in the floor plant receive the alarms. zAlarm application assumes an alarm with associated task details and a chosen recipient is provided. The recommendation is to send such event to only one or reduced number or recipients, to avoid collisions and duplicity of tasks assigned. Devices have the possibility to accept or

decline tasks. Task rejection or failing to reply in time triggers the zAlarm to request another candidate recipient and resending the event.

The main steps / functionalities are:

- The zAlarm app receives an alarm event from the zAnomalyDetector and relays it to the corresponding recipient (or group), a user with a wearable, in the floor plant.
- The user decides to whether to accept or decline the responding to the alarm event, by sending a reply to the zAlarm.

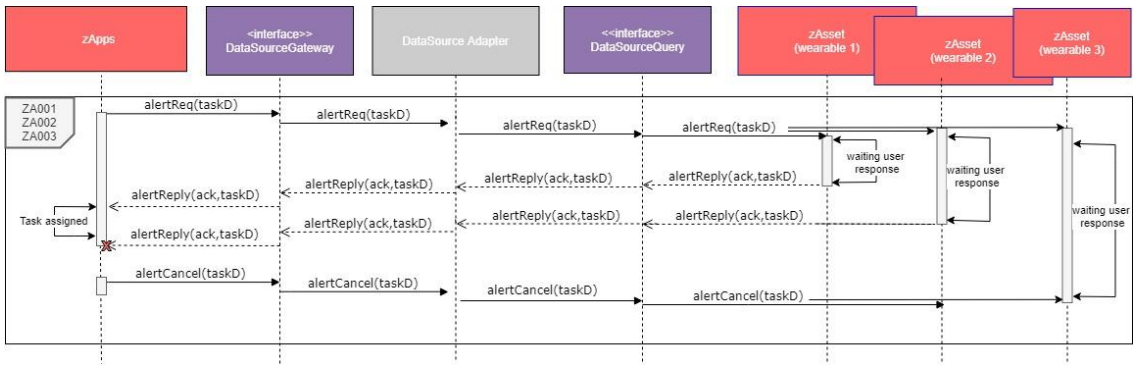


Figure 250: zAlarm Default Workflow

- If the alarm was sent only to one user and the task is accepted, it is marked as assigned. If it was sent to a group of users, zAlarm waits for the first user that accepts to assign the task and inform other users that this alarm is already addressed.

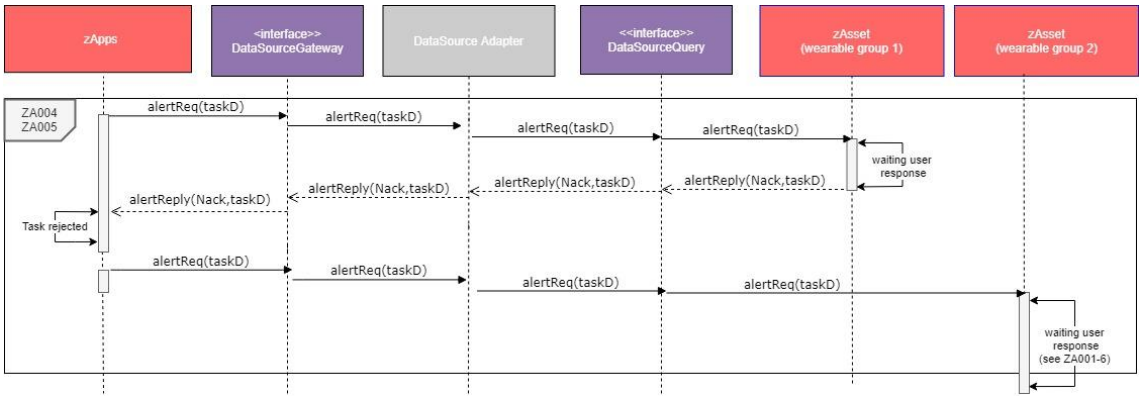


Figure 251: zAlarm Alternative Workflow

- If the alarm was sent only to one user and the task is rejected, zAlarm resends the alarm to another recipient. If it was sent to a group of users, zAlarm waits until receiving rejection messages from all of them or timeout occurs, before resending the alarm to different recipients.
- If all the users fail to reply after a configured time, zAlarm selects another user or group, and sends the alarm task again.

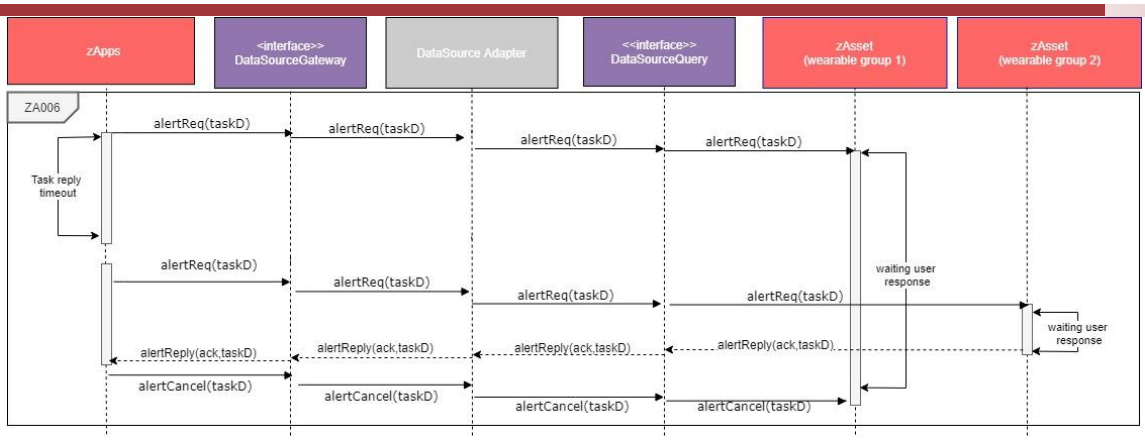


Figure 252: zAalarm Workflow Alternative User Response

7.7 Steel Tubes: Production Monitor (zA4.01-zA4.04)

7.7.1 Overall functional characterization & Context

The demand for a new solution to improve the manufacturing of steel tubes led to the creation of zApps. With the purpose to reduce and minimize the cost, and improve the steel tube production, the following zApps are created:

- zA4.01: zSteelSheetWidthMonitor: To automatically detect the width of the steel sheet to detect if the width of the sheet varies over time
- zA4.02: zHorizontalWeldDetection: To automatically detect the horizontal weld of the steel sheet; this welding is made to connect the different steel coils to each other for production to continue uninterrupted
- zA4.03: zVerticalWeldMonitor: To automatically detect the quality of the vertical weld of the steel sheet. In a situation where the vertical welding has a defect, it causes the final product to be defective
- zA4.04: zShapeTubeMonitor: To automatically detect the conformity of the tube shape

7.7.2 Functions / Features

Subtask	Subtask description
UC41A001 **Set production details**	**Priority:** Must
	Who: User **When:** In run-time **Where:** Cloud **What:** The User sets the details about what is about to be produced **Why:** The system must know how to aggregate the data that is being collected into specific production relationships
Acceptance Criteria	The User sets the details about what is about to be produced
Requirements filled	N/A
UC41A002	**Priority:** Could

Import production details	**Who:** User **When:** In run-time **Where:** Cloud **What:** The User imports a custom file to set the details about what is about to be produced **Why:** When the information already exists in the enterprise, the ability to import it will improve usability
Acceptance Criteria	The User imports a custom file to set the details about what is about to be produced
Requirements filled	N/A
UC41A003 **Save production details**	**Priority:** Should
	Who: User **When:** In run-time **Where:** Cloud **What:** After the User sets the production details, they are saved into storage **Why:** To enable historical data and to provide means for state recovery
Acceptance Criteria	After the User sets the production details, they are saved into storage
Requirements filled	N/A
UC41A004 **Load production details preset**	**Priority:** Should
	Who: User **When:** In run-time **Where:** Cloud **What:** The User browses and sets the details about what is about to be produced from a list of previously stored presets **Why:** When the information already exists in Storage, the ability to load it will improve usability
Acceptance Criteria	The User browses and sets the details about what is about to be produced from a list of previously stored presets
Requirements filled	N/A
UC41A005 **Multilingual support**	**Priority:** Must
	Who: User **When:** In run-time **Where:** Cloud **What:** The ability to switch between languages for the user interface **Why:** The user interface must accommodate local dialects
Acceptance Criteria	The ability to switch between languages for the user interface
Requirements filled	N/A
UC41A006 **Ability to select a zApp from a centralized view**	**Priority:** Should
	Who: User **When:** In run-time **Where:** Cloud **What:** The user can select available zApps for his role **Why:** The user interface should provide a straightforward way to display and access all features
Acceptance Criteria	The user can select available zApps for his role
Requirements filled	N/A
UC41A007 **Role based access control**	**Priority:** Must
	Who: User **When:** In run-time **Where:** Cloud **What:** The platform controls the user access to zApps according to registered users and user roles (Administrator, Project Manager, Operator) **Why:** The platform must be able to provide or deny access to zApps according to user and roles

Acceptance Criteria	The platform controls the user access to zApps according to registered users and user roles (Administrator, Project Manager, Operator)
Requirements filled	N/A
UC41A008 **Manage access control**	**Priority:** Must
	Who: User **When:** In run-time **Where:** Cloud **What:** The User (Administrator) can create, delete, or update users, including attributing roles (Administrator, Project Manager, Operator) **Why:** The zApp or functional parts of the zApp must be restricted to specific users and according to a user role. The role of Administrator should be able to modify the dataset of allowed users
Acceptance Criteria	The User (Administrator) can create, delete, or update users, including attributing roles (Administrator, Project Manager, Operator)
Requirements filled	N/A
UC41A009 **Set production parameters**	**Priority:** Must
	Who: User **When:** In run-time **Where:** Cloud **What:** The User sets the detection parameters for the sensor, adequate to current production **Why:** Depending on the desired end product, the parameters for non-conformant material differs. The user needs to adjust accordingly
Acceptance Criteria	The User sets the detection parameters for the sensor, adequate to current production
Requirements filled	N/A
UC41A010 **Import production parameters**	**Priority:** Could
	Who: User **When:** In run-time **Where:** Cloud **What:** The User imports a custom file to set the detection parameters in respect to what is about to be produced **Why:** When the information already exists in the enterprise, the ability to import it will improve usability
Acceptance Criteria	The User imports a custom file to set the detection parameters in respect to what is about to be produced
Requirements filled	N/A
UC41A011 **Save production parameters**	**Priority:** Should
	Who: User **When:** In run-time **Where:** Cloud, in the user interface **What:** After the user sets the detection parameters for the sensor, they are saved into storage **Why:** To enable historical data and to provide means for state recovery
Acceptance Criteria	After the user sets the detection parameters for the sensor, they are saved into storage
Requirements filled	N/A
UC41A012 **Load production parameters pre-set**	**Priority:** Should
	Who: User **When:** In run-time **Where:** Cloud, in the user interface **What:** The User browses and sets the detection parameters for the sensor, adequate to current production, from a list of previously saved pre-sets **Why:** When the information already exists in Storage, the ability to load it will improve usability

Acceptance Criteria	The User browses and sets the detection parameters for the sensor, adequate to current production, from a list of previously saved pre-sets
Requirements filled	N/A
UC41A013 **Set production status**	**Priority:** Must
	Who: User **When:** In run-time **Where:** Cloud, in the user interface **What:** The user can set the production status to Started, Paused or Stopped **Why:** To avoid collecting and processing unnecessary data, data is discarded when production is not active
Acceptance Criteria	The user can set the production status to Started, Paused or Stopped
Requirements filled	N/A
UC41A014 **Trigger alarm on non-conformity**	**Priority:** Must
	Who: Non-destructive Inspection **When:** In run-time **Where:** Edge **What:** When the detection agent detects a non-conformity on data from a sensor it triggers an alarm **Why:** To minimize the time between an error occurring and the user's response, an alarm must be triggered as soon as possible
Acceptance Criteria	When the detection agent detects a non-conformity on data from a sensor it triggers an alarm
Requirements filled	N/A
UC41A015 **Alarm data is sent to platform**	**Priority:** Should
	Who: Non-destructive Inspection **When:** In run-time **Where:** Edge **What:** When the detection agent detects a non-conformity on data from a sensor, it sends the alarm information to the zApp Service **Why:** When non-conformities are detected on the factory floor, the platform should know about it
Acceptance Criteria	When the detection agent detects a non-conformity on data from a sensor, it sends the alarm information to the zApp Service
Requirements filled	N/A
UC41A016 **Send Notification to zApp UI of relevant users on non-conformity**	**Priority:** Could
	Who: zApp Service **When:** In run-time **Where:** Cloud **What:** When an alarm is received from the premises, a notification is sent to registered users **Why:** To maintain team members farther from the factory floor informed, an alarm is sent to the zApp UI
Acceptance Criteria	When an alarm is received from the premises, a notification is sent to registered users
Requirements filled	N/A
UC41A017 **Save alarm data**	**Priority:** Could
	Who: zApp Service **When:** In run-time **Where:** Cloud **What:** When an alarm is received from the premises, it is stored and associated with the current production **Why:** To maintain historical data
Acceptance Criteria	When an alarm is received from the premises, it is stored and associated with the current production
Requirements filled	N/A
UC41A018	**Priority:** Must

Collect data from sensor	**Who:** Data acquisition **When:** In run-time **Where:** Edge **What:** Data is collected from sensor for analysis **Why:** The data needs to be extracted from the sensor to be further exploited
Acceptance Criteria	Data is collected from sensor for analysis
Requirements filled	N/A
UC41A019 **Send sensor data for analysis**	**Priority:** Must
	Who: Data acquisition **When:** In run-time **Where:** Edge **What:** Data collected from sensor is sent to non-destructive inspection to detect non-conformities **Why:** The collected data needs to be analysed
Acceptance Criteria	Data collected from sensor is sent to non-destructive inspection to detect non-conformities
Requirements filled	N/A
UC41A020 **Upload sensor data**	**Priority:** Could
	Who: Data acquisition **When:** In run-time **Where:** Edge **What:** Data collected from sensor is sent to the ZDMP platform **Why:** The data is stored for historical consultation
Acceptance Criteria	Data collected from sensor is sent to the ZDMP platform
Requirements filled	N/A
UC41A021 **Save sensor data**	**Priority:** Could
	Who: zApp Service **When:** In run-time **Where:** Cloud **What:** When data produced by the sensors is received from the premises, it is stored and associated with the current production **Why:** To maintain historical data
Acceptance Criteria	When data produced by the sensors is received from the premises, it is stored and associated with the current production
Requirements filled	N/A
UC41A022 **Trigger alarm on sampling required**	**Priority:** Must
	Who: zApp Service **When:** In run-time **Where:** Cloud **What:** An alarm is triggered periodically for sample testing **Why:** The product requires to be sample tested at intervals for QA
Acceptance Criteria	An alarm is triggered periodically for sample testing
Requirements filled	N/A
UC41A023 **Submit sampling result**	**Priority:** Must
	Who: User **When:** In run-time **Where:** Cloud **What:** The user submits the testing result **Why:** The product must have all its sample testing associated
Acceptance Criteria	The user submits the testing result
Requirements filled	N/A
UC41A024	**Priority:** Must

Set sampling interval	**Who:** User **When:** In run-time **Where:** Cloud **What:** The user sets the sampling interval **Why:** The user must be able to configure the required sampling interval for testing
Acceptance Criteria	The user sets the sampling interval
Requirements filled	N/A
UC41A025 **Send Notification to zApp UI of relevant users on sample testing**	**Priority:** Could
	Who: zApp Service **When:** In run-time **Where:** Cloud **What:** A notification is sent periodically for sample testing **Why:** The product requires to be sample tested at intervals for QA. Notifying on the zApp increases visibility
Acceptance Criteria	A notification is sent periodically for sample testing
Requirements filled	N/A
UC41A026 **Consult and browse historical sensor data**	**Priority:** Could
	Who: User **When:** In run-time **Where:** Cloud **What:** The user can browse historical sensor data **Why:** To maintain a clear perspective of the production history
Acceptance Criteria	The user can browse historical sensor data
Requirements filled	N/A
UC41A027 **Consult and browse historical alarm data**	**Priority:** Could
	Who: User **When:** In run-time **Where:** Cloud **What:** The user can browse historical alarm data **Why:** To maintain a clear perspective of the production history
Acceptance Criteria	The user can browse historical alarm data
Requirements filled	N/A
UC41A028 **Consult and browse historical production details**	**Priority:** Could
	Who: User **When:** In run-time **Where:** Cloud **What:** The user can browse historical production data **Why:** To maintain a clear perspective of the production history
Acceptance Criteria	The user can browse historical production data
Requirements filled	N/A

Figure 253: zA4.01-zA4.04 Functions

7.7.3 Workflows

Two main workflows of non-trivial cases were identified, transversal to all the implementable zApps. The first workflow pertains to data collection in the context of non-conformity detection. That workflow is described in the following sequence diagram.

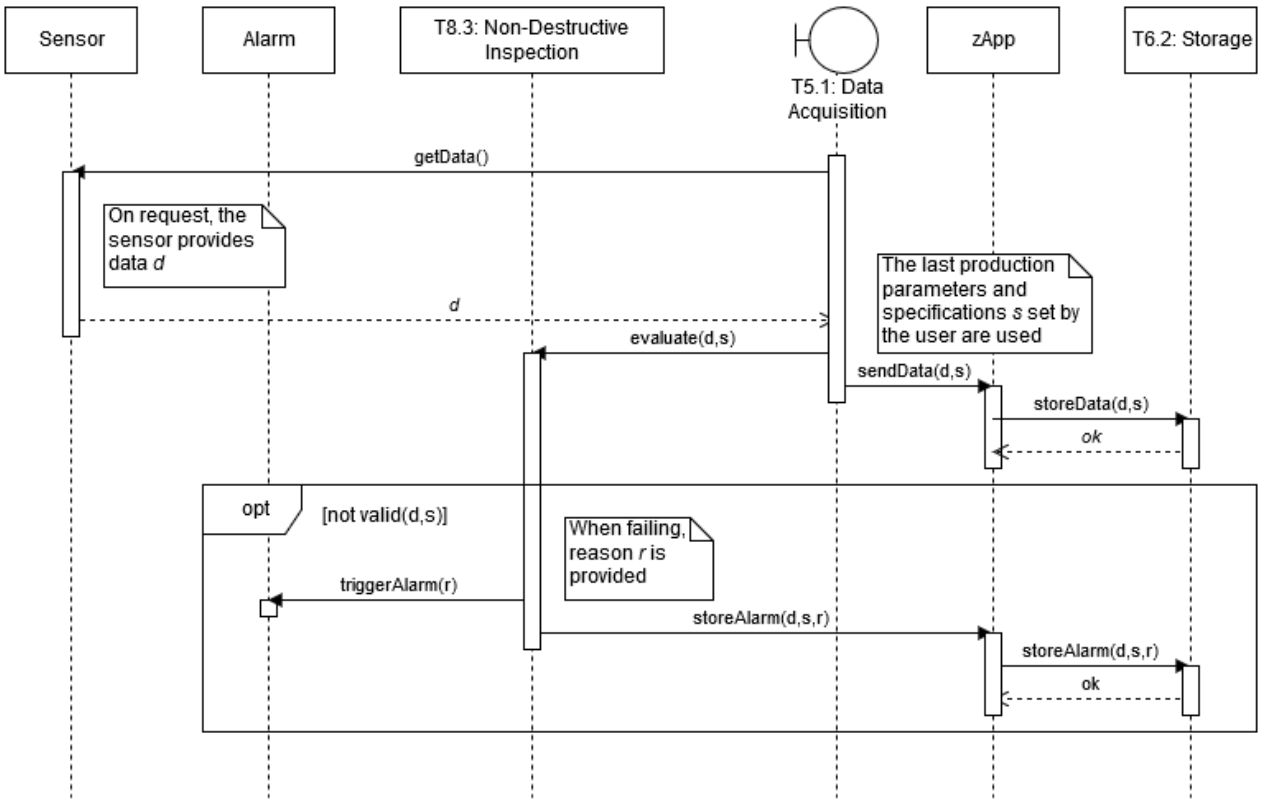


Figure 254: Data collection in the context of non-conformity detection

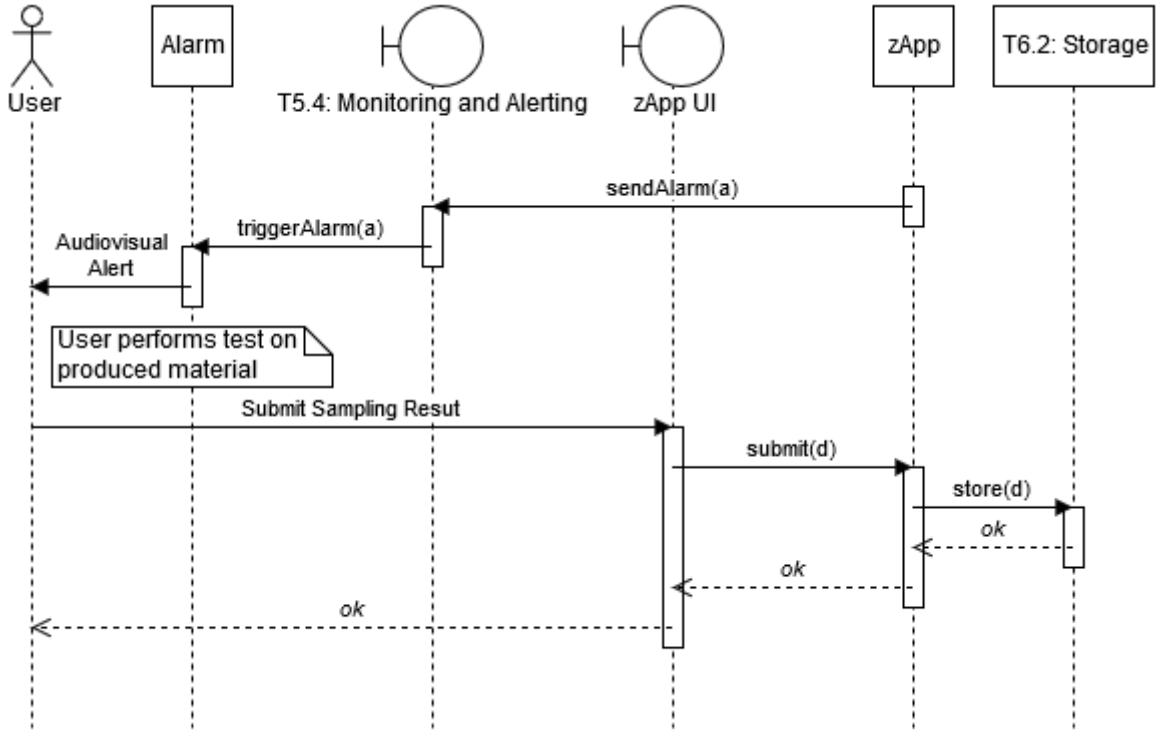


Figure 255: Repeatable warning regarding product sampling

The second workflow identified pertains to the repeatable warning regarding product sampling. That workflow is described in the following sequence diagram.

7.8 Stones Tiles: Equipment Wear Detection (zA4.05-4.09)

Stone cutting is a complex process involving several tools and typically is a time-consuming process, which depends on properties, type, and topography of stones. Stone is a natural material, with intrinsic complexity, and it is difficult for operators to control the quality along the processing phases. Thus, the use case 4.2 Detection Stone Tiles Equipment Wear aims to analyse and study the entire stone cutting process to optimize the stone cutting process and minimize costs and wastes.

Therefore, based on D2.3 - Industry Scenarios and Use Case and analysing the stone cutting process, four zApps are designed. These zApps intends to integrate various stages of stone cutting process and optimize and minimize the waste at each stage.

7.8.1 Overall functional characterization & Context

To optimize the entire stone cutting process four zApps are designed:

- zA4.05: zWiresMonitor: Aims to monitor the wire positions of the stone block cutting machine. This zApp is implemented in the first stage of stone cutting and aims to improve this stage by identifying and detecting anomalies in the cutting wires. It is expected that zWiresMonitor can detect broken wires or irregular wire movements.
- zA4.07: zDetectDefects: Allows the automatic detection of defects in stone slabs. Until now, the stone defect detection is a manual procedure executed by an operator. With the implementation of this zApp, the goal is to automatically detect defects in stone slabs. In this zApp the stone slabs analysis and automatic classification of stone defects are expected.
- zA4.08: zWornOutBladeDetection: Automatically identifies worn out blades during the stone slabs cutting process. This zApp aims to monitor the stone slab cutting machine to avoid possible anomalies during stone cutting.
- zA4.09: zTilesConformity: Aims to evaluate the compliance of final product (stone tiles). In this zApp, stone tiles are evaluated, and their compliance is verified against standard stone tile models.

7.8.2 Functions / Features

Subtask	Subtask description
UC41A001 **Set production details**	**Priority:** Must
	Who: User **When:** In run-time **Where:** Cloud **What:** The User sets the details about what is about to be produced **Why:** The system must know how to aggregate the data that is being collected into specific production relationships
Acceptance Criteria	The User sets the details about what is about to be produced
Requirements filled	N/A
UC41A002 **Save production details**	**Priority:** Should
	Who: User **When:** In run-time **Where:** Cloud **What:** After the User sets the production details, they are saved into storage **Why:** To enable historical data and to provide means for state recovery
Acceptance Criteria	After the User sets the production details, they are saved into storage
Requirements filled	N/A
UC41A003	**Priority:** Should

Load production details preset	**Who:** User **When:** In run-time **Where:** Cloud **What:** The User browses and sets the details about what is about to be produced from a list of previously stored presets **Why:** When the information already exists in Storage, the ability to load it will improve usability
Acceptance Criteria	The User browses and sets the details about what is about to be produced from a list of previously stored presets
Requirements filled	N/A
UC41A004 Multilingual support	**Priority:** Must
	Who: User **When:** In run-time **Where:** Cloud **What:** The ability to switch between languages for the user interface **Why:** The user interface must accommodate local dialects
Acceptance Criteria	The ability to switch between languages for the user interface
Requirements filled	N/A
UC41A005 Ability to select a zApp from a centralized view	**Priority:** Should
	Who: User **When:** In run-time **Where:** Cloud **What:** The user can select available zApps for his role **Why:** The user interface should provide a straightforward way to display and access all features
Acceptance Criteria	The user can select available zApps for his role
Requirements filled	N/A
UC41A006 Role based access control	**Priority:** Must
	Who: User **When:** In run-time **Where:** Cloud **What:** The platform controls the user access to zApps according to registered users and user roles (Administrator, Project Manager, Operator) **Why:** The platform must be able to provide or deny access to zApps according to user and roles
Acceptance Criteria	The platform controls the user access to zApps according to registered users and user roles (Administrator, Project Manager, Operator)
Requirements filled	N/A
UC41A007 Manage access control	**Priority:** Must
	Who: User **When:** In run-time **Where:** Cloud **What:** The User (Administrator) can create, delete, or update users, including attributing roles (Administrator, Project Manager, Operator) **Why:** The zApp or functional parts of the zApp must be restricted to specific users and according to a user role. The role of Administrator should be able to modify the dataset of allowed users
Acceptance Criteria	The User (Administrator) can create, delete, or update users, including attributing roles (Administrator, Project Manager, Operator)
Requirements filled	N/A
UC41A008	**Priority:** Must

Set production parameters	**Who:** User **When:** In run-time **Where:** Cloud **What:** The User sets the detection parameters for the sensor, adequate to current production **Why:** Depending on the desired end product, the parameters for non-conformant material differs. The user needs to adjust accordingly
Acceptance Criteria	The User sets the detection parameters for the sensor, adequate to current production
Requirements filled	N/A
UC41A009 **Import production parameters**	**Priority** Could
	Who: User **When:** In run-time **Where:** Cloud **What:** The User imports a custom file to set the detection parameters in respect to what is about to be produced **Why:** When the information already exists in the enterprise, the ability to import it will improve usability
Acceptance Criteria	The User imports a custom file to set the detection parameters in respect to what is about to be produced
Requirements filled	N/A
UC41A010 **Save production parameters**	**Priority:** Should
	Who: User **When:** In run-time **Where:** Cloud, in the user interface **What:** After the user sets the detection parameters for the sensor, they are saved into storage **Why:** To enable historical data and to provide means for state recovery
Acceptance Criteria	After the user sets the detection parameters for the sensor, they are saved into storage
Requirements filled	N/A
UC41A011 **Load production parameters pre-set**	**Priority:** Should
	Who: User **When:** In run-time **Where:** Cloud, in the user interface **What:** The User browses and sets the detection parameters for the sensor, adequate to current production, from a list of previously saved pre-sets **Why:** When the information already exists in Storage, the ability to load it will improve usability
Acceptance Criteria	The User browses and sets the detection parameters for the sensor, adequate to current production, from a list of previously saved pre-sets
Requirements filled	N/A
UC41A012 **Set production status**	**Priority:** Must
	Who: User **When:** In run-time **Where:** Cloud, in the user interface **What:** The user can set the production status to Started, Paused or Stopped **Why:** To avoid collecting and processing unnecessary data, data is discarded when production is not active
Acceptance Criteria	The user can set the production status to Started, Paused or Stopped
Requirements filled	N/A
UC41A013	**Priority:** Must

Trigger alarm on non-conformity	**Who:** Non-destructive Inspection **When:** In run-time **Where:** Edge **What:** When the detection agent detects a non-conformity on data from a sensor it triggers an alarm **Why:** To minimize the time between an error occurring and the user's response, an alarm must be triggered as soon as possible
Acceptance Criteria	When the detection agent detects a non-conformity on data from a sensor it triggers an alarm
Requirements filled	N/A
UC41A014 **Alarm data is sent to platform**	**Priority:** Should
	Who: Non-destructive Inspection **When:** In run-time **Where:** Edge **What:** When the detection agent detects a non-conformity on data from a sensor, it sends the alarm information to the zApp Service **Why:** When non-conformities are detected on the factory floor, the platform should know about it
Acceptance Criteria	When the detection agent detects a non-conformity on data from a sensor, it sends the alarm information to the zApp Service
Requirements filled	N/A
UC41A015 **Send Notification to zApp UI of relevant users on non-conformity**	**Priority:** Could
	Who: zApp Service **When:** In run-time **Where:** Cloud **What:** When an alarm is received from the premises, a notification is sent to registered users **Why:** To maintain team members farther from the factory floor informed, an alarm is sent to the zApp UI
Acceptance Criteria	When an alarm is received from the premises, a notification is sent to registered users
Requirements filled	N/A
UC41A016 **Save alarm data**	**Priority:** Could
	Who: zApp Service **When:** In run-time **Where:** Cloud **What:** When an alarm is received from the premises, it is stored and associated with the current production **Why:** To maintain historical data
Acceptance Criteria	When an alarm is received from the premises, it is stored and associated with the current production
Requirements filled	N/A
UC41A017 **Collect data from sensor**	**Priority:** Must
	Who: Data acquisition **When:** In run-time **Where:** Edge **What:** Data is collected from sensor for analysis **Why:** The data needs to be extracted from the sensor to be further exploited
Acceptance Criteria	Data is collected from sensor for analysis
Requirements filled	N/A
UC41A018	**Priority:** Must

Send sensor data for analysis	**Who:** Data acquisition **When:** In run-time **Where:** Edge **What:** Data collected from sensor is sent to non-destructive inspection to detect non-conformities **Why:** The collected data needs to be analysed
Acceptance Criteria	Data collected from sensor is sent to non-destructive inspection to detect non-conformities
Requirements filled	N/A
UC41A019 **Upload sensor data**	**Priority:** Could
	Who: Data acquisition **When:** In run-time **Where:** Edge **What:** Data collected from sensor is sent to the ZDMP platform **Why:** The data is stored for historical consultation
Acceptance Criteria	Data collected from sensor is sent to the ZDMP platform
Requirements filled	N/A
UC41A020 **Save sensor data**	**Priority:** Could
	Who: zApp Service **When:** In run-time **Where:** Cloud **What:** When data produced by the sensors is received from the premises, it is stored and associated with the current production **Why:** To maintain historical data
Acceptance Criteria	When data produced by the sensors is received from the premises, it is stored and associated with the current production
Requirements filled	N/A
UC41A021 **Validation production**	**Priority:** Must
	Who: User zApp Service **When:** In run-time **Where:** Edge **What:** The user must be able to validate the results **Why:** To avoid errors, the user must validate the results of automatic process
Acceptance Criteria	The user must be able to validate the results
Requirements filled	N/A
UC41A022 **Send Notification to zApp UI of relevant users on sample testing**	**Priority:** Could
	Who: zApp Service **When:** In run-time **Where:** Cloud **What:** A notification is sent periodically for sample testing **Why:** The product requires to be sample tested at intervals for QA. Notifying on the zApp increases visibility
Acceptance Criteria	A notification is sent periodically for sample testing
Requirements filled	N/A
UC41A023 **Consult and browse historical sensor data**	**Priority:** Could
	Who: User **When:** In run-time **Where:** Cloud **What:** The user can browse historical sensor data **Why:** To maintain a clear perspective of the production history
Acceptance Criteria	The user can browse historical alarm data
Requirements filled	N/A
UC41A024	**Priority:** Could

Consult and browse historical alarm data	**Who:** User **When:** In run-time **Where:** Cloud **What:** The user can browse historical alarm data **Why:** To maintain a clear perspective of the production history
Acceptance Criteria	The user can browse historical alarm data
Requirements filled	N/A
UC41A025 Consult and browse historical production details	**Priority:** Could
	Who: User **When:** In run-time **Where:** Cloud **What:** The user can browse historical production data **Why:** To maintain a clear perspective of the production history
Acceptance Criteria	The user can browse historical production data
Requirements filled	N/A

Figure 256: zA4.05-4.09 Functions

7.8.3 Workflows

In this section is exposed the sequential diagrams that represents all interactions between components and users in each zApp. The sequential diagrams are divided by zApp and in each of them the interactions between the components and users can be visualized.

7.8.3.1 zWireMonitor and zWornOutBlade

The following diagram explains these zApps functions and the necessary interactions with other components.

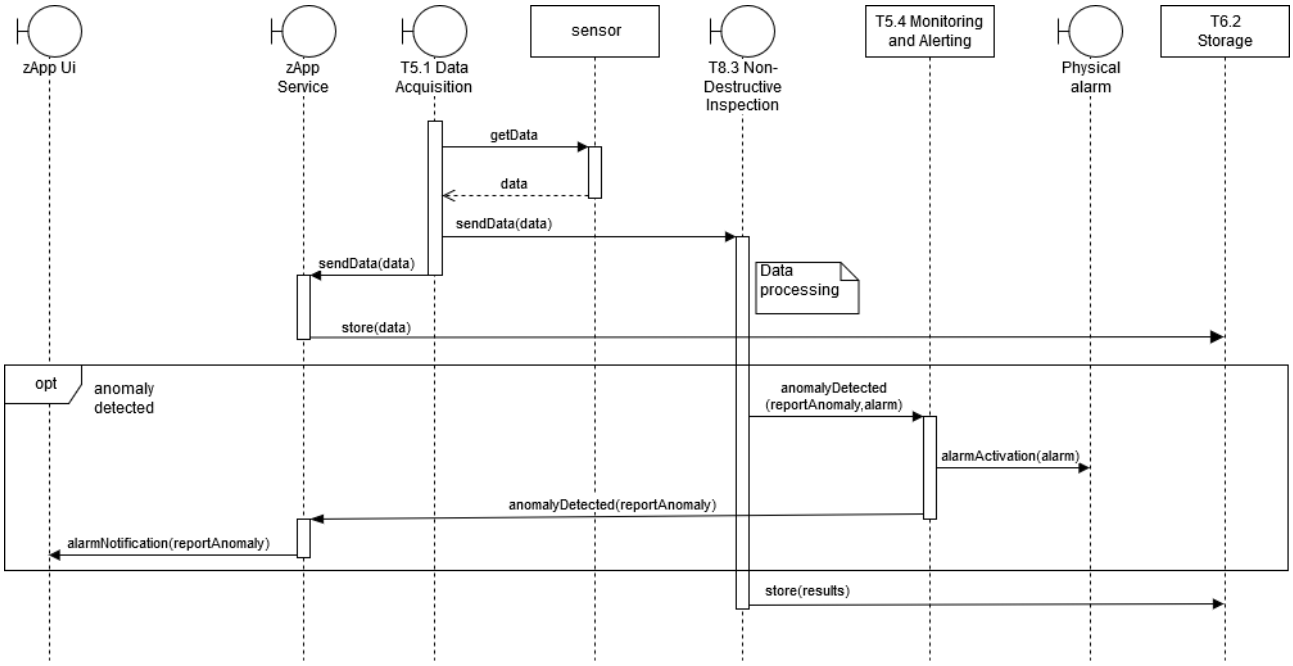


Figure 257: zWireMonitor and zWornOutBlade Sequence Diagram

7.8.3.2 zDetectDefects

The following diagram explains this function and the necessary interactions with other components.

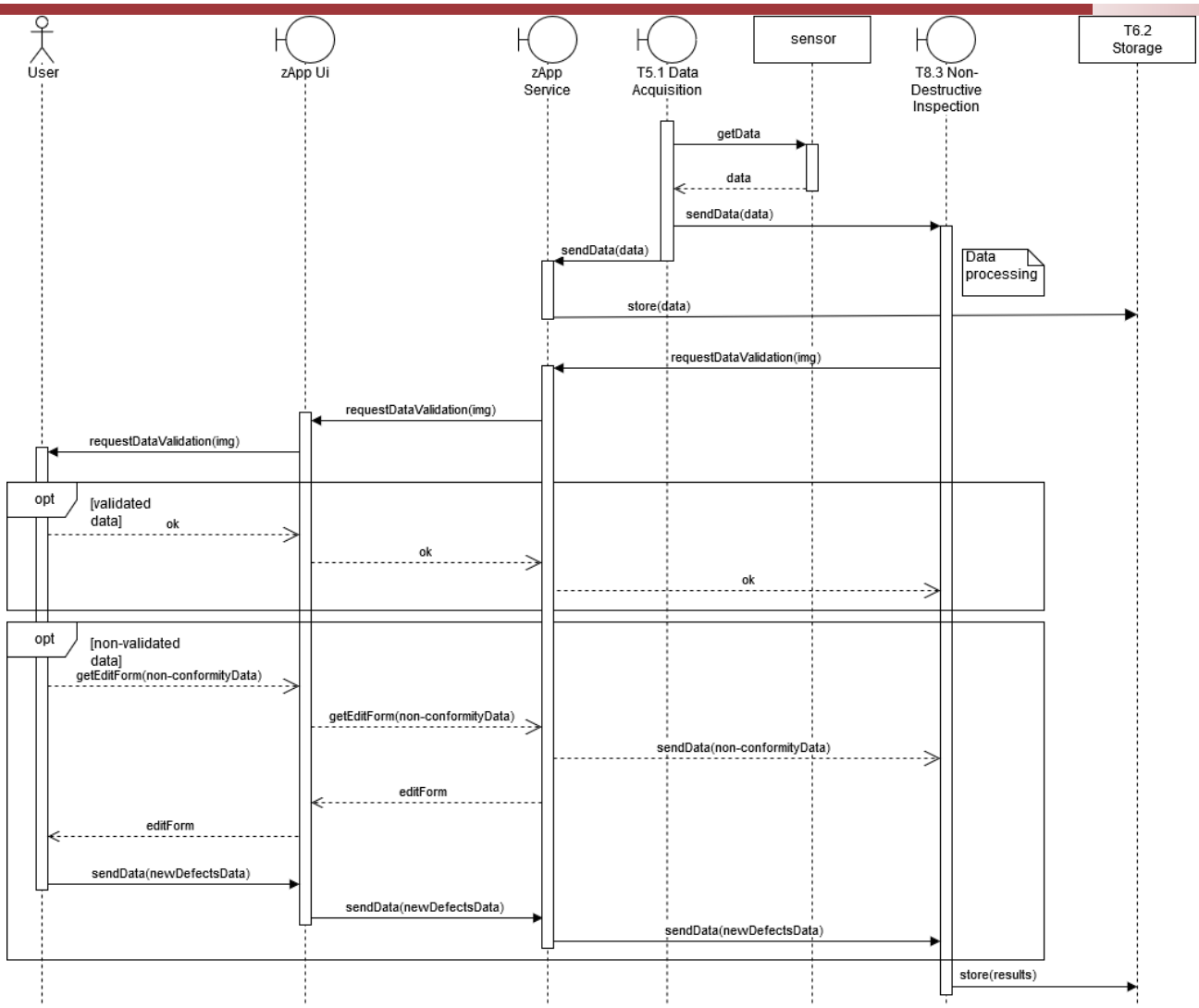


Figure 258: zDetectDefect Sequence Diagram

7.8.3.3 zTilesConformity

The following diagram explains this zApps function and the necessary interactions with other components.

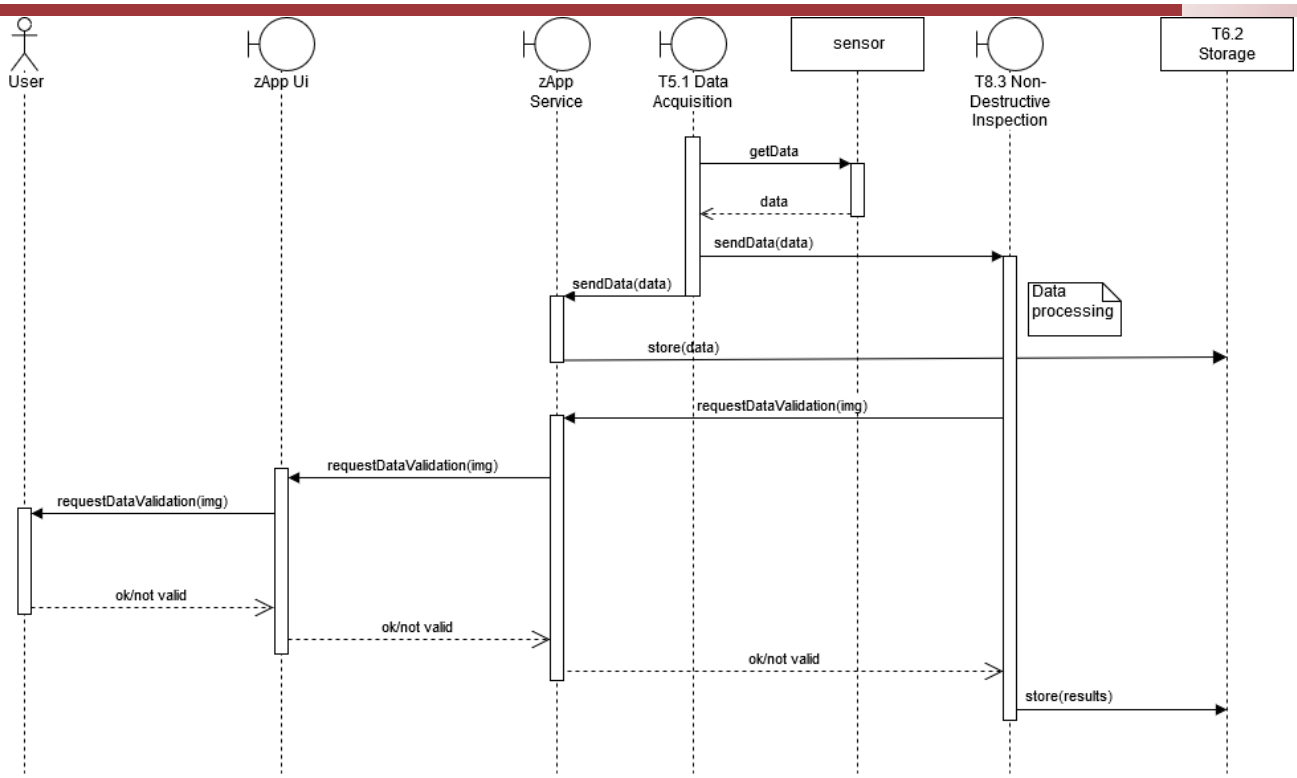


Figure 259: zTilesConformity Sequence Diagram

7.9 zRemoteQC (zA4.10)

7.9.1 Overall functional characterization & Context

The objective of zRemoteQC is to allow access to and easy archiving of documentary evidence of compliance regarding material including their specifications. This facilitates the documentation assessment and the detection of potential errors, even before the supplies leave the manufacturing facility. It will also allow the access to production quality control records of the corresponding material lots if the user chooses to do so.

7.9.2 Functions / Features

Subtask	Subtask description
zA4.10.001	**Priority:** Must
Role based access control and feature availability	**Who:** User **When:** In run-time **Where:** Cloud **What:** The platform controls the user access to zApps and individual features according to registered users and user roles (Administrator, Supplier, Works Contractor, Supervisor) **Why:** The platform must be able to provide or deny access to the zApps and individual features according to user and roles to reflect different user roles within the company
Acceptance Criteria	A registered user with an allowable role can access a zApp or specific features, and an unregistered user or with an unallowed role is unable to access a zApp or specific features

Requirements filled	N/A
zA4.10.002	**Priority:** Must
Manage access control	**Who:** User **When:** In run-time **Where:** Cloud **What:** The user (Administrator) can create, delete, or update users, including attributing roles (Administrator, Supplier, Works Contractor, Supervisor) **Why:** The users and roles need to be managed to reflect internal changes within the company
Acceptance Criteria	The user (Administrator) can create, delete, and update users. The User (Administrator) can change the role of a user
Requirements filled	N/A
zA4.10.003	**Priority:** Must
Creation and edition of Works Contractors	**Who:** User **When:** In run-time **Where:** Cloud **What:** The user (Supplier) can create a new or edit an existing Works Contractor **Why:** The user (Supplier) must be able to manage the existing dataset of Works Contractors
Acceptance Criteria	The user (Supplier) can create or edit Works Contractors
Requirements filled	N/A
zA4.10.004	**Priority:** Must
Association of materials to projects and Works Contractor	**Who:** User **When:** In run-time **Where:** Cloud **What:** The user (Supplier) can associate a new material to a project of a Works Contractor **Why:** The user (Supplier) must be able to associate the materials to the Works Contractor
Acceptance Criteria	The user (Supplier) can associate a new material to a project of a Works Contractor
Requirements filled	N/A
zA4.10.005	**Priority:** Must
Display detailed information on materials	**Who:** User **When:** In run-time **Where:** Cloud **What:** The user (Supplier) can inspect lots of materials being supplied, offering details in the form of documentation **Why:** The user (Supplier) must be able to access a list of materials to supply or already supplied
Acceptance Criteria	The user (Supplier) can list and access detailed information of lots of materials associated
Requirements filled	N/A
zA4.10.006	**Priority:** Should
Collect information of materials	**Who:** Data acquisition **When:** In run-time **Where:** Edge **What:** The system can extract information regarding the production status of materials from Suppliers' legacy systems and send it to the platform **Why:** The system should be able to extract real-time production information regarding specific materials
Acceptance Criteria	The system can extract and store production information
Requirements filled	N/A
zA4.10.007	**Priority:** Must

Manage deliveries	**Who:** User **When:** In run-time **Where:** Cloud **What:** The user (Supplier) can view a delivery schedule per client, retrieve detailed information for each order and assign a material to fulfil it **Why:** The user (Supplier) will manage the delivery schedule for existent orders
Acceptance Criteria	The user (Supplier) can view delivery schedule per client, retrieve detailed information for each order and assign a material to fulfil it
Requirements filled	N/A
zA4.10.008 **Authorize shipment**	**Priority:** Must **Who:** User **When:** In run-time **Where:** Cloud **What:** The user (Supplier) can authorize shipment of a fulfilled order **Why:** The user (Supplier) will manage the shipment
Acceptance Criteria	The user (Supplier) can authorize a shipment
Requirements filled	N/A
zA4.10.009 **Consult messages**	**Priority:** Should **Who:** User **When:** In run-time **Where:** Cloud **What:** The user can access a list of received messages and read them **Why:** The user will be notified of actions in the system and a system for displaying them is required
Acceptance Criteria	The user can access a list of notifications and read its content
Requirements filled	N/A

Figure 260: zRemoteQC Functions

7.10 zRescheduler (zA4.11)

7.10.1 Overall functional characterization & Context

The objective of zRemoteQC is to allow access to and easy archiving of documentary evidence of compliance regarding material including their specifications. This facilitates the documentation assessment and the detection of potential errors, even before the supplies leave the manufacturing facility. It will also allow the access to production quality control records of the corresponding material lots if the user chooses to do so.

7.10.2 Functions / Features

Subtask	Subtask description
zA4.11.001 **Role based access control and feature availability**	**Priority:** Must **Who:** User **When:** In run-time **Where:** Cloud **What:** The platform controls the user access to zApps and individual features according to registered users and user roles (Administrator, Supplier, Works Contractor, Supervisor) **Why:** The platform must be able to provide or deny access to the zApps and individual features according to user and roles to reflect different user roles within the company
Acceptance Criteria	A registered user with an allowable role can access a zApp or specific features, and an unregistered user or with an unallowed role is unable to access a zApp or specific features
Requirements filled	N/A

zA4.11.002 **Manage access control**	**Priority:** Must **Who:** User **When:** In run-time **Where:** Cloud **What:** The user (Administrator) can create, delete, or update users, including attributing roles (Administrator, Supplier, Works Contractor, Supervisor) **Why:** The users and roles need to be managed to reflect internal changes within the company
Acceptance Criteria	The user (Administrator) can create, delete, and update users. The User (Administrator) can change the role of a user
Requirements filled	N/A
zA4.11.003 **Works Contractor consults messages**	**Priority:** Must **Who:** User **When:** In run-time **Where:** Cloud **What:** The user (Works Contractor) can access messages regarding delivery dates **Why:** The user needs to be contacted regarding delivery date changes to have an up to date delivery calendar
Acceptance Criteria	The user can list and read messages
Requirements filled	N/A
zA4.11.004 **Works Contractor performs a reschedule**	**Priority:** Must **Who:** User **When:** In run-time **Where:** Cloud **What:** The user (Works Contractor), given a change in delivery date, performs a rescheduling **Why:** When delivery dates change, the project calendar must update accordingly
Acceptance Criteria	The user can perform a rescheduling
Requirements filled	N/A
zA4.11.005 **Works Supervisor consults messages**	**Priority:** Must **Who:** User **When:** In run-time **Where:** Cloud **What:** The user (Works Supervisor) can access messages regarding schedule changes **Why:** The users need to be aware of schedule changes
Acceptance Criteria	The user can list and read messages
Requirements filled	N/A
zA4.11.006 **Works Supervisor approves/disapproves a new schedule**	**Priority:** Must **Who:** User **When:** In run-time **Where:** Cloud **What:** The user (Works Supervisor) can approve or disapprove a schedule **Why:** The users need to be able to approve or disapprove changes to the schedule
Acceptance Criteria	The user can approve or disapprove a new schedule and that choice is respected
Requirements filled	N/A
zA4.11.007	**Priority:** Must

Supplier reports a delay	**Who:** User **When:** In run-time **Where:** Cloud **What:** The user (Supplier) reports a delay for a delivery **Why:** To maintain a consistent schedule, the zApp needs to be aware of changes in deliveries
Acceptance Criteria	The user can submit a delay
Requirements filled	N/A
zA4.11.008	**Priority:** Must
Supplier reports delivery	**Who:** User **When:** In run-time **Where:** Cloud **What:** The user (Supplier) reports a delivery has been completed **Why:** To maintain a consistent schedule, the zApp needs to be aware of changes in deliveries
Acceptance Criteria	The user can submit the delivery completion
Requirements filled	N/A
zA4.11.009	**Priority:** Should
User imports, exports legacy system data	**Who:** User **When:** In run-time **Where:** Cloud **What:** The user (Works Contractor and Works Supervisor) imports data in MS Project and MS Excel regarding scheduling. The data is exported back to the system on change **Why:** The organisation uses its internal system for keeping an up to date schedule. Connecting it with the platform would improve utility
Acceptance Criteria	The user can import MS Excel and MS Project files into the platform. When changes occur in the platform, they are exported back to the organisation
Requirements filled	N/A

Figure 261: zRescheduler zA 4.11 Functions

7.11 zMaterialTracker (zA4.12)

7.11.1 Overall functional characterization & Context

The objective of zMaterialTracker is to allow the recording of the use of a specific material at a specific location, and based on that, to allow access to all the documentation related to that specific material, be it construction records, quality control records, shipment records or production control records.

7.11.2 Functions / Features

Subtask	Subtask description
zA4.12.001	**Priority:** Must
Role based access control and feature availability	**Who:** User **When:** In run-time **Where:** Cloud **What:** The platform controls the user access to zApps and individual features according to registered users and user roles (Administrator, Supplier, Works Contractor, Supervisor) **Why:** The platform must be able to provide or deny access to the zApps and individual features according to user and roles to reflect different user roles within the company

Acceptance Criteria	A registered user with an allowable role can access a zApp or specific features, and an unregistered user or with an unallowed role is unable to access a zApp or specific features
Requirements filled	N/A
zA4.12.002 Manage access control	**Priority:** Must
	Who: User **When:** In run-time **Where:** Cloud **What:** The user (Administrator) can create, delete, or update users, including attributing roles (Administrator, Supplier, Works Contractor, Supervisor) **Why:** The users and roles need to be managed to reflect internal changes within the company
Acceptance Criteria	The user (Administrator) can create, delete, and update users. The User (Administrator) can change the role of a user
Requirements filled	N/A
zA4.12.003 User imports drawing files	**Priority:** Must
	Who: User **When:** In run-time **Where:** Cloud **What:** The user (Works Contractor, Works Supervisor) imports drawings from the legacy systems **Why:** The legacy systems contain drawings related to projects that need to be imported in ordered to be referenced
Acceptance Criteria	The user can import drawings into the platform and the platform understands the different areas described in the drawings
Requirements filled	N/A
zA4.12.004 User associates a material with a section	**Priority:** Must
	Who: User **When:** In run-time **Where:** Cloud **What:** The user (Works Contractor, Works Supervisor) selects a section reference and associates an existent material for application **Why:** To control the application of materials they must be positioned accurately and accounted for
Acceptance Criteria	The user can reference a material to a location
Requirements filled	N/A
zA4.12.005 User can visually select a location in a drawing	**Priority:** Should
	Who: User **When:** In run-time **Where:** Cloud **What:** The user (Works Contractor, Works Supervisor) selects a section reference from a visual representation **Why:** To facilitate user perception, visual tools will be offered
Acceptance Criteria	The user can select a section reference from a visual representation
Requirements filled	N/A
zA4.12.006 User imports PDF, Excel, and Word files	**Priority:** Should
	Who: User **When:** In run-time **Where:** Cloud **What:** The user (Works Contractor, Works Supervisor) imports files from the legacy systems containing information regarding the materials **Why:** To maintain an overall view of all the documentation related to a material, information needs to be imported
Acceptance Criteria	The user can import files into the platform related to a material
Requirements filled	N/A
zA4.12.007	**Priority:** Should

User browses location and material information	**Who:** User **When:** In run-time **Where:** Cloud **What:** The user (Works Contractor, Works Supervisor) browses the materials and locations in the system and can search for any information **Why:** To facilitate inspection of applied materials, it should be easy to locate it from either vector
Acceptance Criteria	The user can find information related to a material either by searching for it directly or from a location
Requirements filled	N/A

Figure 262: zA4.12 Functions

7.12 zMaterialID (zA4.13)

7.12.1 Overall functional characterization & Context

The purpose of zMaterialID is to create an identification system capable of creating a unique identifier for varied materials and corresponding quality control information. Through this identifier the materials will be traceable throughout the production process

7.12.2 Functions / Features

Subtask	Subtask description
zA4.13.001 Role based access control and feature availability	**Priority:** Must
	Who: User **When:** In run-time **Where:** Cloud **What:** The platform controls the user access to zApps and individual features according to registered users and user roles (Administrator, Supplier, Works Contractor, Supervisor) **Why:** The platform must be able to provide or deny access to the zApps and individual features according to user and roles to reflect different user roles within the company
Acceptance Criteria	A registered user with an allowable role can access a zApp or specific features, and an unregistered user or with an unallowed role is unable to access a zApp or specific features
Requirements filled	N/A
zA4.13.002 Manage access control	**Priority:** Must
	Who: User **When:** In run-time **Where:** Cloud **What:** The user (Administrator) can create, delete, or update users, including attributing roles (Administrator, Supplier, Works Contractor, Supervisor) **Why:** The users and roles need to be managed to reflect internal changes within the company
Acceptance Criteria	The user (Administrator) can create, delete, and update users. The User (Administrator) can change the role of a user
Requirements filled	N/A
zA4.13.003 User imports material id	**Priority:** Should
	Who: User **When:** In run-time **Where:** Cloud **What:** The user (Supplier) imports from his legacy system the id related to a material **Why:** The user must supply the system with an id related to produced material. If the legacy system already contains such id, it might be less prone to errors

Acceptance Criteria	The user can select and upload an id from his legacy system into the platform
Requirements filled	N/A
zA4.13.004 User defines material id	**Priority:** Must
	Who: User **When:** In run-time **Where:** Cloud **What:** The user (Supplier) defines manually the id related to a material **Why:** The user must supply the system with an id related to produced material
Acceptance Criteria	The user can define an id for a material
Requirements filled	N/A
zA4.13.005 User inspects material id	**Priority:** Should
	Who: User **When:** In run-time **Where:** Cloud **What:** The user (Supplier) can inspect and browse the materials already in the system **Why:** The user might need to have an overarching view of the current production status within the system
Acceptance Criteria	The user can find the information for any material already in the system
Requirements filled	N/A

Figure 263: zA4.13 Functions

7.13 zXRAYMonitor & Analytics (zA3.01 & zA3.02)

7.13.1 Overall functional characterization & Context

zXRAYMonitor and zXRAYAnaytics are two applications that aim to improve the maintenance and operations of an X-Ray test unit. Hence, this section describes both applications together for the sake of clarity. zXRAYMonitor facilitates the correct execution of the tests planned for a specific product in an inspection unit. The test unit conducts tests based on X-ray Computed Tomography (CT) technology. The application detects the product type of the product under test via a barcode reader or RFID reader. Barcodes or RFID tags encode the part number: A unique identifier for the product type. When the product type is identified, the application loads the parameters for the product type into the testing unit. The X-ray inspection unit is used to characterise specific properties of the product (like properties of the materials). Thus, for example, the application can be used to set up the threshold parameters that determine the tests for a property of a product type. When the test unit has performed the test, the application collects and stores the tests results together with the X-ray image generated by the X-ray inspection unit. Functions of zXRAYMonitor are identified with ids zA3.01-9. Based on the information provided by zXRAYMonitor, zXRAYAnalytics provides graphic dashboards showing statistical analysis to the test engineer, so that the programming of the test unit can be further improved.

7.13.2 Functions / Features

- **Data sources configuration:** Backend function to configure the data sources used to integrate master data. Master data includes product type information, and inspection test information.
- **Industrial data collection configuration:** Backend function to configure the connections to exchange data with the test unit control. The user can define different industrial variables. Industrial variables are used to read the product instance part

number, load the X-ray inspection program parameters (eg thresholds for each property), read the X-ray inspection unit results, and read the location of the X-ray.

- **Product type configuration:** Backend function to edit the properties and test sequence for a specific product type. The user can select the product type and edit the test sequence. For every test, the user can edit the configuration parameters of the test. The configuration of a test consists of the write values and industrial variables for the testing program parameters and the industrial variables used to collect the results.
- **Part number detection:** To detect the product type unique identifier of the product under test.
- **Test unit preparation:** To load the configuration parameters of the X-ray inspection program for the product instance under test. This function consists in writing the specified configuration values into the specified industrial variables using the configured connections.
- **Test detection:** To detect the test being performed in the test unit
- **Test unit results collection:** To collect and store the results of the test unit, linked to the configuration parameters, the product part type and the location of the X-ray tomography image. This function consists in reading the specified industrial variables using the configured connections.
- **Results analysis**: To make a detailed statistical analysis of the results, comparing new measurements with historic data and analyse trends in material properties to predict possible out-of-tolerance parts.

These functions can be further decomposed into the following subtasks that have to be realised:

Subtask	Subtask description
ZA3.01.1 Configure master data sources	**Priority:** Should
	Who: ZDMP consultant **When / Where:** During configuration (runtime), on premise **What:** User selects data sources (files, database connections) to integrate master data **Why:** To integrate master data information
Acceptance Criteria	The user can create a new master data configuration The user can create a new data source The user can select data sources that have been previously configured in the platform The user can update a master data configuration with the selected data source Master data provides access to product type information
Requirements filled	RQ_0240, RQ_421
ZA3.01.2 Configure test unit connection	**Priority:** Should
	Who: ZDMP consultant **When / Where:** During configuration (runtime), on premise **What:** User configures a connection to the test unit control **Why:** To configure the connection to the test unit control and enable data exchange
Acceptance Criteria	The user can create a new unit model The user can edit the connection parameters Optionally, the user could browse the field network to search for available connections
Requirements filled	RQ_0240, RQ_0288
ZA3.01.3	**Priority:** Should

Configure industrial variables	**Who:** ZDMP consultant **What:** User configures industrial variables **When / Where:** During configuration (runtime), on premise **Why:** To uniquely identify a test parameter in the test unit control program
Acceptance Criteria	The user can create a new industrial variable The user can specify the necessary parameters to read and write data from the variable The user can edit the model to specify which variables are used to read and write test unit data The user can select industrial variables that have been previously configured The user can update the model with the selected variables
Requirements filled	RQ_239, RQ_0240, RQ_421, RQ_0288
ZA3.01.4 Edit test configuration of product type	**Priority:** Must
	Who: Test Engineer **When / Where:** During configuration (runtime), on premise **What:** User selects a product type **Why:** To edit the configuration of the test
Acceptance Criteria	The user can edit the configuration The user can select a product type from master data The user can set the product type of the configuration
Requirements filled	RQ_0240, RQ_245, RQ_0258, RQ_0288
ZA3.01.5 Edit test sequence configuration	**Priority:** Must
	Who: Test Engineer **When / Where:** During configuration (runtime), on premise **What:** User edits the configuration of the tests for a product type **Why:** To specify the configuration of the tests for a product type
Acceptance Criteria	The user can edit the test sequence of the configuration
Requirements filled	RQ_0240, RQ_245, RQ_0258
ZA3.01.6 Set test configuration	**Priority:** Must
	Who: Test Engineer **When / Where:** During configuration (runtime), on premise **What:** User sets the configuration of a specific test **Why:** To specify the values of the test program parameters and where to read the results
Acceptance Criteria	The user can create a test configuration for a product type The test configuration has a unique test identifier The user can select an industrial variable to read the test identifier The user can add a sample image The user can add a write parameter to a test configuration to write the value of a configuration parameter in the test unit For every write parameter, the user can set a name For every write parameter, the user can select an industrial variable to write the parameter For every write parameter, the user can set the value to write For every write parameter, the user can set the industrial variable to write The user can add a read parameter to a test configuration to read the value of the part type or test result variable For every read parameter, the user can set a name For every read parameter, the user can set an industrial variable to read the value from
Requirements filled	RQ_0240, RQ_243, RQ_244, RQ_245, RQ_0253, RQ_0258
ZA3.01.7 **Detect product type**	**Priority:** Must
	Who: Test unit data collection adapter **When / Where:** Before the test unit starts the test sequence (runtime), in the test unit **What:** Read the part number of the product instance under test

	Why: To detect which is the product type and load the right configuration parameters
Acceptance Criteria	The test unit model contains the parameters to connect to the test unit control and exchange information The test unit adapter can connect to the test unit control The test unit adapter can publish data to exchange data with other components The test unit adapter can update the test unit model with the topics of the parameters to be exchanged The test unit control is connected to a reader and stores the part number code in the part number variable The test unit adapter can read the part number variable control and publish the part number code value in the configured read parameter
Requirements filled	RQ_246
ZA3.01.8 Load configuration parameters	**Priority:** Must
	Who: Test unit data collection adapter **When / Where:** Before the test unit starts the test sequence (runtime), in the test unit **What:** write the values of the test configuration parameters in the test unit control program **Why:** To configure the test unit for the product type
Acceptance Criteria	The test unit adapter can get the variables and values to write for the product type in the test unit model The test unit adapter can write the values in the test unit control variables indicated in the configuration
Requirements filled	RQ_0239, RQ_0240, RQ_0248
ZA3.01.9 Test detection	**Priority:** Must
	Who: Test unit data collection adapter **When / Where:** When the test unit starts the test sequence (runtime), in the test unit **What:** Read the test performed in the test unit **Why:** To know which is the test that the unit performed
Acceptance Criteria	The test unit adapter can read the test unique identifier in the configured variable
Requirements filled	RQ_0239
ZA3.01.10 Collect test results	**Priority:** Must
	Who: Test unit data collection adapter **When / Where:** When the test unit finishes a test (runtime), in the test unit **What:** Read the test results **Why:** To detect the test results
Acceptance Criteria	The test unit adapter can read the test results in the configured variables The test unit adapter should save the binary data of the X-Ray tomography image in an image file in a network file system The test unit adapter can publish the test results so that they can be accessed by other applications The test results should include a resource identifier of the X-Ray tomography image The test results may include the binary data of the X-Ray tomography image
Requirements filled	RQ_0239, RQ_254
ZA3.01.11 Save test results	**Priority:** Must
	Who: Application storage **When / Where:** When the test unit starts the test sequence (runtime), in the test unit **What:** Save the test results **Why:** To provide information and insights to the test engineer
Acceptance Criteria	The test unit adapter can create a test record to store the results The test record includes the parameters and values used for the test The test record includes the product type The test records include the test unique identifier

	The test records should include the resource identifier of the X-Ray tomography image
Requirements filled	RQ_0239, RQ_249, RQ_0250, RQ_251, RQ_252, RQ_255, RQ_0256, RQ_0257, RQ_261
ZA3.02.1 Analyse results	**Priority:** Must
	Who: Test engineer **When / Where:** When the test engineer is optimising the test unit program, in the test department **What:** Analyse the results **Why:** To further improve the test program
Acceptance Criteria	The test engineer can set parameters to filter data and configure analytics Product quality prediction can use test results records as inputs Product quality prediction can identify trends in the properties of the materials and predict the properties of future batches Product quality prediction can publish the results
Requirements filled	RQ_0239, RQ_260, RQ_262, RQ_263, RQ_264 RQ_0268, RQ_0269, RQ_0270, RQ_0271, RQ_0272, RQ_0273, RQ_0274
ZA3.02.2 Send notifications	**Priority:** Must
	Who: Purchase department / Suppliers **When / Where:** When the statistical analysis predicts a possible out-of-tolerance part **What:** Send an email notification **Why:** To warn involved parties
Acceptance Criteria	Monitoring and alerting can subscribe to analytic results Monitoring and alerting can publish notifications when the results match the configured rules The test engineer can check statistical properties of the test results The test engineer can check the correlation of the parameters and test results
Requirements filled	RQ_0239, RQ_265, RQ_0266, RQ_0267, RQ_0275, RQ_277, RQ_0278, RQ_0279

Figure 264: zA3.01 & zA3.02 Functions

7.13.3 Workflows

7.13.3.1 Low level configuration

This workflow sets up the application configuration parameters that are more integrated into the ZDMP Platform, like data sources and connections to exchange data with manufacturing assets. The main steps are:

- Configure data sources for master data
- Configure the connection to the test unit controller
- Configure the resource location of the variables that is used to exchange information with the test unit controller

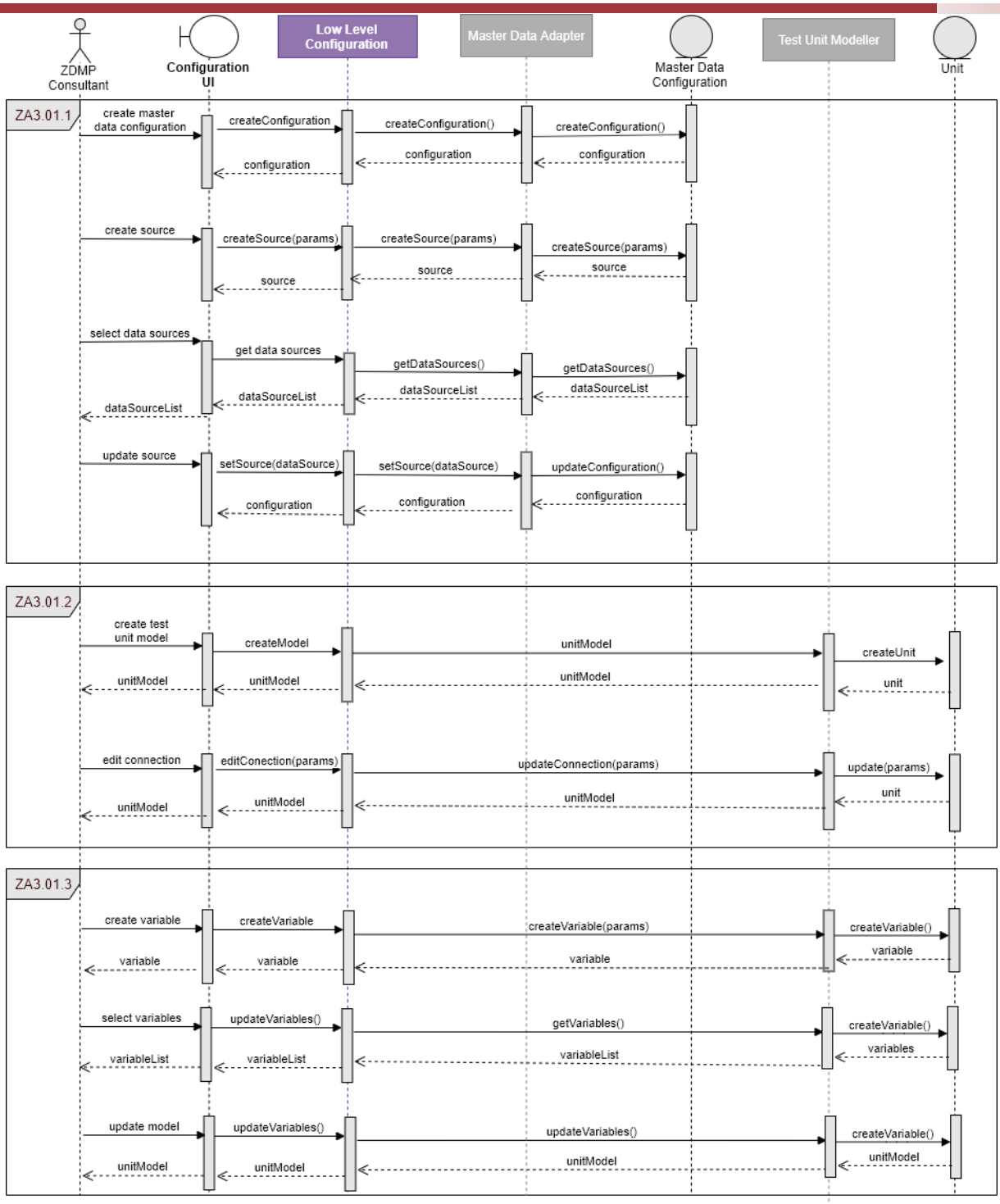


Figure 265: Low Level Configuration Sequence Diagram

7.13.3.2 Test unit sequence configuration

This workflow sets up the configuration of the tests to be conducted for every product type. The main steps are:

- Select product type
- Edit test sequence configuration
- Edit test configurations

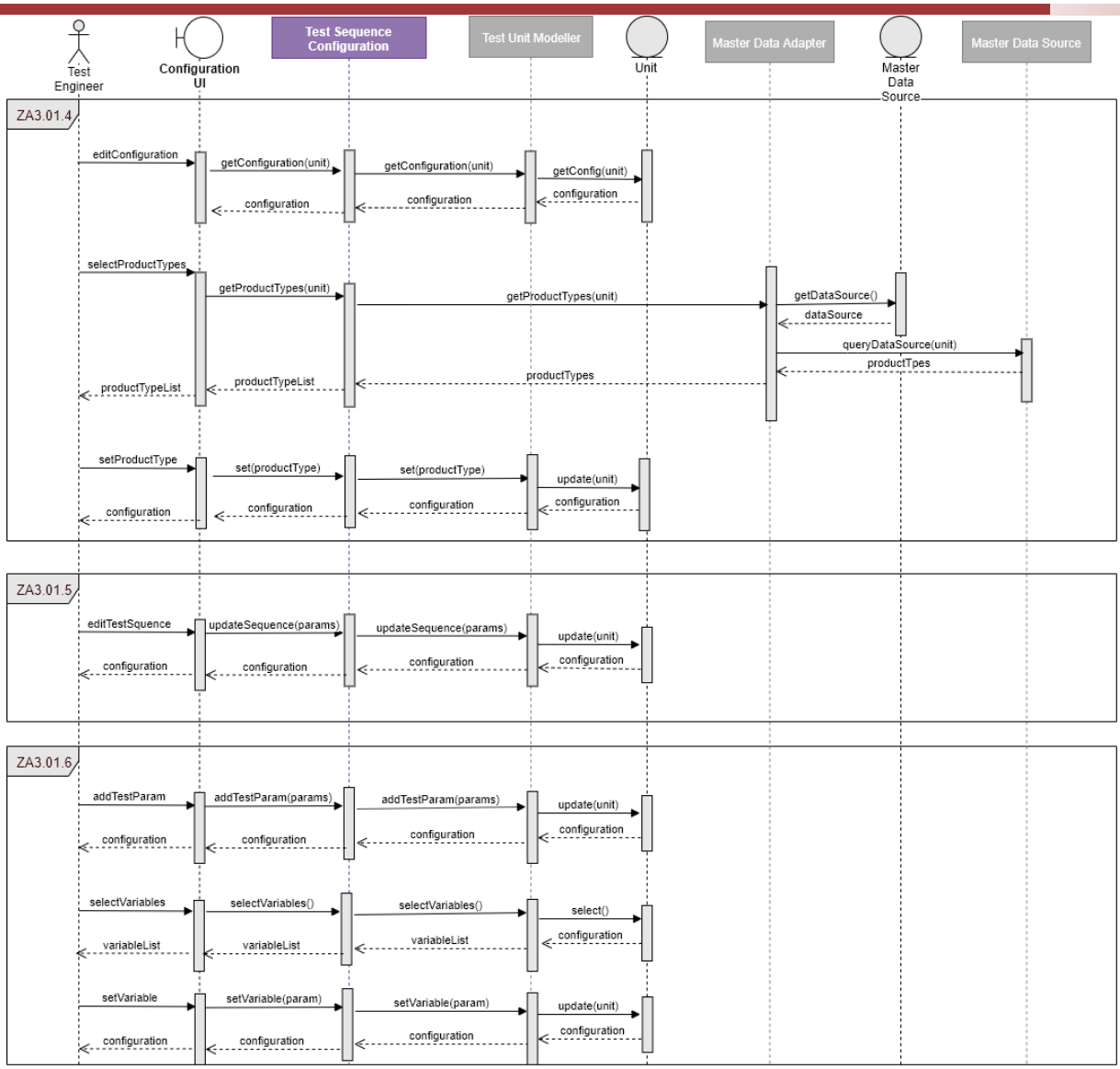


Figure 266: Test Unit Sequence Configuration Sequence Diagram

7.13.3.3 Test unit data exchange

This workflow manages all data exchange with the test unit. The main steps are:

- Detect product type
- Load configuration parameters
- Collect tests results
- Store tests results

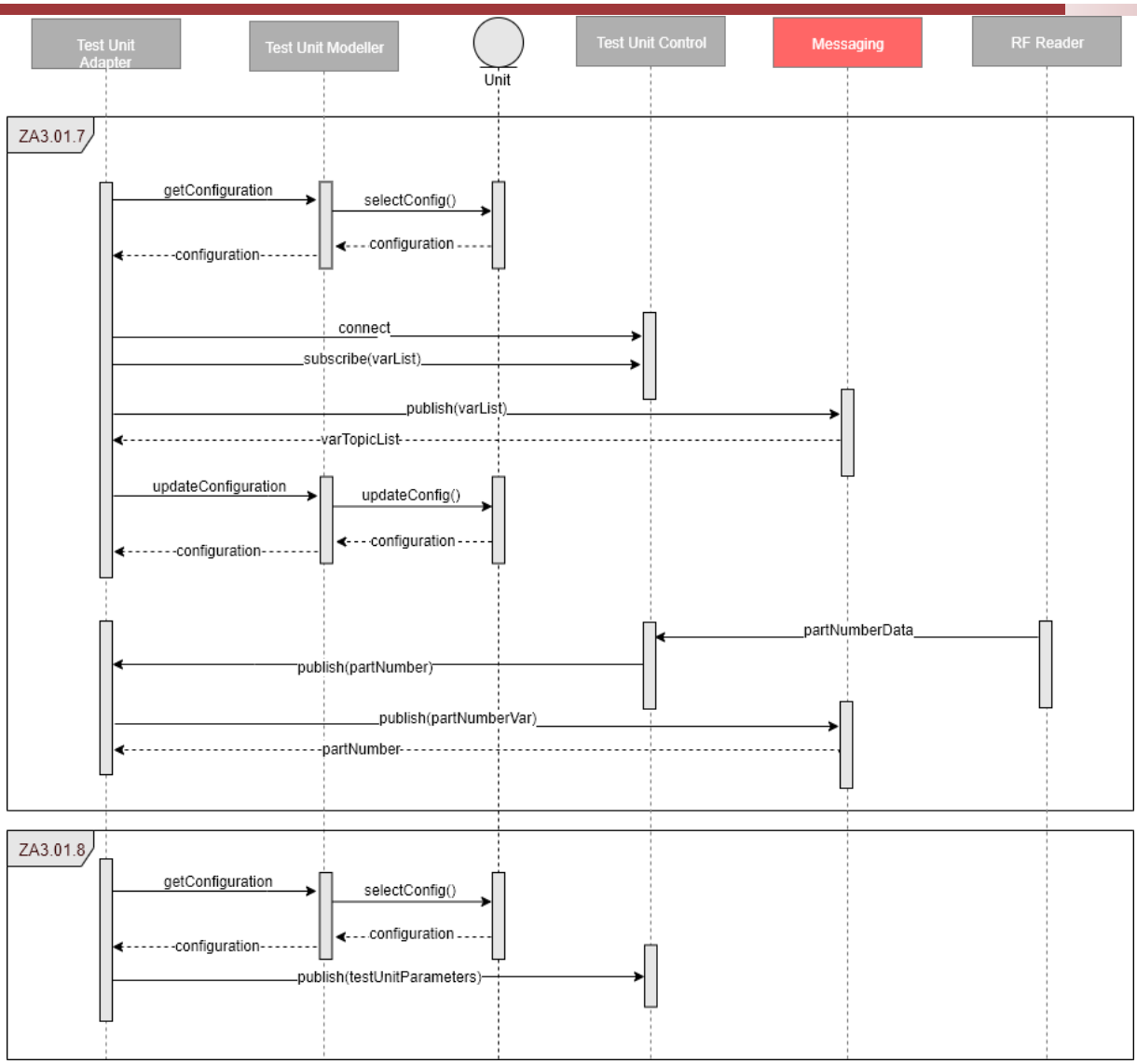


Figure 267: Product Type Detection and Configuration Parameter Load

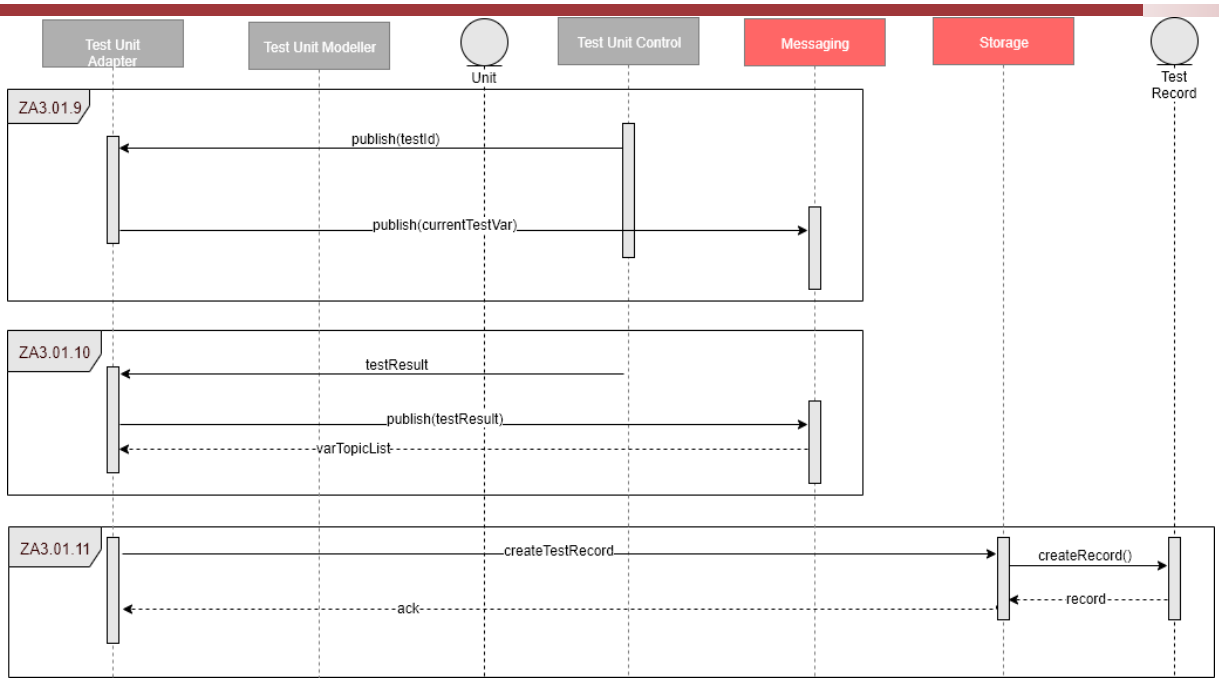


Figure 268: Test Results Collection and Storage

7.13.3.4 Result Analysis

This workflow provides insights on the collected data. The main steps are:

- Analyse results
- Send notifications

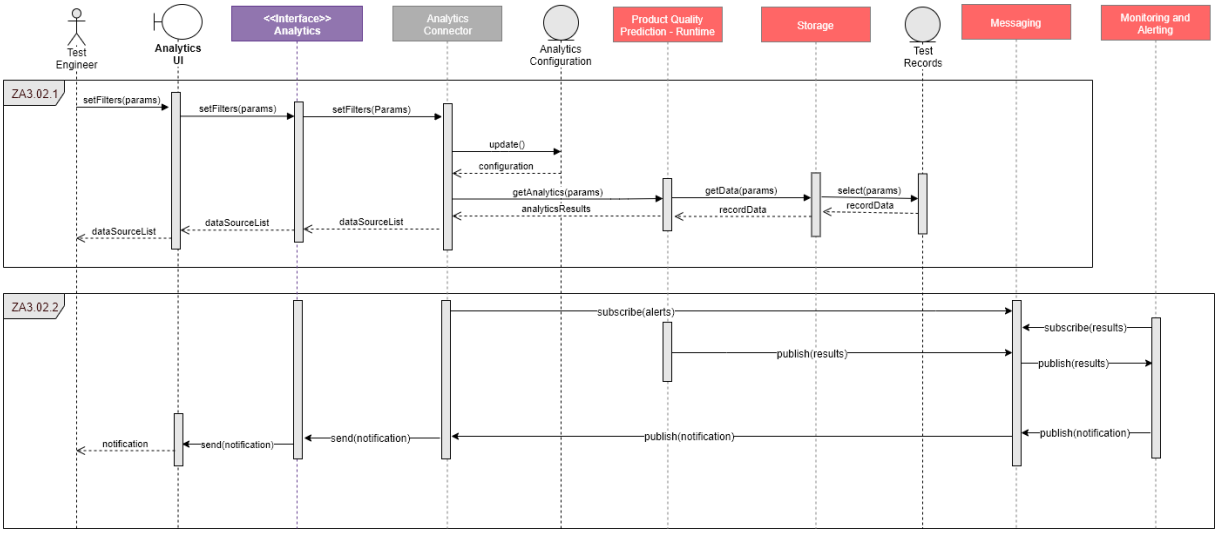


Figure 269: Result Analysis Sequence Diagram

7.13.4 Additional Issues

Additional issues have appeared after the description of the functional specification.

Issue	Description	Next Steps	Lead (Rationale)
Filled Requirements	The following requirements with a “must“-priority were targeted at different platform components, but are specific to this application and thus partially filled by its functions RQ_0241, RQ_0246, RQ_0247, RQ_0248, RQ_0249, RQ_050, RQ_0251, RQ_0252, RQ_0255, RQ_0262, RQ_0263, RQ_0264, RQ_0265, RQ_0266, RQ_0267, RQ_0268, RQ_0269, RQ_0270, RQ_0271, RQ_0272, RQ_0273, RQ_0274, RQ_0277	Discuss with requirement providers who to solve the issue	Product owner UPV

7.14 zFeedbackMFT (zA3.03)

7.14.1 Overall functional characterization & Context

zFeedbackMFT enables the connection to a test unit to collect the results of manual tests. In manual tests, the test unit drives the product under test (electronic component consisted of a display, a set of LED indicators, and a set of controls) into the test configuration (eg display showing a test image, activate controls to drive some LED indicators on, etc.) and the operator pushes an actuator to indicate whether the unit is conformant with the test or not. To make this decision, the operator uses a reference image and additional information describing the status of the product under test when there are no failures. This information is available in a user interface of the application. The user interface also shows an image of the product in the test unit taken with an external vision interface.

7.14.2 Functions / Features

- **Data sources configuration:** Backend function to configure the data sources used to integrate master data. Master data includes product type information, test information, and reference images.
- **Industrial data collection configuration:** Backend function to configure the connections to exchange data with the test unit control. The user can define different industrial variables. Industrial variables are used to read the product instance part number, the current test, and the manual test results (determined by the operator through actuators).
- **Product type configuration:** Backend function to edit the properties and test sequence for a specific product type. The user can select the product type and edit the test sequence. For every test, the user can edit the configuration parameters of the test. The configuration of a test consists industrial variables used to collect the results.
- **Part number detection:** To detect the product type unique identifier of the product under test.
- **Test detection:** To detect the test being performed in the test unit
- **Test operator support:** To present the information of the test to the operator in a friendly user interface.
- **Test unit results collection:** To collect and store the results of the manual test unit this function consists in reading the specified industrial variables representing the status of the actuators pushed by the operator.

These functions can be further decomposed into the following subtasks that have to be realised:

Subtask	Subtask description
ZA3.03.1 Configure master data sources	**Priority:** Should
	Who: ZDMP consultant **When / Where:** During configuration (runtime), on premise **What:** User selects data sources (files, database connections) to integrate master data **Why:** To integrate master data information
Acceptance Criteria	The user can create a new master data configuration The user can create a new data source The user can select data sources that have been previously configured in the platform The user can update a master data configuration with the selected data source Master data should include part type information Master data should include manual test configuration
Requirements filled	RQ_0281, RQ_0286
ZA3.03.2 Configure test unit and camera connection	**Priority:** Should
	Who: ZDMP consultant **When / Where:** During configuration (runtime), on premise **What:** User configures a connection to the test unit control **Why:** To enable data exchange
Acceptance Criteria	The user can create a new unit model The user can edit the connection parameters Optionally, the user could browse the field network to search for available connections
Requirements filled	RQ_0281
ZA3.03.3 Configure industrial variables	**Priority:** Should
	Who: ZDMP consultant **When / Where:** During configuration (runtime), on premise **What:** User configures industrial variables **Why:** To exchange data with the test unit control program
Acceptance Criteria	The user can create a new industrial variable The user can specify the necessary parameters to read and write data from the variable The user can edit the model to specify which variables are used to read and write test unit data The user can select industrial variables that have been previously configured The user can update the model with the selected variables
Requirements filled	RQ_0281
ZA3.03.4 Edit test configuration of product type	**Priority:** Must
	Who: Test Engineer **When / Where:** During configuration (runtime), on premise **What:** User selects a product type **Why:** To edit the configuration of the test
Acceptance Criteria	The user can edit the configuration The user can select a product type from master data The user can set the product type of the configuration
Requirements filled	RQ_0281
ZA3.03.5 Edit test sequence configuration	**Priority:** Must
	Who: Test Engineer **When / Where:** During configuration (runtime), on premise **What:** User edits the configuration of the tests for a product type **Why:** To specify the configuration of the tests for a product type
Acceptance Criteria	The user can edit the test sequence configuration

Requirements filled	RQ_0280, RQ_0281
ZA3.03.6 Set manual test configuration	**Priority:** Must
	Who: Test Engineer **When / Where:** During configuration (runtime), on premise **What:** User sets the configuration of a specific manual test **Why:** To specify the location of the reference image and additional parameters to support the operator
Acceptance Criteria	The user can create a manual test The manual test has a unique identifier The user can set a name for the manual test The user can assign a reference image to the manual test
Requirements filled	RQ_0281, RQ_0282, RQ_0283, RQ_0284, RQ_0287
ZA3.03.7 Detect part number	**Priority:** Must
	Who: Test unit adapter **When / Where:** When the test unit starts the test sequence (runtime), in the test unit **What:** Detect the product part number of the product under test **Why:** To determine the product type of the product under test
Acceptance Criteria	The test unit model contains the parameters to connect to the test unit control and exchange information The test unit adapter can connect to the test unit control The test unit adapter can publish data to exchange data with other components The test unit adapter can update the test unit model with the topics of the parameters to be exchanged The test unit control is connected to a reader and stores the part number code in the part number variable The test unit adapter can read the part number variable control and publish the part number code value in the configured read parameter
Requirements filled	RQ_0295
ZA3.03.8 Detect current test	**Priority:** Must
	Who: Test unit adapter **When / Where:** When the test unit drives the test product to perform a manual test (runtime), in the test unit **What:** Detect the test that is conducted in the unit **Why:** To determine the test unique identifier of the product test
Acceptance Criteria	The test unit model contains the parameters to connect to the camera control The test unit model contains the parameters to connect to the test unit control The test unit adapter can connect to the camera control The test unit adapter can publish in messaging the resource identifier of the pictures The test unit adapter can update the unit model with the topic of the resource identifier of the image The test unit adapter can connect to the test unit control The test unit adapter can read the test unique identifier of a new test in the test unit control The test unit adapter can capture a new image when the new test is detected The test unit adapter stores the image in a network file system in the field network The test unit adapter can publish a resource identifier for the image
Requirements filled	RQ_0295, RQ_0289, RQ_0290, RQ_0291, RQ_0292, RQ_0293, RQ_0296
ZA3.03.9 Show current test information	**Priority:** Must
	Who: Operator **When / Where:** When the operator is conducting the manual test **What:** User visualises the text image and the additional information of the manual test **Why:** To conduct the manual test correctly

Acceptance Criteria	The presentation backend can read the topics of the test identifier, the part number, and the image resource identifier The presentation backend can subscribe to these topics The user can visualize the reference image associated to the current test The user can visualize the camera image associated to the current test The user can visualize the partNumber The user can visualize the additional information parameters associated to the test (via test identifier)
Requirements filled	RQ_0294, RQ_0297, RQ_0300
ZA3.03.10 **Conduct manual test**	**Priority:** Must
	Who: Operator **When / Where:** When the operator is conducting the manual test **What:** Indicate manual test result **Why:** To conduct the manual test
Acceptance Criteria	The operator can push an actuator in the test unit to determine the result of the test
Requirements filled	RQ_0298, RQ_0299
ZA3.03.11 **Collect results**	**Priority:** Must
	Who: Test unit adapter **When / Where:** When the operator has indicated the result of the manual test **What:** Read the test result **Why:** To store the results
Acceptance Criteria	The operator can use an actuator to specify the test result The test unit adapter can read the result from the test unit control
Requirements filled	None
ZA3.03.12 **Store results**	**Priority:** Must
	Who: Application storage **When / Where:** When the operator has indicated the result of the manual test **What:** Store the results **Why:** For further analysis in other applications
Acceptance Criteria	The test unit adapter can store the results of the test The results include the test unique identifier The results include the product type The results include the resource identifier of the image
Requirements filled	RQ_0285, RQ_286, RQ_0301, RQ_302, RQ_0303, RQ_304

Figure 270zA3.03 Features

7.14.3 Workflows

7.14.3.1 Low level configuration

This workflow sets up the application configuration parameters that are more integrated into the ZDMP Platform, like data sources and connections to exchange data with manufacturing assets. The main steps are:

- Configure data sources for master data
- Configure the connection to the test unit controller
- Configure the resource location of the variables to be used to exchange information with the test unit controller

The workflow is analogous to the low-level configuration of zXRayMonitor in Figure 265.

7.14.3.2 Test unit sequence configuration

This workflow sets up the configuration of the tests to be conducted for every product type. The main steps are:

- Select product type
- Edit test sequence configuration
- Edit test configurations

The workflow is similar to the test unit sequence configuration workflow of zXRayMonitor depicted in Figure 266.

7.14.3.3 Operational data exchange

This workflow manages all data exchange with the test unit. The main steps are:

- Detect product type
- Load configuration parameters
- Show information to operator
- Collect tests results
- Store tests results

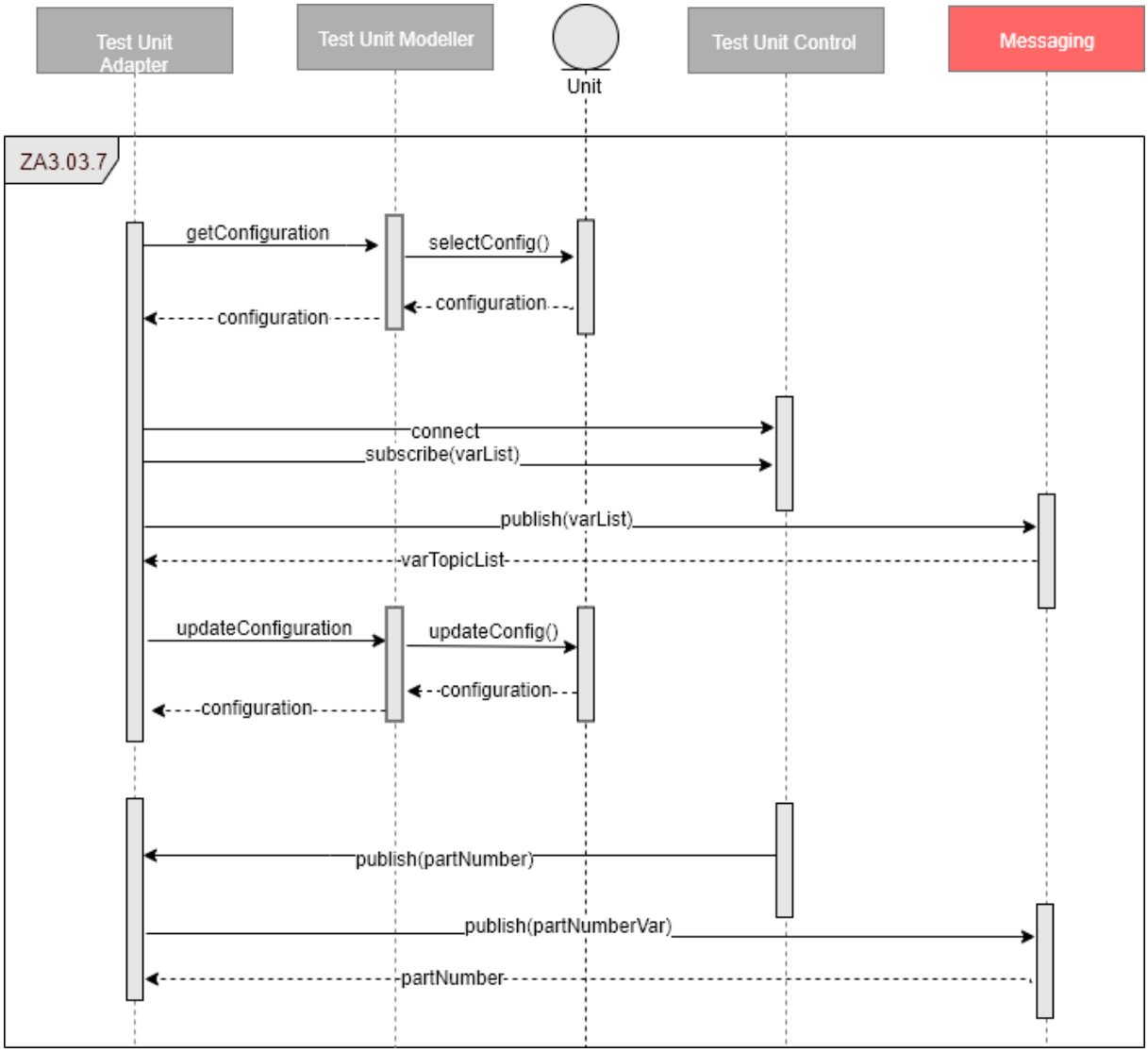


Figure 271: Product Type Detection Sequence Diagram

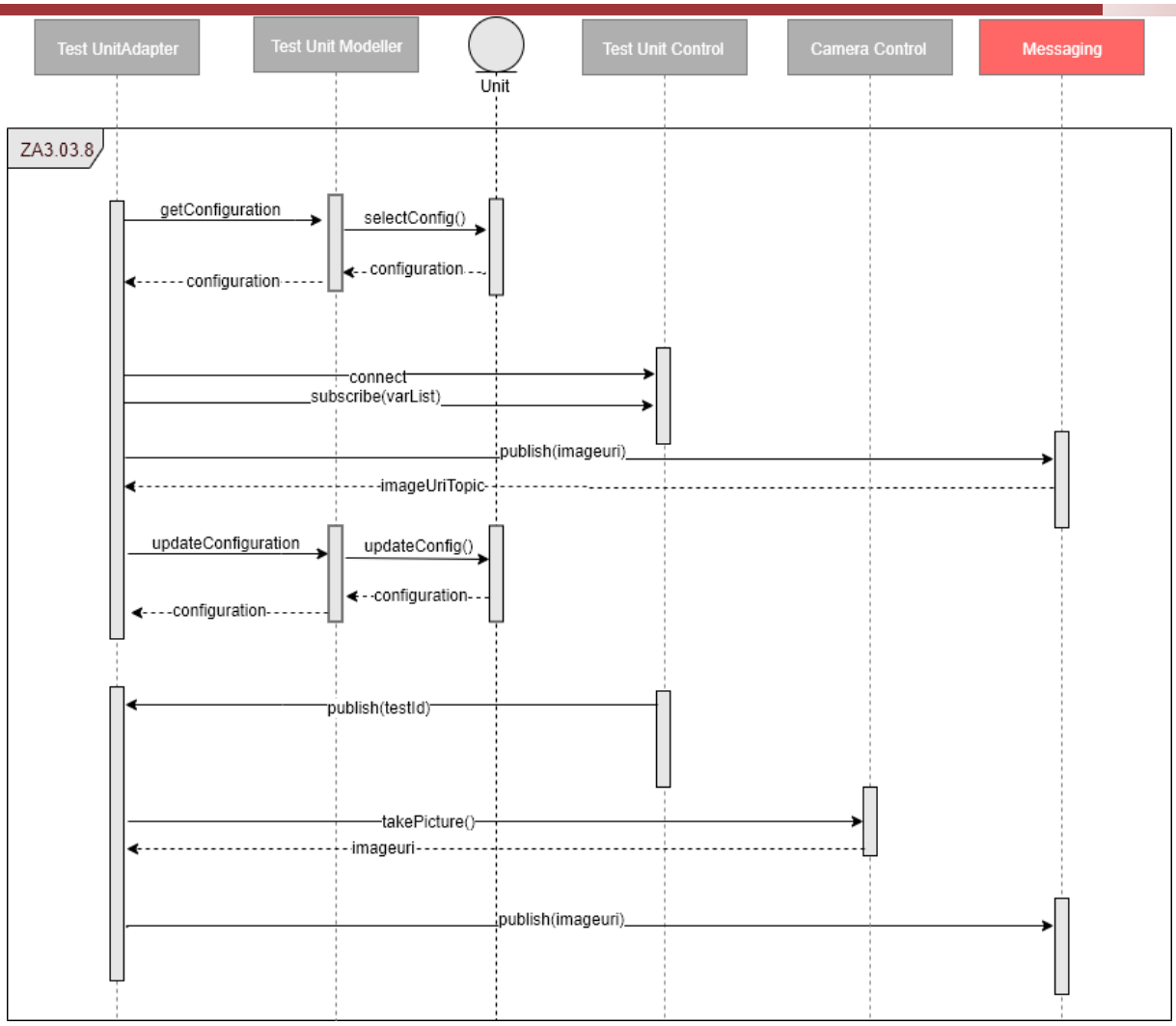


Figure 272: Test Identifier Detection Sequence Diagram

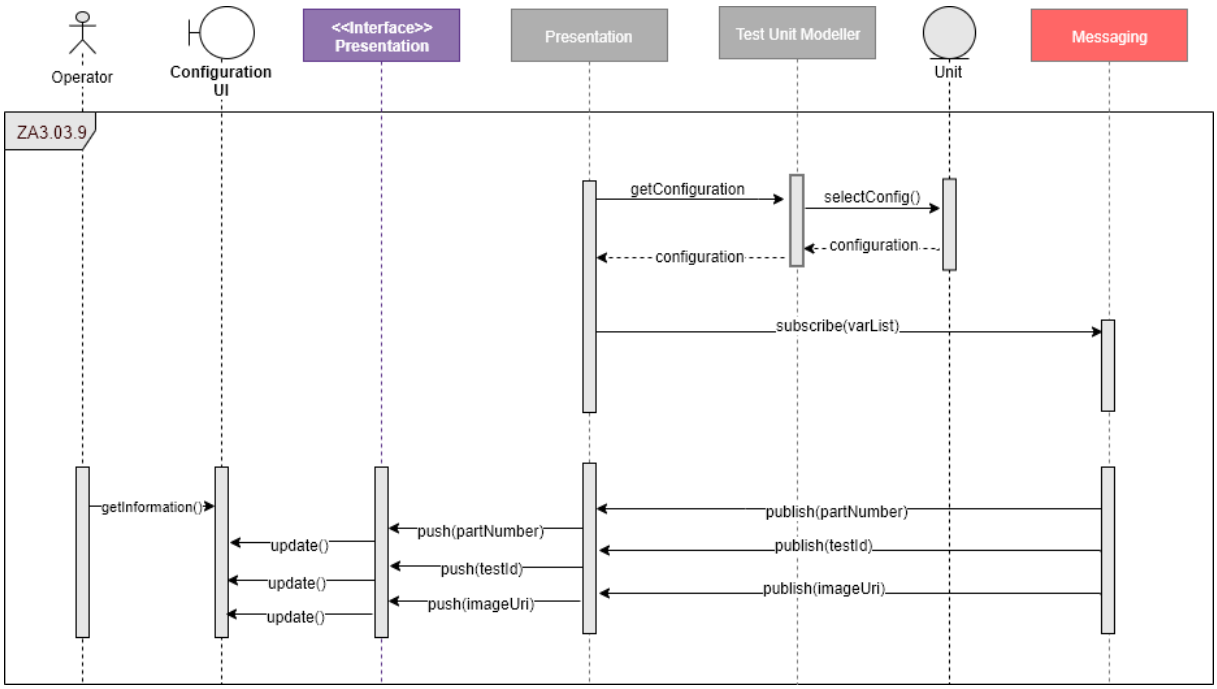


Figure 273: Current Test Presentation Sequence Diagram

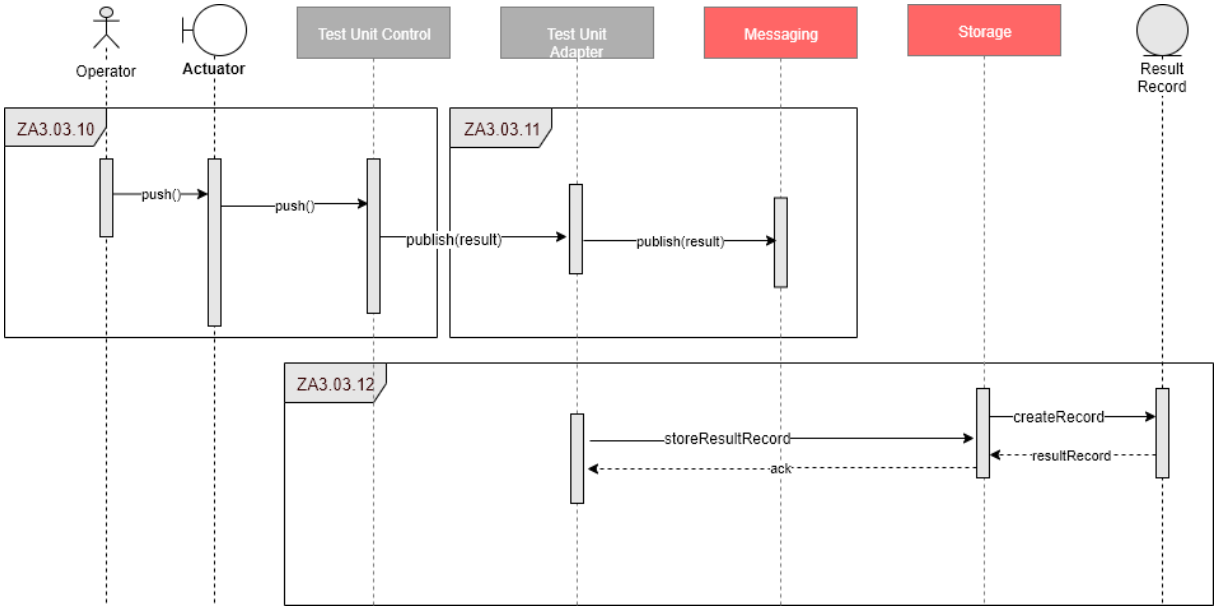


Figure 274: Result Collection and Storage Sequence Diagram

7.14.4 Additional Issues

Additional issues have appeared after the description of the functional specification.

Issue	Description	Next Steps	Lead (Rationale)
Filled Requirements	The following requirements with a “must“-priority were targeted at different platform components, but are specific to this application and thus partially filled by its functions RQ_0282, RQ_0283, RQ_0284, RQ_0285, RQ_0286, RQ_0287, RQ_0288, RQ_0289, RQ_0290, RQ_0291, RQ_0292, RQ_0293, RQ_0294, RQ_0295, RQ_0296, RQ_0297, RQ_0298, RQ_0299, RQ_0300, RQ_0301, RQ_0302, RQ_0303, RQ_0304	Discuss with requirement providers who to solve the issue	Product owner UPV

7.15 zArtificial IntelligenceAFT (zA3.06)

7.15.1 Overall functional characterization & Context

The zArtificialIntelligenceAFT helps test engineers fine tune the parameters of automatic tests. The test units rely on image processing algorithms to perform automatic tests and the parameters of these algorithms must be set accurately to minimise the occurrence of false automatic test results. zArtificialIntelligenceAFT correlates false automatic test results with the parameters of the algorithm, presents analytical reports to the test engineer, and suggests parameter adjustments to minimise the occurrence of false automatic test results, based on machine learning algorithms. False test results are detected using different available control loops: Feedback from the test engineer in the analysis station, feedback from the final manual test, and feedback from the customer acceptance tests. The zFeedbackAFT collects and stores the results of the tests and the feedback loops, thus providing the data used by the zArtificialIntelligenceAFT. The application provides backend interfaces to edit the configuration of the training data sources and the optimisation cost function.

7.15.2 Functions / Features

- **Optimisation configuration:** Backend function to edit some configuration parameters of the optimisation algorithm, like the location of the training data sources or weighting parameters to model the impact for the company of a false automatic test results at each feedback control loop (eg a false negative at the customer site is more expensive than a false positive detected at the analysis station).
- **Results analysis**: Statistical analysis functions and user interfaces to present the test engineer with historic data statistical analysis and trends of the measurement results.
- **Parameter set optimisation**: Optimisation functions based on machine learning to calculate the set of optimal parameters that minimise the probability of false automatic test results in the test system, based on the training data. The optimisation algorithm uses a cost function dependent of the number of false automatic test results at the different feedback loops.
- **Optimal parameter cost representation:** Graphic dashboards and user interfaces to present the test engineer with the difference between the current test unit configuration and the optimal configuration parameters. For every parameter, the user interfaces represent the cost function against the different values used in the training data.

These functions can be further decomposed into the following subtasks that have to be realised:

Subtask	Subtask description
ZA3.06.1 Configure data sources	**Priority:** Should
	Who: ZDMP consultant **When / Where:** During configuration (runtime), on premise **What:** User selects data sources **Why:** To configure data sources with the test records used by the optimisation algorithm
Acceptance Criteria	The user can create a new configuration

	The user can create a data source The user can select data sources that have been previously configured in the platform The user can update the configuration with the selected data source
Requirements filled	RQ_0328
ZA3.06.2 Configure optimisation algorithm	**Priority:** Should
	Who: ZDMP consultant **When / Where:** During configuration (runtime), on premise **What:** User configures additional configuration parameters of the optimisation algorithm **Why:** To better adapt to the results to the context of the company
Acceptance Criteria	The user can set some configurations of the optimisation algorithm The configuration is updated with the parameters provided by the user
Requirements filled	RQ_0329
ZA3.06.3 Analyse automatic test results	**Priority:** Should
	Who: Test engineer **When / Where:** After a positive (non-conformance) result, on the analysis station (on premise) **What:** Analyse the results of automatic tests **Why:** To obtain in-depth insight of the false automatic test results under analysis
Acceptance Criteria	The user can obtain graphic reports showing statistical analysis of historic data related to an automatic test result The user can analyse historic data based on different criteria The test engineer can set parameters to filter data and configure analytics Process quality can use test results records as inputs
Requirements filled	RQ_0330
ZA3.06.4 Check optimal parameter configuration	**Priority:** Must
	Who: Test Engineer **When / Where:** After a positive (non-conformance) result, on the analysis station (on premise) **What:** get optimal configuration parameters and compare with current configuration **Why:** To edit the configuration of the test after a false automatic test result
Acceptance Criteria	The user can check the optimal configuration parameters for every product type The user can compare the current configuration with the optimal configuration The user can compare optimal and current configuration parameters with the training data
Requirements filled	RQ_0327, RQ_0330

Figure 275: zA3.06 Features

7.15.3 Workflows

7.15.3.1 Optimisation configuration

This workflow provides access to some configuration parameters of the optimization algorithm. The main steps are:

- Edit training data sources
- Edit additional parameters

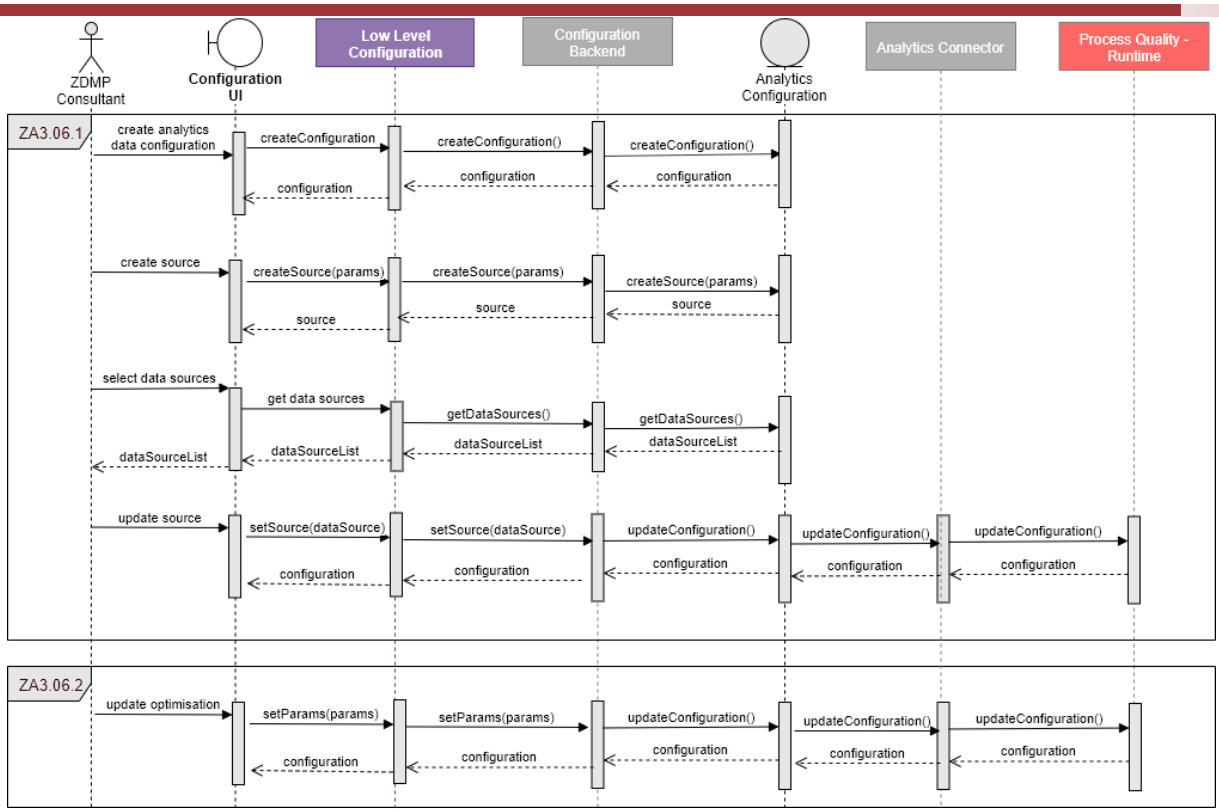


Figure 276: Optimisation Configuration Diagram

7.15.3.2 Result Analysis

This workflow provides insights on the training data and the configuration parameter optimization results. The main steps are:

- Analyse results
- Check optimal configuration parameters

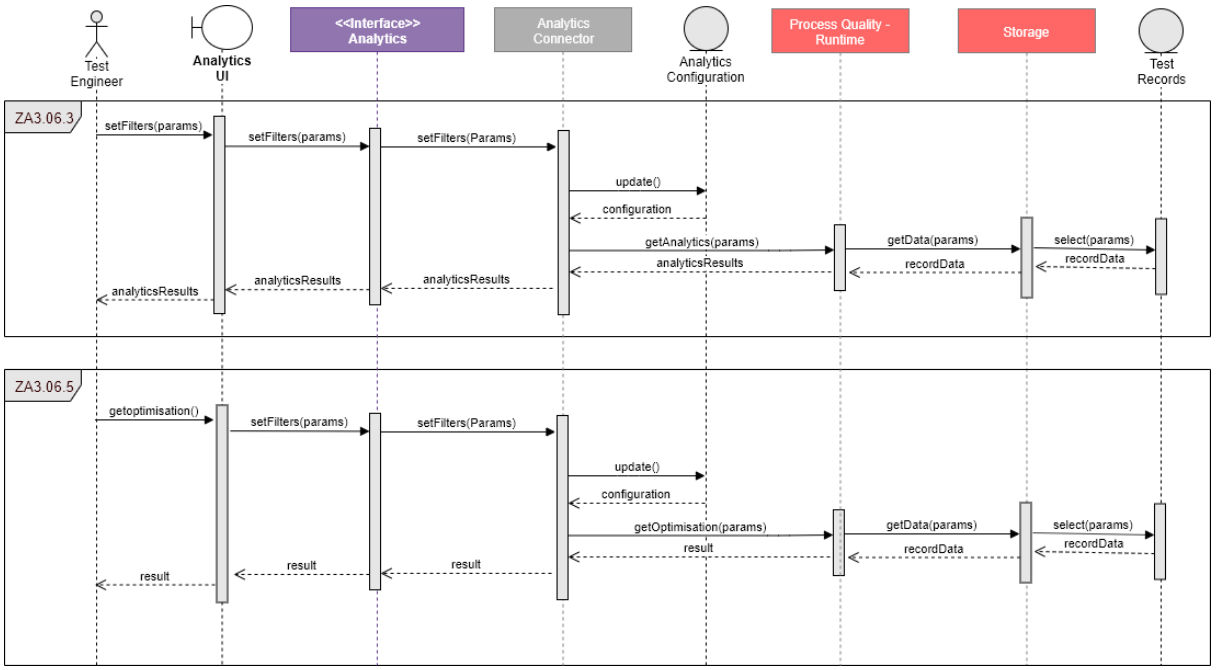


Figure 277: Result Analysis Sequence Diagram

7.15.4 Additional Issues

Additional issues have appeared after the description of the functional specification.

Issue	Description	Next Steps	Lead (Rationale)
Filled Requirements	The following requirements with a “must“-priority were targeted at different platform components, but are specific to this application and thus partially filled by its functions RQ_0327, RQ_0328, RQ_0330	Discuss with requirement providers who to solve the issue	Product owner UPV

7.16 zDriver & zLineData & zDataArchiveControl (zA3.07&3.08, 3.15)

7.16.1 Overall functional characterization & Context

zDriver, zLineData, and zLineControl aim to facilitate the connection and management of data exchange with the different workstations in the assembly line. Other applications (eg zPowerManager, zVisualManager and zCycleTimeManager) use these functions to send data (configuration data and commands) to and receive data from (manufacturing operations data) the components of the assembly line. For the sake of clarity, this section describes the three applications together. zDriver component manages the exchange of data with the different workspaces of the assembly line. The zDriver component runs in a line server and is connected to the controllers of the different workspaces to collect the necessary data. To comply with industrial security requirements, the zDriver implements a client application, eg based on the Message Queueing Telemetry Transport (MQTT) protocol, to connect to other components in the ZDMP Platform. Communications in the line use a proprietary Transport Control Protocol (TCP). zDriver is thus basically a gateway application providing backend interfaces to manage the modelling of the assembly line and the configuration of the connection to the different workspaces therein. Functions of zDriver are identified with ids ZA3.07 [1-4]. zLineData provides backend functionalities to model the assembly line and control the data workflows. Functions of zLineData are identified with ids ZA3.08 [1-6]. Finally, zDataArchiveControl implements data retention policies to reduce the volume of industrial data, using ZDMP storage component functions. Functions of zDataArchiveControl are identified with ids ZA3.15.1.

7.16.2 Functions / Features

- **Industrial data collection configuration:** Backend function to configure the connections to exchange data with the different workstations. The user can define different industrial variables. Industrial variables are used to read the (part serial number, workspace start (check-in) time, workspace finish (check-out) time, operation result, error information, machine status, material consumption, and material stock available at each workspace. They are also used to load information into the workspace control to optimise product changeover and minimise set-up time.
- **Assembly Line configuration:** Backend function to configure the organisation of the assembly line into workspaces. The user can define workspaces, assign them a

name to better identify them, and arrange then in serial or parallel configurations within the assembly line. For every workspace, the user can define the configuration of the data collection variables of each workstation.

These functions can be further decomposed into the following subtasks that have to be realised:

Subtask	Subtask description
ZA3.08.1 Configure storage	**Priority:** Should
	Who: ZDMP consultant **When / Where:** During configuration (runtime), on premise **What:** User configures data management and data retention policies **Why:** To set up application storage
Acceptance Criteria	The user can select data retention policies to maintain relevant time series data
Requirements filled	RQ_0331, RQ_0332, RQ_0333, RQ_079, RQ_0380, RQ_0382, RQ_0386
ZA3.08.2 Configure manufacturing data connection	**Priority:** Should
	Who: ZDMP consultant **When / Where:** During configuration (runtime), on premise **What:** User selects data sources (files, database connections) to integrate manufacturing data **Why:** To integrate master data and manufacturing execution system information
Acceptance Criteria	The user can create a new manufacturing data configuration The user can create a new data source The user can select data sources that have been previously configured in the platform The user can update a master data configuration with the selected data source Manufacturing data provides access to product type information Manufacturing data provides access to work order information Manufacturing data provides access to BoM information Manufacturing data provides access to operations information
Requirements filled	RQ_0331, RQ_0332, RQ_0333, RQ_0372,
ZA3.08.3 Configure test unit connection	**Priority:** Should
	Who: ZDMP consultant **When / Where:** During configuration (runtime), on premise **What:** User configures a connection to a workspace control **Why:** To enable data exchange
Acceptance Criteria	The user can create a new assembly line model The user can edit the connection parameters The user can select data connections that have been previously configured in the platform Optionally, the user could browse the field network to search for available connections
Requirements filled	RQ_0331 RQ_0332, RQ_0333
ZA3.08.4 Configure industrial variables	**Priority:** Should
	Who: ZDMP consultant **When / Where:** During configuration (runtime), on premise **What:** User configures industrial variables **Why:** To exchange data with the test unit control program
Acceptance Criteria	The user can select data sources that have been previously configured in the platform
Requirements filled	RQ_0331 RQ_0332, RQ_0333
ZA3.08.5 Edit assembly line configuration	**Priority:** Must
	Who: Production Engineer **When / Where:** During configuration (runtime), on premise

	What: User configures the assembly **Why:** To edit the configuration of the line
Acceptance Criteria	The user can create new workstations The user can organise the workstations in the line (workstation order and parallel or serial configuration)
Requirements filled	RQ_0331 RQ_0332, RQ_0333
ZA3.08.6 **Edit workstation configuration**	**Priority:** Must
	Who: Test Engineer **When / Where:** During configuration (runtime), on premise **What:** User edits the configuration of a workstation **Why:** To specify the variables used to exchange data with the controller
Acceptance Criteria	The user can edit the name of the workstation The user can edit the product-in variable to read the product type of a product entering the workstation The user can edit the product-out variable to read the product type of a product leaving the workstation The user can edit the status variable to read the status of the workstation The user can edit error variable(s) to read the occurrence of failures in the workstation The user can edit the variables to read the stock of components The user can edit the variables to read energy consumption The user can edit the variables to write automatic changeover parameters The user can edit the variables to write commands to switch on and off components
Requirements filled	RQ_0331, RQ_0332, RQ_0333
ZA3.07.1 **Detect product type**	**Priority:** Must
	Who: Assembly line data collection adapter **When / Where:** When a product enters or leaves a workstation (runtime), in the assembly line **What:** Read the part number of the product instance that enter or leaves the workstation **Why:** To detect which is the product type, load the right configuration parameters into the workstations, and measure the cycle time
Acceptance Criteria	The assembly line model contains the parameters to connect to the different workstation program logic controllers and exchange information The assembly line adapter can connect to the different workstation program logic controllers The assembly line adapter can publish data to exchange data with other components The assembly line adapter can update the assembly line model with the topics of the parameters to be exchanged The assembly line adapter is connected (TCP/IP) to the program logic controllers in the workstations The assembly line adapter can read the part number variable in the program logic controller and publish the part number code value in the configured product-in or product-out variable
Requirements filled	RQ_0334, RQ_0363
ZA3.07.2 **Load configuration parameters**	**Priority:** Must
	Who: Assembly line data collection adapter **When / Where:** Before the first product instance of a work order enters a workstation **What:** write the values of the automatic changeover parameters in the workstation program logic control **Why:** To configure the workstation for the product type
Acceptance Criteria	The assembly line adapter can get the variables and values to write for the product type and workstation from the assembly line model

	The assembly line adapter can write the values in the test unit control variables indicated in the configuration
Requirements filled	RQ_0354
ZA3.07.3 Production data collection	**Priority:** Must
	Who: Assembly line data collection adapter **When / Where:** When the workstation is processing a product instance (runtime), in the assembly line **What:** Read the data in the workstation **Why:** To know the status of the workstation, the material stock levels, energy consumption, alarms, etc in the workstation
Acceptance Criteria	The assembly line adapter can read the status variables The assembly line adapter can read error variable(s) The assembly line adapter can read the stock level variables The user can edit read energy consumption variables The assembly line adapter creates a production data record with the collected data The production data record contains the read variable values The production data record contains the product type The production data record contains manufacturing operations management data like check-in and check-out timestamps
Requirements filled	This subtask fills several requirements for zPowerManagement, zVisualManager and zCycleTimeManager
ZA3.07.4 Production data storage	**Priority:** Must
	Who: Application storage **When / Where:** When the production data is received from the workstation controller **What:** Store the production data records **Why:** For further analysis in other applications
Acceptance Criteria	The assembly line adapter can store production data records
Requirements filled	This subtask fills several requirements for zPowerManagement, zVisualManager and zCycleTimeManager
ZA3.15.1 Production data storage	**Priority:** Must
	Who: Application storage **When / Where:** According to the data retention policy configuration **What:** Delete or archive records **Why:** To reduce the volume of industrial data
Acceptance Criteria	The user can restore data
Requirements filled	RQ_0385, RQ_0386

Figure 278: zA3.07&3.08, 3.15 Functions

7.16.3 Workflows

7.16.3.1 Low level configuration

This workflow sets up the application configuration parameters that are more integrated into the ZDMP Platform, like data sources and connections to exchange data with manufacturing assets. The main steps are:

- Configure data retention policies
- Configure master data sources
- Configure the connection to the workspace controllers
- Configure the resource location of the variables that is used to exchange information with the workspace controllers

The workflow is analogous to the low-level configuration of zXRayMonitor in Figure 265.

7.16.3.2 Assembly line configuration

This workflow sets up the configuration of the model of the assembly line. The main steps are:

- Select assembly line
- Edit assembly line organisation configuration
- Edit workspace configurations

The workflow is similar to the test unit sequence configuration workflow of zXRayMonitor depicted in Figure 266.

7.16.3.3 Operational data exchange

This workflow manages all data exchange with the test unit. The main steps are:

- Detect product type
- Load configuration parameters
- Collect production data

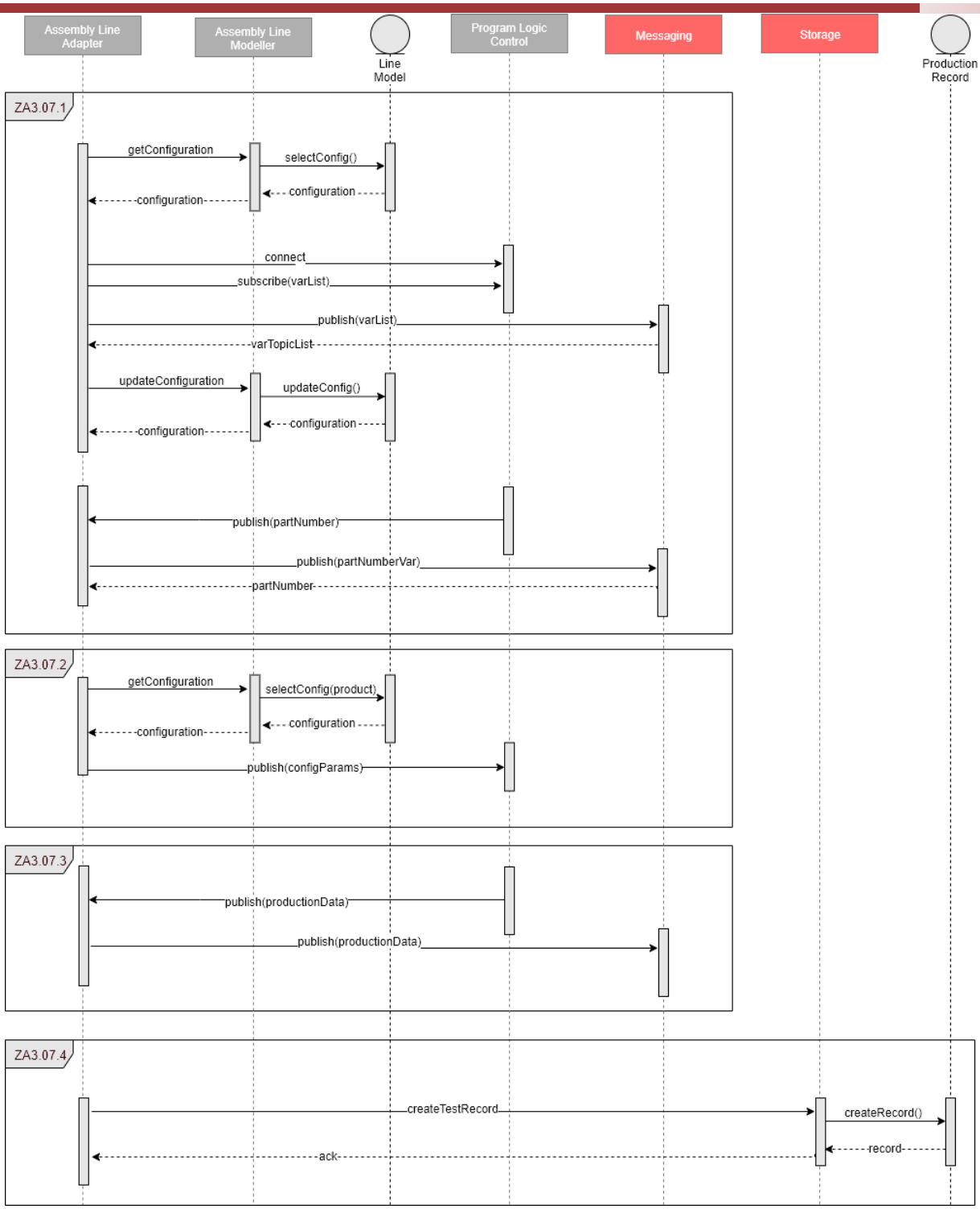


Figure 279: Operational Data Collection Sequence Diagram

7.16.4 Additional Issues

Additional issues have appeared after the description of the functional specification.

Issue	Description	Next Steps	Lead (Rationale)
Filled Requirements	The following requirements with a “must“-priority were targeted at different platform components, but are specific to this application and thus partially filled by its functions RQ_0333, RQ_0363, RQ_0382, RQ_0384, RQ_0385, RQ_0386	Discuss with requirement providers who to solve the issue	Product owner UPV

7.17 zVisualManager, zCycleTimeManager, zAutomaticCall, zPowerManagement (zA3.09, zA3.11, zA3.12, zA3.13)

7.17.1 Overall functional characterization & Context

zVisualManager is a backend application to configure visual graphic dashboards to show Key Process Indicators (KPIs) calculated using the assembly line data collected by the zDriver application, and events that are triggered eg when an indicator decreases below a given threshold, or when a workstation is in a given failure state. The user can also configure e-mail notifications to specific users (eg maintenance engineer) for every configured event. To do so, the application applies standard formulas and methods to obtain indicators from industrial data records. Functions of zVisualManager are identified with id ZA3.09.1. zCycleTimeManager uses this functions to calculate production KPIs like cycle time, availability, performance, quality, and Overall Equipment Efficiency (OEE), both per line and per workstation. Through the provided backend interfaces, the user can either set up objectives for every indicator or collect these objectives from manufacturing operations management data sources. zPowerManagement applies the same concept to power management, calculating Energy Performance Indicators (EnPIs) from the collected data. Common functions of zCycleTimeManager and zPowerManager are identified with ids ZA3.11 [1-2]. Finally, zAutomaticCall manages event processing and notifications. These functions are identified with id ZA3.12 .1.

7.17.2 Functions / Features

- **Configuration:** Backend functions to configure the application. The user can define the formulas to calculate the KPIs form the production data records. Industrial variables are used to read the (part serial number, workspace start (check-in) time, workspace finish (check-out) time, operation result, error information, machine status, material consumption, and material stock available at each workspace. They are also used to load information into the workspace control to optimise product changeover and minimise set-up time. The user can use different graph types to set up the visual dashboards and configure the notifications.
- **Production KPI calculation:** Calculation of cycle times, production KPIs and EnPIs from production data records using the configured formulas.
- **e-mail notifications:** Notifications to users according to the provided information.

These functions can be further decomposed into the following subtasks that have to be realised:

Subtask	Subtask description
ZA3.09.1	**Priority:** Should

Configuration	**Who:** ZDMP consultant **What:** User configures KPI calculation, dashboards, and notifications **Why:** To set up parameters for power management, efficiency management, as well as configure dashboards and notifications
Acceptance Criteria	The user can configure formulas using attributes of production data records The user can configure the dashboards used to show production KPIs and EnPIs The user can configure the rules to generate notifications and their behaviour The user can implement energy efficiency strategies based on EnPIs, dashboards, rules, and notifications
Requirements filled	RQ_0340, RQ_0341, RQ_0348, RQ_0350, RQ_0351, RQ_0352, RQ_0353, RQ_0359, RQ_0361, RQ_0362
ZA3.12.1 KPIs Visualisation	**Priority:** Must
	Who: Production Engineer **When / Where:** When a new production data record is published (runtime), in the assembly line **What:** Visualises graphic dashboards showing the calculated KPIs **Why:** To evaluate the performance of the assembly line and workstations
Acceptance Criteria	The monitoring and alerting component can subscribe to receive production data records The monitoring and alerting component calculates KPIs when a new production data record is published The user can visualise the KPIs in the configured dashboards The dashboards are updated with every production data record
Requirements filled	RQ_0348
ZA3.12.2 Calculate Indicators	**Priority:** Must
	Who: Production Engineer **When / Where:** When a new production data record is published (runtime), in the assembly line **What:** Creates indicators where the KPI data is trending **Why:** To forecast the performance of the assembly line and workstations
Acceptance Criteria	The application calculates OEE indicator and components (performance, availability, quality) The application calculates energy efficiency indicators (consumption per production unit)
Requirements filled	RQ_0365, RQ_0366, RQ_0367, RQ_0369
ZA3.12.1 Receive notification	**Priority:** Must
	Who: Production Engineer **When / Where:** When a new production notification is published (runtime), in the assembly line **What:** User receives a notification **Why:** To take immediate actions and contain the problem
Acceptance Criteria	The user receives a notification according to the configured rules The user can receive e-mail notifications.
Requirements filled	RQ_0343, RQ_0344, RQ_0345, RQ_0347, RQ_0349

Figure 280: zA3.09, zA3.11, zA3.12, zA3.13 Functions

7.17.3 Workflows

7.17.3.1 Configuration

This workflow sets up the configuration of the application. The main steps are:

- Configure formulas
- Configure dashboards
- Configure notifications

The workflow is analogous to the low-level configuration of zXRayMonitor in Figure 265.

7.17.3.2 Operational data exchange

This workflow manages all data exchange with the Production Engineer unit. The main steps are:

- Receive production data records
- Calculate KPIs
- Update dashboards
- Send notifications

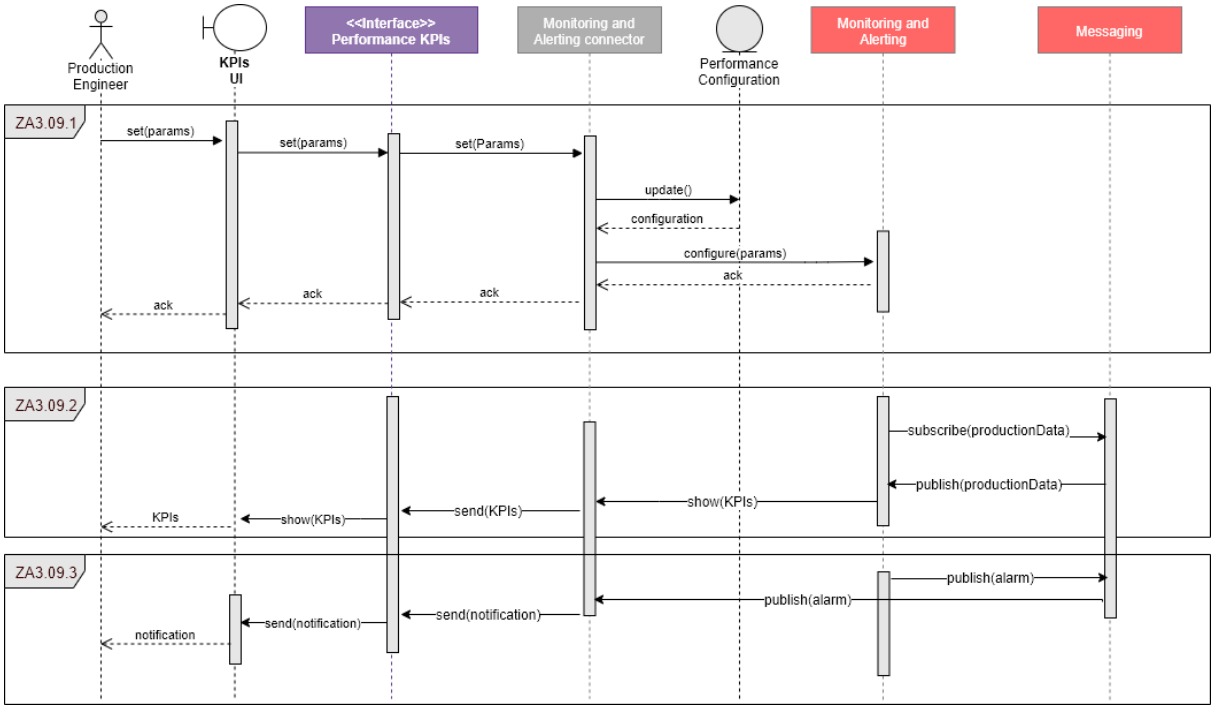


Figure 281: Operational Data Collection Sequence Diagram

7.17.4 Additional Issues

Additional issues have appeared after the description of the functional specification.

Issue	Description	Next Steps	Lead (Rationale)
Filled Requirements	The following requirements with a “must“-priority were targeted at different platform components, but are specific to this application and thus partially filled by its functions RQ_0343, RQ_0344, RQ_0345, RQ_0347, RQ_0349, RQ_0365, RQ_0366, RQ_0367, RQ_0369	Discuss with requirement providers who to solve the issue	Product owner UPV

7.18 zProductVersionControl, zAutomaticMaterialOrdering (zA3.10, zA3.14)

7.18.1 Overall functional characterization & Context

zProductVersionControl supports product changeover through the exchange of data with both machines (workstation control) and humans (operator). The operator indicates which is the next work order for the assembly line, by selecting one order from the planned work order sequence. The detection of an instance of the new product type in every workspace, via the zDriver application, triggers a product changeover event that is used to forward relevant information to optimise product changeover and ensure product quality. The layout of the workstations and the sequence of operations is obtained from the assembly line model and the integrated manufacturing operations data. Functions of zProductVersionControl are identified with ids ZA3.10.1-3. zAutomaticMaterialOrdering checks material stock levels at every workspace and generates picking orders for the materials needed to avoid shortage. Operators get detailed instructions on the sequence of operations needed to complete the work order. Data exchange with machine control is used to automate the changeover process as much as possible, setting up certain parameters of the workspace control to minimise changeover time and process variability. When the stock levels go below pre-established levels, the application generates an automatic picking order to prevent stock breaks. zAutomaticMaterialOrdering functions are identified with ids ZA3.14. 1-2.

7.18.2 Functions / Features

- **Changeover control:** Start the changeover process in a workstation. The work order is set manually by the operator in the first workstation. Based on the manufacturing operations information, the application sets the next work order for the next workstations. When the application detects that the first unit of the next work order is entering a workstation, the application writes the configured changeover parameters into the workstation programming logic control
- **Operator dashboards:** Dashboards containing information to present to the operator with detailed information about the status of the current work order and the next work order

These functions can be further decomposed into the following subtasks that have to be realised:

Subtask	Subtask description
ZA3.10.1 Work order start	**Priority:** Must
	Who: Operator **What:** Confirm the next work order **When / Where:** When a new work order starts, in the first workstation of the assembly line **Why:** To start a work order
Acceptance Criteria	The operator can select the work orders from a list of planned work orders The list of planned work orders is read from the integrated manufacturing operations data sources
Requirements filled	RQ_0337
ZA3.10.2	**Priority:** Must
	Who: Operator

Changeover parameters write	**What:** Publish the changeover parameters of a workstation **When / Where:** When a new work order starts, in any workstation of the assembly line **Why:** To evaluate the performance of the assembly line and workstations
Acceptance Criteria	The assembly line adapter is subscribed to receive changeover parameter variable values The application is subscribed to receive product-in variable values from the assembly line adapter The assembly line adapter publishes the product type of the next work order The application publishes the values for the changeover parameters of the next order
Requirements filled	RQ_0335, RQ_0336, RQ_0338, RQ_0339,
ZA3.10.3 Visualise changeover dashboards	**Priority:** Must
	Who: Operator **What:** visualise updated information about the current work order and the next work order **When / Where:** When a new production record is published, in any workstation of the assembly line **Why:** To optimise operations and resource allocation
Acceptance Criteria	The user can visualise instructions to perform the operation of the current work order in the current workstation The user can visualise the status of the next work order in the previous workstations The user can visualise the expected time of arrival of the first product instance of the next workstation
Requirements filled	RQ_0337
ZA3.14.1 Check stock levels	**Priority:** Must
	Who: Operator **What:** Check the stock levels for components and materials **When / Where:** When a new production record is published, in any workstation of the assembly line **Why:** To optimise operations and resource allocation
Acceptance Criteria	The user can check the stock levels for the different components of the product type of the current work order in the workstation
Requirements filled	RQ_0337, RQ_0374
ZA3.14.2 Send automatic material ordering	**Priority:** Must
	Who: Picking operator **What:** User receives a notification **When / Where:** When the stock level of components or materials are not enough to complete the work order, in any workstation of the assembly line **Why:** To prevent stock break take immediate actions and contain the problem
Acceptance Criteria	The user receives a notification according to the configured rules The user can receive e-mail notifications.
Requirements filled	RQ_0373, RQ_0375, RQ_0376, RQ_0378

Figure 282: zA3.10, zA3.14 Functions

7.18.3 Workflows

7.18.3.1 Operational data exchange

This workflow manages all data exchange with the Production Engineer unit. The main steps are:

- Start work order

- Configure workstation
- Update information
- Send picking orders for materials and components

7.18.4 Additional Issues

Additional issues have appeared after the description of the functional specification.

Issue	Description	Next Steps	Lead (Rationale)
Filled Requirements	The following requirements with a “must“-priority were targeted at different platform components, but are specific to this application and thus partially filled by its functions RQ_0335, RQ_0336, RQ_0338, RQ_0373, RQ_0376	Discuss with requirement providers who to solve the issue	Product owner UPV

7.19 zArtificial IntelligenceMFT (zA3.04)

7.19.1 Overall functional characterization & Context

The zArtificialIntelligenceMFT application helps test engineers in order to reduce the quality incidents in parallel with a more secure and efficient optical inspection process. The test units consist of image processing algorithms to perform manual tests and the parameters of these algorithms must be set accurately to minimise the occurrence of false manual test results. The application provides backend interfaces to edit the configuration of the training data sources and the optimisation cost function.

7.19.2 Functions / Features

- **Optimisation configuration:** this function is used to edit some configuration parameters of the optimisation algorithm, like the location of the training data sources or weighting parameters to model the impact for the company of a false automatic test results at each feedback control loop.
- **Results analysis**: this feature presents the historic data statistical analysis and trends of the measurement results to test engineer.
- **Parameter set optimisation:** based on machine learning algorithms there are optimization functions that calculate the set of optimal parameters for minimising the probability of false manual test results in the test system, based on the training data.
- **Optimal parameter cost representation:** this functionality consists of graphic dashboards and user interfaces to present the difference between the current test unit configuration and the optimal configuration parameters to the test engineer.

These functions can be further decomposed into the following subtasks that have to be realised:

Subtask	Subtask description
ZA3.04.1 Configure data sources	**Priority:** Should
	Who: ZDMP consultant **When / Where:** During configuration (runtime), on premise **What:** User selects data sources **Why:** To configure data sources with the test records used by the optimisation algorithm
Acceptance Criteria	The user can create a new configuration, a new data source, also select / update a data source.
Requirements filled	RQ_0307
ZA3.04.2 Configure optimisation algorithm	**Priority:** Should
	Who: ZDMP consultant **When / Where:** During configuration (runtime), on premise **What:** User configures additional configuration parameters of the optimisation algorithm **Why:** To better adapt to the results to the context of the company
Acceptance Criteria	The user can set some configurations of the optimisation algorithm The configuration is updated with the parameters provided by the user
Requirements filled	RQ_0306
ZA3.04.3 Analyse automatic test results	**Priority:** Should
	Who: Test engineer **When / Where:** After a positive (non-conformance) result, on the analysis station (on premise)

	What: Analyse the results of automatic tests **Why:** To obtain in-depth insight of the false automatic test results under analysis
Acceptance Criteria	The user can obtain graphic reports showing statistical analysis of historic data related to an automatic test result The user can analyse historic data based on different criteria The test engineer can set parameters to filter data and configure analytics Process quality can use test results records as inputs
Requirements filled	RQ_0305
ZA3.04.4 Check optimal parameter configuration	**Priority:** Must
	Who: Test Engineer **When / Where:** After a positive (non-conformance) result, on the analysis station (on premise) **What:** get optimal configuration parameters and compare with current configuration **Why:** To edit the configuration of the test after a false automatic test result
Acceptance Criteria	The user can check the optimal configuration parameters for every product type The user can compare the current configuration with the optimal configuration The user can compare optimal and current configuration parameters with the training data
Requirements filled	RQ_0308

Figure 283: zA3.04 Functions

7.19.3 Workflows

The following sub-sections describe the sequence diagrams of the zArtificialIntelligenceMFT component.

7.19.3.1 Optimisation configuration

This workflow provides access to some configuration parameters of the optimization algorithm. The functions shown are:

- Edit training data sources
- Edit additional parameters

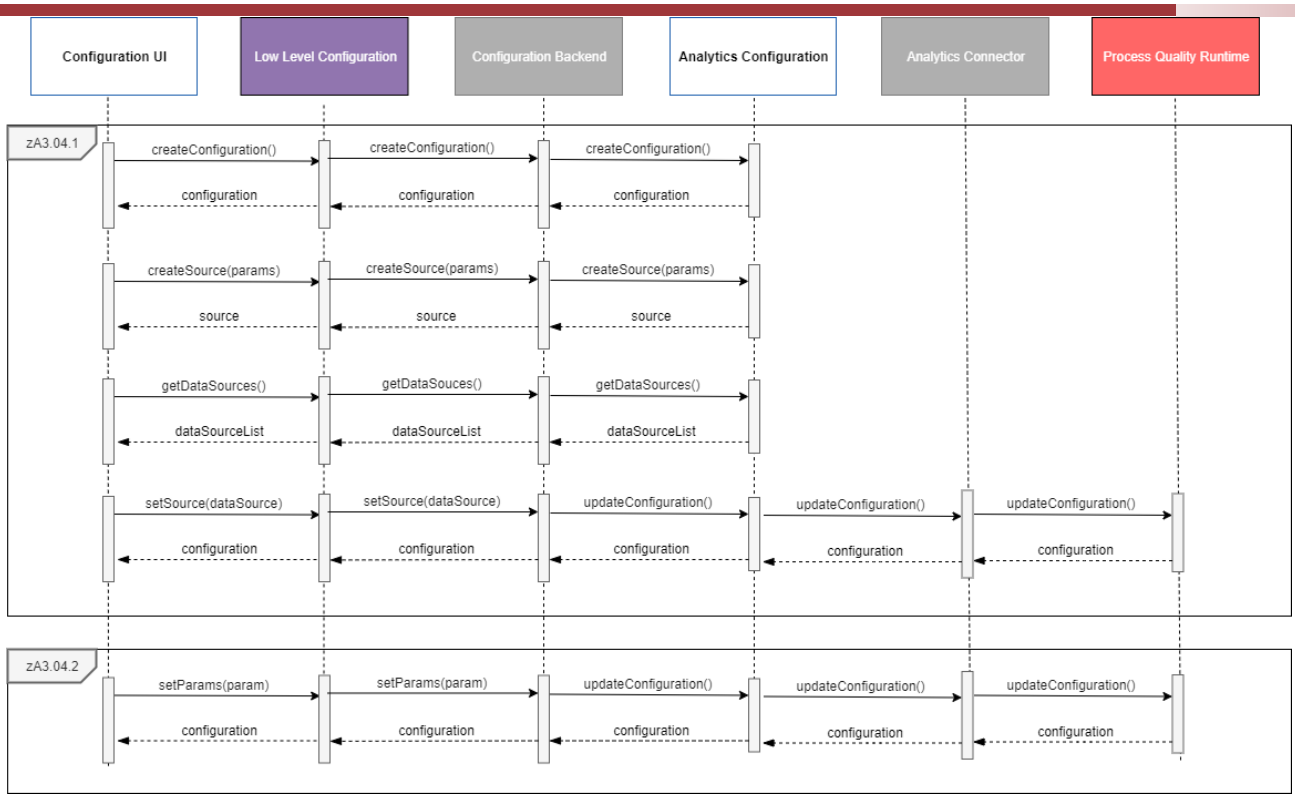


Figure 284: Optimization Configuration Sequence Diagram

7.19.3.2 Result Analysis

This workflow ilustrates flows on the training data and the configuration parameter optimization results. The functions displayed are:

- Analyse results
- Check optimal configuration parameters

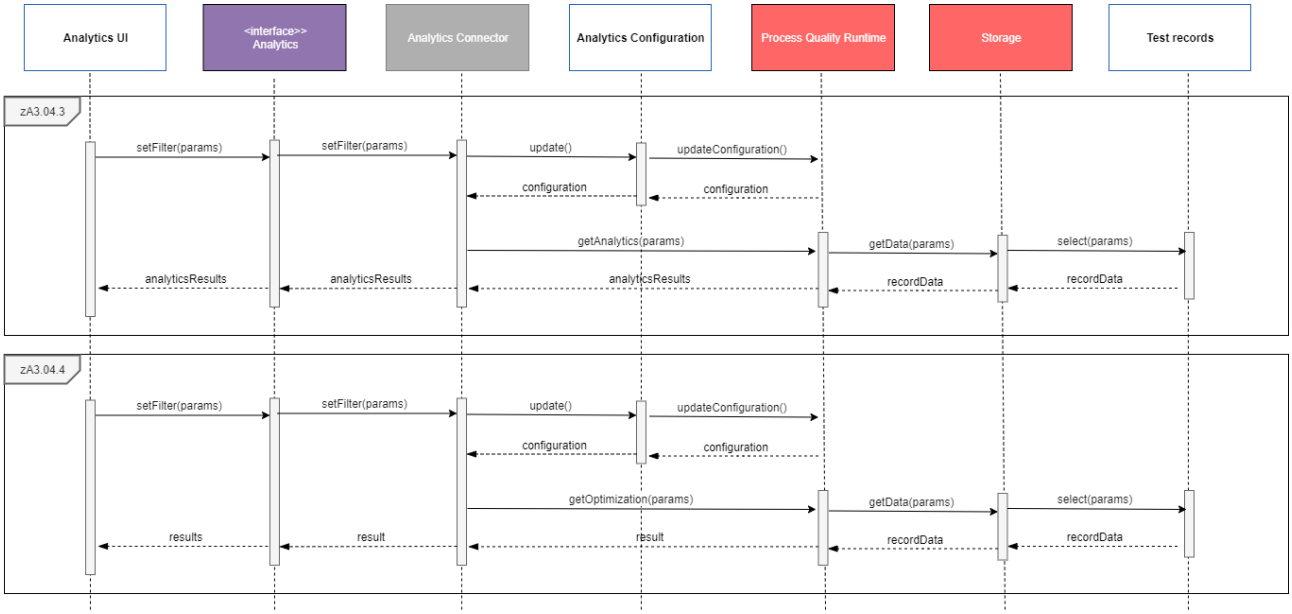


Figure 285: Result Analysis Sequence Diagram

7.20 zFeedbackAFT (zA3.05)

7.20.1 Overall functional characterization & Context

The zFeedbackAFT application assures the interface with Automatic Final Test, correlates the test image, tests results, and increases the quality of test result. The test unit uses a reference image and additional information describing the status of the product under test when there are no failures. This application has an user interface that shows an image of the product in the test unit taken with an external vision interface.

7.20.2 Functions / Features

- **Data sources configuration:** this function is used to configure the data sources used to integrate master data. Master data includes product type information, test information, and reference images.
- **Industrial data collection configuration:** this functionality is dealing with configation of the connections to exchange data with the test unit control. The user can define different industrial variables. Industrial variables are used to read the product instance part number, the current test, and the automatic test results.
- **Product type configuration:** this backend function edits the properties and test sequence for a specific product type. The user can select the product type and edit the test sequence. For every test, the user can edit the configuration parameters of the test. The configuration of a test consists industrial variables used to collect the results.
- **Part number detection:** this is to detect the product type unique identifier of the product under test.
- **Test detection**: this function detects the test being performed in the test unit
- **Test operator support:** this functionality is used to present the information of the test to the operator in a friendly user interface.
- **Test unit results collection:** this is to collect and store the results of the automatic test unit.

These functions can be further decomposed into the following subtasks that have to be realised:

Subtask	Subtask description
ZA3.05.1 Configure master data sources	**Priority:** Should
	Who: ZDMP consultant **When / Where:** During configuration (runtime), on premise **What:** User selects data sources (files, database connections) to integrate master data **Why:** To integrate master data information
Acceptance Criteria	The user can create a new master data configuration and a new data source The user can select data sources that have been previously configured in the platform The user can update a master data configuration with the selected data source
Requirements filled	RQ_0321
ZA3.05.2 Configure test unit and camera connection	**Priority:** Should
	Who: ZDMP consultant **When / Where:** During configuration (runtime), on premise **What:** User configures a connection to the test unit control **Why:** To enable data exchange

Acceptance Criteria	The user can create a new unit model The user can edit the connection parameters
Requirements filled	RQ_0325
ZA3.05.3 Configure industrial variables	**Priority:** Should
	Who: ZDMP consultant **When / Where:** During configuration (runtime), on premise **What:** User configures industrial variables **Why:** To exchange data with the test unit control program
Acceptance Criteria	The user can create a new industrial variable The user can specify the necessary parameters to read and write data from the variable
Requirements filled	RQ_0319
ZA3.05.4 Result test collection	**Priority:** Must
	Who: Test unit adapter **When / Where:** During optical tests (runtime), on premise **What:** For every test, the test unit data aquisition adapter must collect the unique code of the product under test, the results of the test, and the configuration parameters used by the unit test automation program **Why:** To detect false positives and false negatives results.
Acceptance Criteria	Product codes, configuration parameters and test results collected.
Requirements filled	RQ_0322
ZA3.05.5 Result test storage	**Priority:** Must
	Who: Apllication Storage **When / Where:** During optical tests (runtime), on premise **What:** zFeedBackAFT must store the tests results, linked to the unit test identifier, the product under test unique identifier, and the configuration parameters used **Why:** To perform future analysis
Acceptance Criteria	Product identified and configuration data stored
Requirements filled	RQ_0323
ZA3.05.6 Analysis results collection	**Priority:** Must
	Who: Test engineer **When / Where:** During optical tests (runtime), on premise **What:** For every positive (failure detected) result, the zFeedbackAFT application needs to collect the result of the analysis in the analysis station **Why:** To improve the quality of future tests
Acceptance Criteria	Analysis result collected
Requirements filled	RQ_0326

Figure 286: zA3.05 Functions

7.20.3 Workflows

7.20.3.1 Low level configuration

This workflow sets up the application configuration parameters that are more integrated into the ZDMP Platform, like data sources and connections to exchange data with manufacturing assets. The main steps are:

- Configure data sources for master data
- Configure test unit and camera connection
- Configure industrial variables

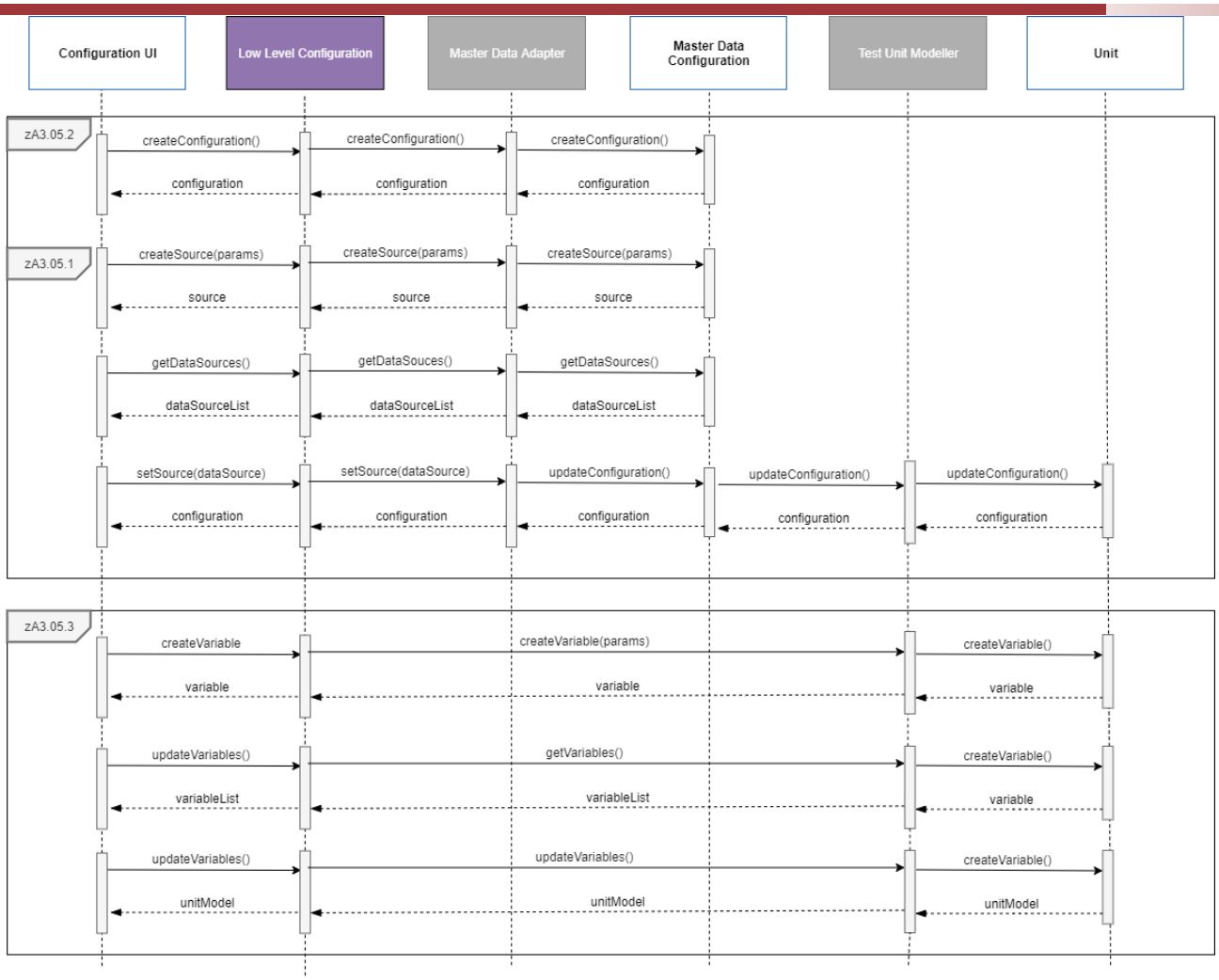


Figure 287: Low level configuration Sequence Diagram

7.20.3.2 Operational Data Exchange

This workflow provides access to some configuration parameters of the optimization algorithm. The functions shown are:

- Result test collection
- Result test storage
  Results collection analysis

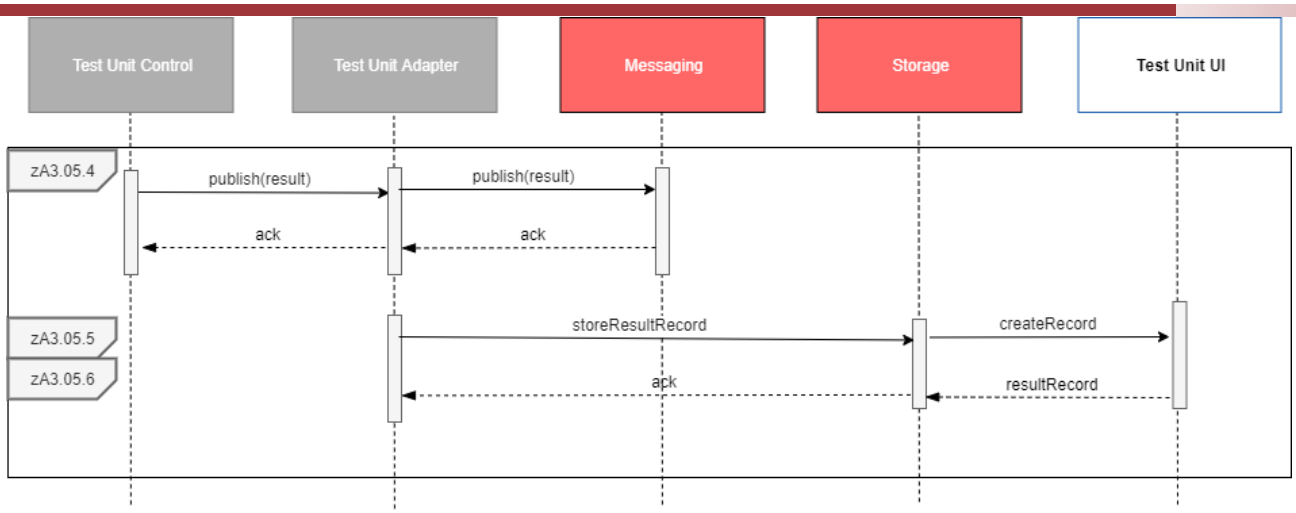


Figure 288: Operational Data Exchange Sequence Diagram

8 Conclusions

The current document represents a considerable amount of specification work by the technical partners that went into understanding the requirements presented by T4.1 and T4.2 and reflect on the single functions that would need implementation as well as the communication necessary to make the functions work out.

The state of specifications within this document was based of the subcomponents as defined by D4.3.1 Architecture Specification and was only expanded on in the components presented in WP7 and WP8.

The presented zApps that will be created within WP9 and WP10 were not planned to be defined within specifications, but as they present the missing link between the use cases and the base platform components of WP5-6 and the process and product quality specific components of WP7-8, special care was taken to also define the zApps in more detail and present their functions.

The target of this document was in part to analyse the requirements and the fulfilment of those by the respective functions of the presented components. As the level of detail in the requirements document as well as the presented detail in this document was by far exceeding the expectations of the task lead of the functional specification, the discussion and necessary changed in plans in order to fill all currently unfilled requirements will have to be continued in the following months, as well as reassigning zApp-specific requirements from base components to specific zApps, which were not considered in the requirements definition as all requirements were only targeted towards the base components.

To enable doing this work soon, all the requirements filled by the single modules and zApps were collected in the respective functions' tables, and all not filled requirements that were targeted at a certain task were put into a specific table.

The main idea of this Functional Specification was to make component-, module- and zApp-creators aware of the expectations towards their software from the users and the technical partners. Even though a very heavy document was produced in this way, much detail about single functions was collected and, in the course, many fruitful discussions were started and concluded in the making of this document. Much could still be refined and more closely aligned and analysed, but the consortium is now moving towards defining interfaces for the Technical Specification. These interfaces will be the definition the developers can work against in the system and will be subject to change and therefore be defined in an online web system based on Slate, Gitlab and Swagger.

Annex A: History

Document History	

Versions	• V1.01:
	  • PM Reviewed version and for EU Submission
	• V1.02-3:
	  • ASC minor fixes, updated logos, updated two sections

Contributions	ICE: • Philip Usher • Stuart Campbell • James Tryand • Ann Campbell SAG: • Martin Hess • Marc Dorchain SIVECO: • Mircea Vasile CET: • Ernesto Bedrina • Sandra Vilaplana • Juan Pardo VSYS: • Alessandro Liani • Mauro Fabrizioli ASC: • Tim Dellas • Nargis Tahara • Laura Caroline PROF: • Daniela Kirchberger • Baghbanpourasl Amirreza SOFT: • Christian Melchiorre • Lorenzo Cesario ROOT: • Alvaro Moretón Poch ITI: • Santiago Cáceres Elvira • Paco Velverde UPV: • Francisco Fraile IKER: • Marc Barceló • Urko Leturiondo • Oscar Salgado UOS-ITI: • Juri Papay UNIN: • Carlos Lopes TUT: • Ronal Bejarano

ZERO DEFECTS
Manufacturing
Platform
ZDMP

www.zdmp.eu