ZDMP: Zero Defects Manufacturing Platform

ZERO DEFECTS Manufacturing Platform ZDMP

WP1: Management: Procedures, Metrics, Coordination, and Reporting

D006: Technical Management - Overview Report - Vs: 1.0.5A

ZDMP ID: D1.4.1a

Deliverable Lead and Editor: Barrena Francisco, ITI

Contributing Partners: ITI, ICE

Date: 2020-06

Dissemination: Public

Status: EU Approved

Abstract

This technical management overview report gives a high-level view of the degree of completion of the development activities of ZDMP project including software security and quality management. It also provides an overview of the methodology followed to produce the software, and comprehensive guidelines to developers to integrate components in the ZDMP platform.

Grant Agreement: 825631

Document Status

Deliverable Lead	Barrena Francisco, ITI
Internal Reviewer 1	Fraile Francisco, UPV
Internal Reviewer 2	Gabilondo Juan José, ETXE
Internal Reviewer 3	Stuart Campbell, ICE
Type	Deliverable
Work Package	WP1: Management: Procedures, Metrics, Coordination, and Reporting
ID	D006: Technical Management - Overview Report
Due Date	2020-06
Delivery Date	2020-06
Status	EU Approved

Project Partners:

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

Executive Summary

The purpose of this report is to give a general view of the development activity. It is accompanied by specific report on the technical work, namely:

- D009 Technical Management: WP5 Report
- D012 Technical Management: WP6 Report
- D015 Technical Management: WP7 Report
- D018 Technical Management: WP8 Report

Furthermore, online documentation is made available publicly with complementary information to these reports (software.zdmp.eu), such as per component: Surce code, latest release, APIs, features, installation, minimum requirements for use, and how to run.

The current report provides a general overview of the methodologies followed in the project to achieve its objectives, and it gives a comprehensive high level view on the work carried out at ZDMP components until M18 (June 2020).

The document gives insights about the requirements coverage achieved, activity statistics extracted directly from the ZDMP GitLab, quality indicators, risks and data issues and several considerations about the integration of the different components.

Note that the development of zApps has not yet started and their development will be added in the updates of this document.

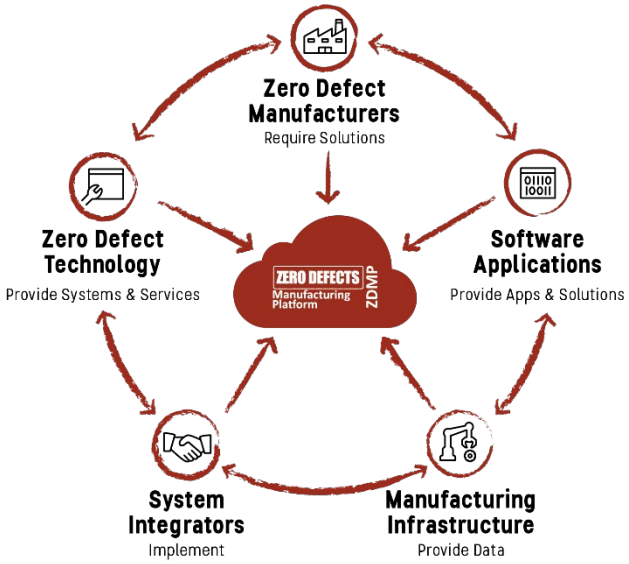
Table of Contents

0 Introduction 1
1 ZDMP Software Development 5
1.1 Reporting 5
1.2 Approach 7
1.2.1 Methodology 7
1.2.2 Source Code Management 8
1.2.3 Continuous Integration 9
1.2.4 Issue Tracking 10
1.2.5 System Deployment with Docker 10
1.2.6 Software Design Quality Analysis and Evaluation 11
1.2.7 Software Construction Quality 11
1.3 Reviewer's comments 12
2 Components Integration 14
2.1 Commons 14
2.2 Component orchestration 14
2.2.1 Monorepository flavour 15
2.2.2 Multirepository flavour 16
2.2.3 Proprietary code or external repository flavour 17
2.2.4 Mixed flavour 17
2.3 Platform orchestration 20
3 Components Overview 21
3.1 M18 Report 21
3.1.1 Functional Requirements Status 21
3.1.2 Activity Statistics in GitLab 21
3.1.3 Software Quality 25
3.1.4 Security Assessment 31
3.1.5 Risk and Data Issues 40
4 Conclusions 41

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 will allow 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 optimize the overall processes regarding time constraints and product quality, the achieving the zero defects.

0.2 Deliverable Purpose and Scope

The purpose of this document “D006 Technical Management – Overview report” is to clearly show the management foundations to produce software efficiently and according to purpose. The report contains information about the software development methodology

which is being followed to produce the different components of the ZDMP platform. It takes into account common aspects of all pieces in terms of requirements coverage, security, quality assessment and integration issues, giving to the reader an overview on how these had been tackled and to what extent these have been successfully covered.

Specifically, the DOA states the following regarding this Deliverable:

T1.4	**Technical Management: Leadership and Technical Reporting**			**ITI**	**M4-36**
D006 D007 D008	Technical Management: Overview Report	R	PU	18, (24), 30, (36), (42) 48	RDI3, 5, 8
This task handles the two primary management aspects of ZDMP at the RTD level: Technical research (specification, development, etc) and scientific research. Although as an IA, Scientific Research is extremely limited, it is expected several papers will be produced. • Research management will ensure both the efficiency of the research in and out of ZDMP and to enable influencing, knowledge exchange, and RTD dissemination to society. RTD management will thus ensure the uptake and application of scientific evidence, paper production, conference presentations, etc. and relate to the Scientific/Research workshop. • On the technical side, the software development process must be managed and is concerned primarily with the production aspect of software development, as opposed to the technical aspect, such as software tools which are encompassed in Task 1.5. These processes exist primarily for supporting the management of software development and are generally skewed towards addressing business concerns. Thus, the Technical Manager and all RDI leads have time allocation to do this.					

0.3 Target Audience

The main target of this document is firstly the ZDMP component and development WP leads to have an overview of the activities carried out so far and their degree of completion. Secondly any party, from inside or outside ZDMP consortium, who wants to create or integrate software into the ZDMP platform, as the document outlines the basic requirements to achieve integration.

0.4 Deliverable Context

Its relationship to other documents is as follows:

Primary Preceding documents:

- **D021 – Methodology Document**: Provides the foundations of collaborative software development and points to the different tools that are being used in the project.
- **D048 – Requirements Document and Update**: Contains the end user requirements and those are mapped to the different ZDMP components.
- **D051 – Global Architecture Specification and Update**: Depicts the first and primary version of the architecture
- **D053 – Functional Specification and Update**: Contains the functional specification of ZDMP components and zApps.
- **D055 – Technical Specification and Update**: Describes the APIs of the ZDMP components.

Primary Dependant documents:

- **D009 – Technical Management**: WP5 Report: Gives an in-depth view of components developed within WP5

- **D012 – Technical Management**: WP6 Report: Gives an in-depth view of components developed within WP6
- **D015 – Technical Management**: WP7 Report: Gives an in-depth view of components developed within WP7
- **D018 – Technical Management**: WP8 Report: Gives an in-depth view of components developed within WP8

0.5 Document Structure

This deliverable is broken down into the following sections:

- **Section 1: ZDMP Software Development:** An introduction to this deliverable including a general overview of ZDMP software development, in addition it addresses the reviewer's comments
- **Section 2: Components Integration:** Technical requirements to be meet by any software piece to be integrated into ZDMP platform.
- **Section 3: Components Overview:** Provides an overview of the software components developed in terms of requirements coverage, security issues, quality management, integration and data and risks issues
- **Section 4: Conclusions:** Main outcomes recap and lessons learned

- **Annexes**:
  - **Annex A**: Document History
  - **Annex B**: References
  - **Annex C:** Security Logging, Traceability and Event
  - **Annex D:** Functional Requirements Status

0.6 Document Status

This document is listed in the Description of Action as "public" since it provides information which may be relevant to external parties wanting to produce or integrate software in the ZDMP platform.

0.7 Document Dependencies

This document is part of an iteration of living deliverables. This is the first version produced in M18. Future versions will be submitted for approval will be delivered in M30 and M48. Consortium-internal updates of this report will be produced in M24, M36, and M42. Each version will provide updated information mainly – but not only – in the component overview and the reviewer's recommendations sections. The additions will be clearly marked in further iterations of the document.

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

- Online documentation for each component is available at: https://software.zdmp.eu

0.10 Reading Notes

- None

0.11 Document Updates

Following the M9 Review comments to D021 Methodology Document and technical management, the issues raised were addressed as follows:

Issue	Comment
Several comments related to the hybrid-agile methodology, quality procedures	Due to their comprehensive nature see Section 1.3

1 ZDMP Software Development

1.1 Reporting

This document is the first introduction of the software development conducted in ZDMP. There are a number of deliverables and documents reporting the development activities since ZDMP solutions encompass a large number of software components all interconnected. In addition, there are 3 formal iterations of most of these documents at M18, M30, and M48 so the series of deliverables are repeating on a living basis.

This introductory section serves to clarify where to find the information.

Considering M18, Figure 1 provides a summary of the topics addressed in each one of the ZDMP deliverables:

ZDMP Deliverables					Content	Type
Title		ID M18	ID M30	ID M48		
Technical Management Overview Report		D006	D007	D008	High level status update on: • Software methodology followed in ZDMP • Integration issues • Reviewers recommendations • Requirements coverage • Software quality • Security assessment • Risk and Data Issues	PDF document (this report)
Technical Management:						
Technical Management: WP5 Report		D009	D010	D011	Degree of KPIs achievement until M18 per WP Complete detailed information per component and task on: • Architectural blocks development status • Planned activities for the period • Progress • Limitations found • Risks foreseen • Links to public documentation	PDF documents – one for each WP
Technical Management: WP6 Report		D012	D013	D014		
Technical Management: WP7 Report		D015	D016	D017		
Technical Management: WP8 Report		D018	D019	D020		
Software Components						
WP5	Data Acquisition and IIoT	D059	D060	D061	All these deliverables are marked in the GA as “OTHER (Prototype)” They correspond to the software produced in the technical work packages Cover documents have been created with information directly taken from the online public documentation. In case of differences, the online documentation prevails, as it is the	Software & Online Public Documentation
	Robust Industrial Network Support	D062	D063	D064		
	Data Harmonisation and Interoperability	D065	D066	D067		
	Orchestration, Monitoring, and Alerting	D068	D069	D070		
	Distributed & Autonomous Computing	D071	D072	D073		
	ZDMP AI and Analytics	D074	D075	D076		

WP6	SDK: Applications and Service Builder	D077	D078	D079	most up to date information about the component. All these components are already comprehensively described in WP1 Technical Management deliverables, but the information gathered in the public documentation is the one interesting for audiences out of ZDMP ecosystem For each one of the components (or as otherwise stated in the work packages reports), the following information can be found per component: • A general description • Source code • Latest release • Open API specification • Features • Requirements (to use) • Installation instructions • How to use it Links to each one of the components produced can be found at https://software.zdmp.eu/	
	Secure Business Cloud	D080	D081	D082		
	Human Collaboration Environment	D083	D084	D085		
	Platform Integration and Federation	D086	D087	D088		
	Inter-platform Interoperability	D089	D090	D091		
WP7	Preparation Stage: Start-up optimisation	D092	D093	D094		
	Production Stage: Equipment Performance Optimisation	D095	D096	D097		
	Material and Energy Efficiency	D098	D099	D100		
	Process Quality Assurance	D101	D102	D103		
WP8	Characterization and Modelling	D104	D105	D106		
	Pre-Production: Product Quality Prediction	D107	D108	D109		
	Production: Non-Destructive Product Inspection	D110	D111	D112		
	Production: Supervision	D113	D114	D115		

Figure 1: ZDMP deliverables and content

In addition, the following questions may guide the reader on how and to find relevant information:

- **I am internal in the project or reviewer, and I want to check the progress and status of each component. Is it here? Where can I find it?**
  Individual reports for each one of the work packages (from 5 to 8) must be checked for concrete component and tasks details. In this overview report, no component details are provided, only some insights at a project high-level view
- **Where can I find the requirements coverage until the date?**
  See Section 3.1.1 and Annex D
- **Have the developed components taken security into account?**
  Yes. Firstly, there are several generic components devoted to address specific security issues; for example:T5.2 Secure Installation, T5.2 Secure Communication, T5.2 Secure Authentication and Authorisation and T6.2 Security Designer. All components in the platform make use of them. Secondly, since security is an issue that ZDMP considers highly important, a monitoring and traceability system has been designed that will increase the overall security of the platform. Complete details can be found in the report of WP5 (D009 – Technical Management: WP5 Report), WP6 (D012 – Technical Management: WP6 Report), the online documentation available of the previous components, and in Annex C
- **Where are the source code/executables files?**
  Links are available in the online public documentation repository for all related files. Most of the components are open source, largely under Apache 2.0 license (See D026 Regulation and Trustworthy System / Data Management Document), but some others are proprietary with specific licenses. It is not the purpose of this document, or

the software documentation, to discuss software licenses. In any case IPR information, including licenses, can be found also at: software-ipr.zdmp.eu (https://es.zdmp.eu/ipr-documents)

- **If I am interested in integrating my development/component/service/application in ZDMP platform, where can I check APIs and other aspects that I have to take into account?**
  Component integration guidelines can be found in Section 2. To check a concrete API coming from a ZDMP Component examine the online documentation at https://software.zdmp.eu
- **Are there any instructions or guidelines on how to use the software?**
  Yes, as part of the public online documentation there are complete instructions on how to run a component, including software and hardware requirements, and steps to be taken
- **How can I be sure the software has sufficient quality?**
  In this report the methodological approach can be found in the next section, and software quality technical details can be found in Section 3.1.3. However, please note that ZDMP is an Innovation Project and not a commercial activity and as such it will produce software that can contractually only be pre commercialisation

1.2 Approach

The basis for the software development approach was already established in D021 Methodology Document delivered in M9 and where a set of agreements about source code management, and tools to be used amongst others were made.

The following subsections take these agreements and examine how they have been applied. The red boxes provide a summary of the contents in D021 Methodology Document and these are briefly discussed afterwards.

1.2.1 Methodology

A hybrid methodology is proposed in ZDMP, a mixture of **Waterfall**, and more agile approaches (**Prototyping and SCRUM**).

Waterfall was used in the work previous to development itself including the gathering requirements and building technical elements necessary for coding (architecture, functional specs, etc.).

A mix of prototyping and agile have been used during the software development phase. ZDMP has formed small teams around each one of the components with each having a component responsible - “component leads”.

An Agile approach has been taken since the beginning of development. Component leads were running a total of 8 sprints on average per component until M18. Each sprint has lasted two weeks and has involved a minimum of two audio conferences per sprint.

Early prototypes were developed for most of the components with software running after two/three months from the start of the development. These prototypes were presented in the first virtual meeting of the project, and all partners were invited to provide feedback, including industrial end users.

This approach of two weeks sprints and early prototypes has worked well for the period and it is planned to be maintained in the next phase.

1.2.2 Source Code Management

A **GitLab project** has been created for each one of the components and zApps. The component/zApp lead developer is responsible to keep their GitLab project up to date with all the information (repository files, issues etc), and to follow the guidelines established in this document.

Each Gitlab project will have a set of features available, such as its own shared repository, issues etc

A dedicated GitLab has been set up for the ZDMP software development work, and it is available at: gitlab.zdmp.eu (https://zdmp-gitlab.ascora.eu/.)

It is the common source code repository where Gitlab projects have been created for each one of the components, and Figure 2 shows the structure created. For most of components GitLab acts as a central code repository and activities are tracked and monitored by making use of the “issues” functionality.

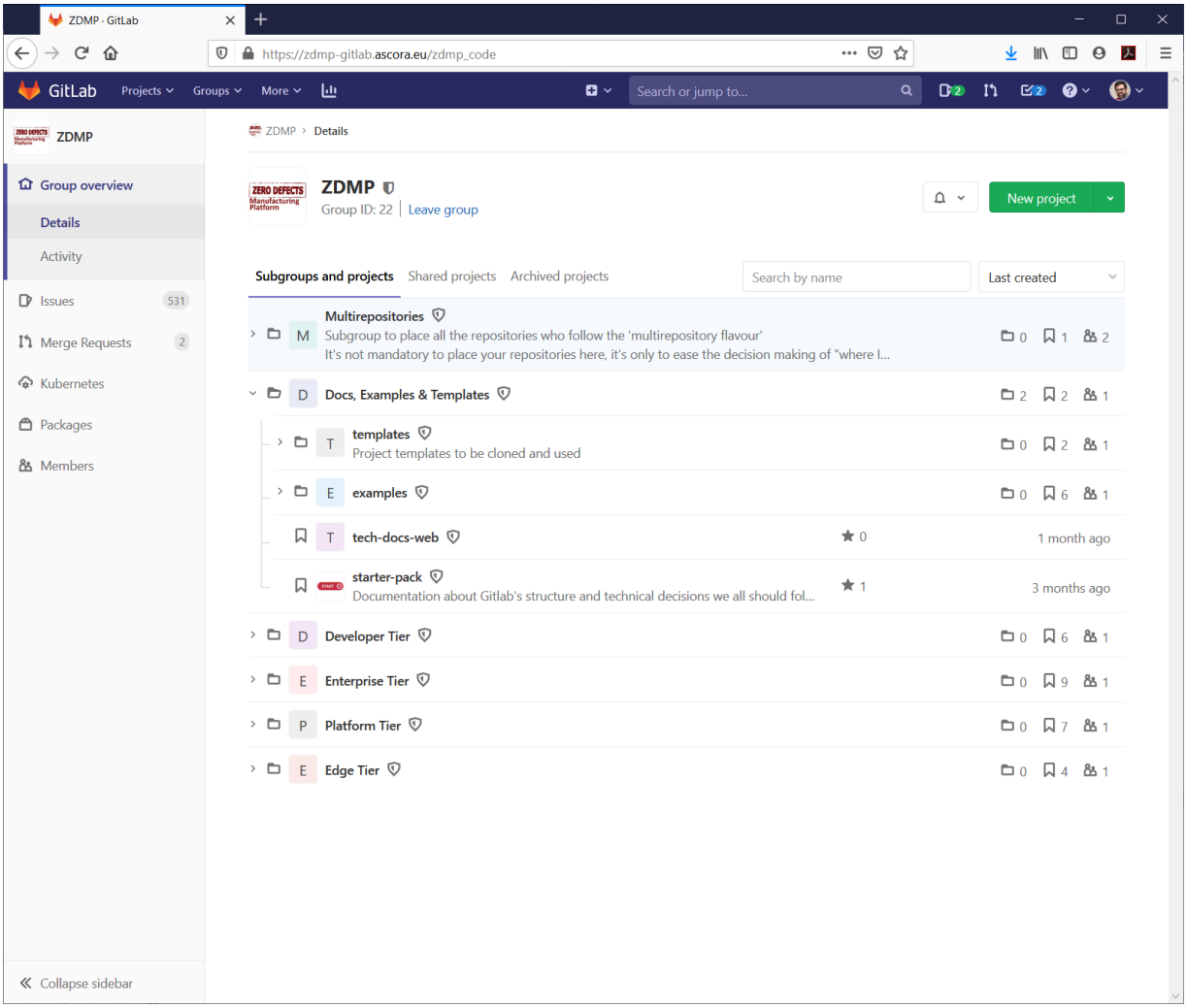


Figure 2: ZDMP GitLab

1.2.3 Continuous Integration

GitLab Continuous Integration, Delivery and Deployment (GitLab CI/CD) is selected, since it allows zero effort integration with GitLab and does not depend on any plugin. Each component or zApp leaders configures the right pipelines in GitLab according to their needs

Basic pipeline functionality is in place which provides at a minimum: Automatic public documentation generation, code quality, and security inspection.

Figure 3 shows a screenshot of the Product Assurance Run-time component pipelines.

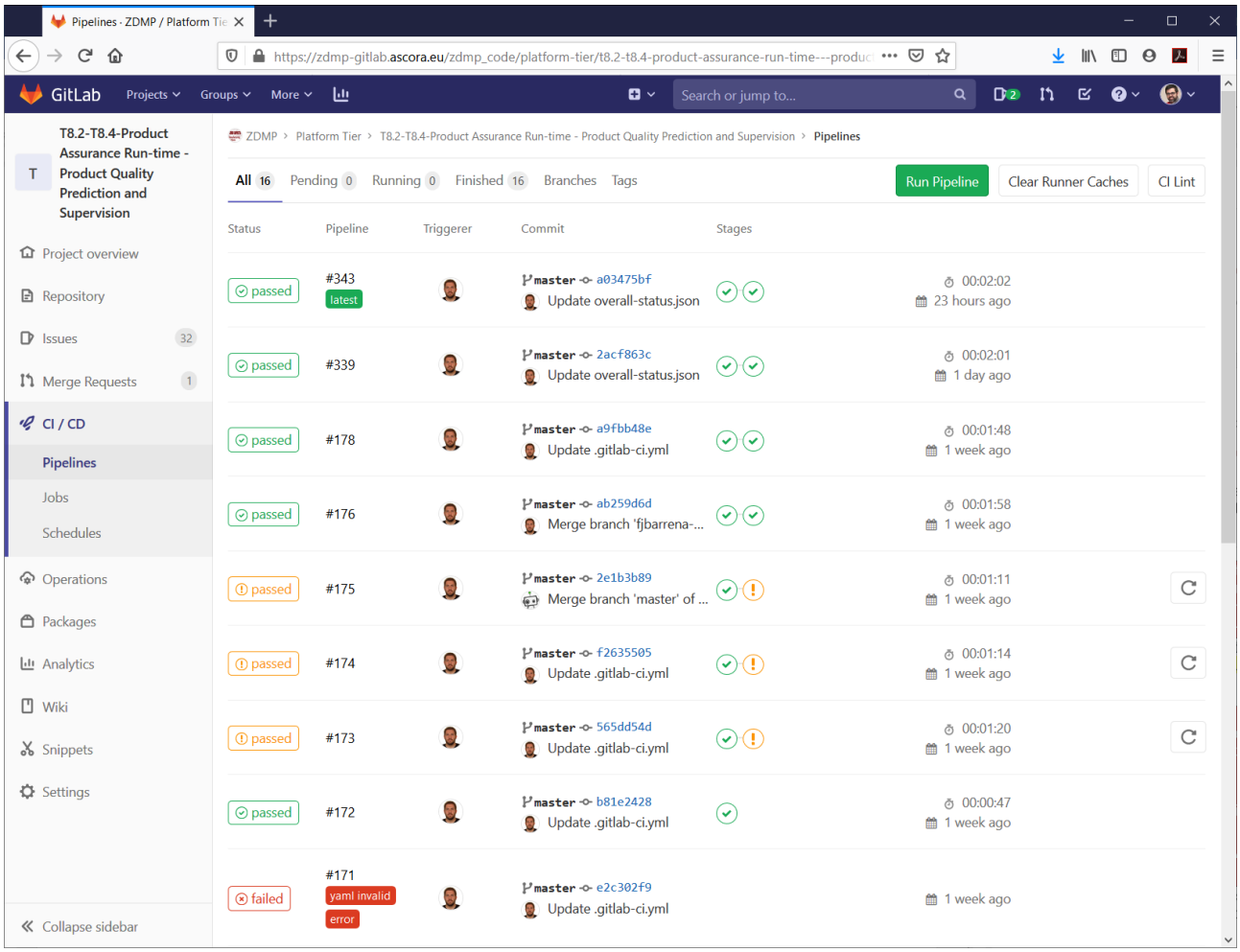


Figure 3: Product Assurance Run-time component Pipeline

1.2.4 Issue Tracking

An issues board is created for each one of the GitLab projects. A set of labels will be created to categorise issues. The issues will be used to create the roadmap of the component or zApp, so each issue will be a micro task to be conducted in the project.

The zApp issue detail must contain a list of the requirements that are directly tackling. If new requirements emerge during the project, new issues will be open to cover them.

Component or zApp leaders will be responsible update the issues list and assign responsibilities between the participants.

The Issue board has been used to track the micro activities and split the work between sprints of the different teams. Component leads and contributors keep the list up to date with latest achievements and the tasks they are working on.

As the development of zApps has not started, these have not been configured in GitLab and the requirements have not been mapped yet.

Labels have been used to differentiate the sprints. As an example, Figure 4 shows the board of issues of the Product Assurance Run-time component.

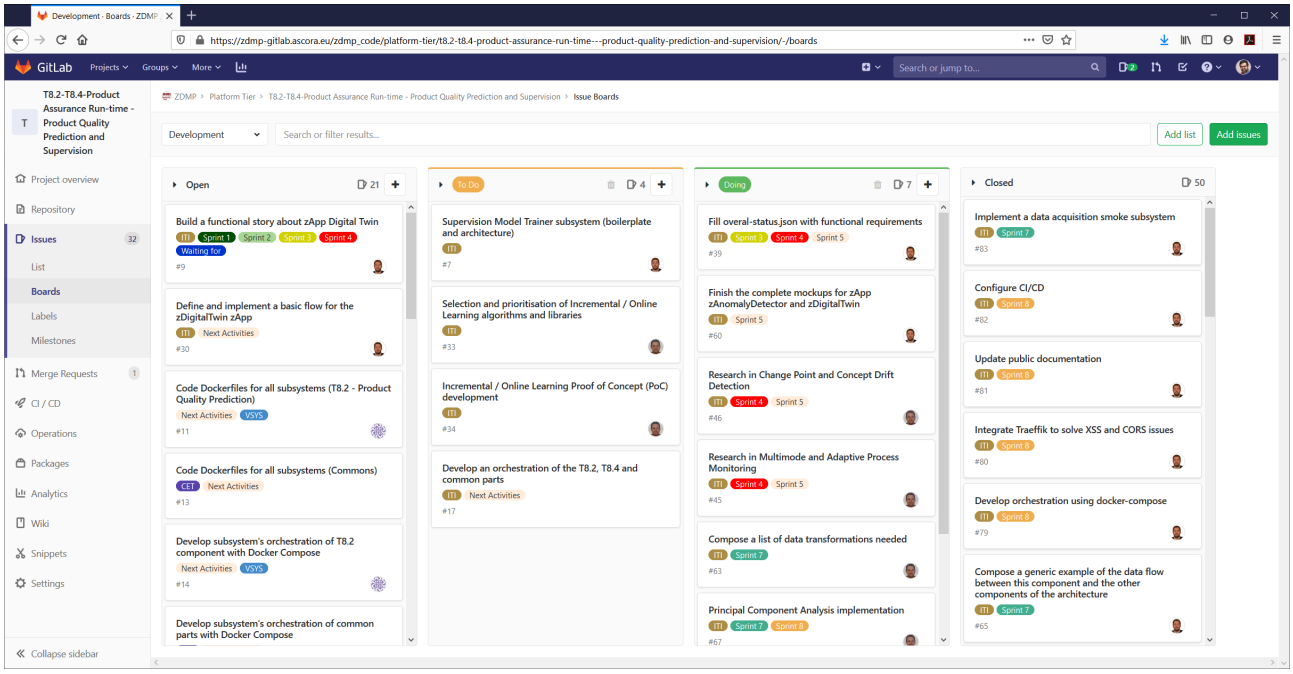


Figure 4: Board of issues of Product Assurance Run-time component

1.2.5 System Deployment with Docker

Docker will be the preferred solution to create, deploy and deliver any component or application, being delivered as containers.

Docker is the main technical tool for the components ZDMP builds. Whenever possible, components are containers that expose one or several APIs making them easy to integrate with other components and applications.

In addition, this approach makes easy to install and deploy the services that a component is providing since it is normally just downloading and deploying “a docker”.

1.2.6 Software Design Quality Analysis and Evaluation

The design of the different components will be reviewed by the architecture leader (ICE) each 6 months to guarantee that the definitions are consistent with the architecture design (D4.3), the functional specification (D4.4), and the technical specification (D4.5), and putting an emphasis on its completeness, consistency, and correctness.

In addition, component leaders will agree on the static analysis to be carried out in case of need, as well as the simulation and prototyping activities to evaluate the design, as these will strongly depend on the component, prior to the start of any development or integration.

The degree of architectural completeness, in terms of what has been produced, is reported in the technical management WP reports per component. Details are given in deliverables D009, D012, D015 and D018.

Prototyping activities have started, but the work of evaluation and simulation will take place in later stages of the project.

1.2.7 Software Construction Quality

Sonar will be used and integrated with GitLab to provide code analysis.

A specific instance of Sonar has been deployed as a tool to help developers in producing software of the requested quality, and at the same time to ensure the quality of the code is high.

The ZDMP Sonar page is available at: sonar.zdmp.eu (https://zdmp-sonar.iti.upv.es/

An analysis extracted from this tool can already be checked in Section 3.1.3.

Figure 5 shows some detail of the Sonar ZDMP Dashboard.

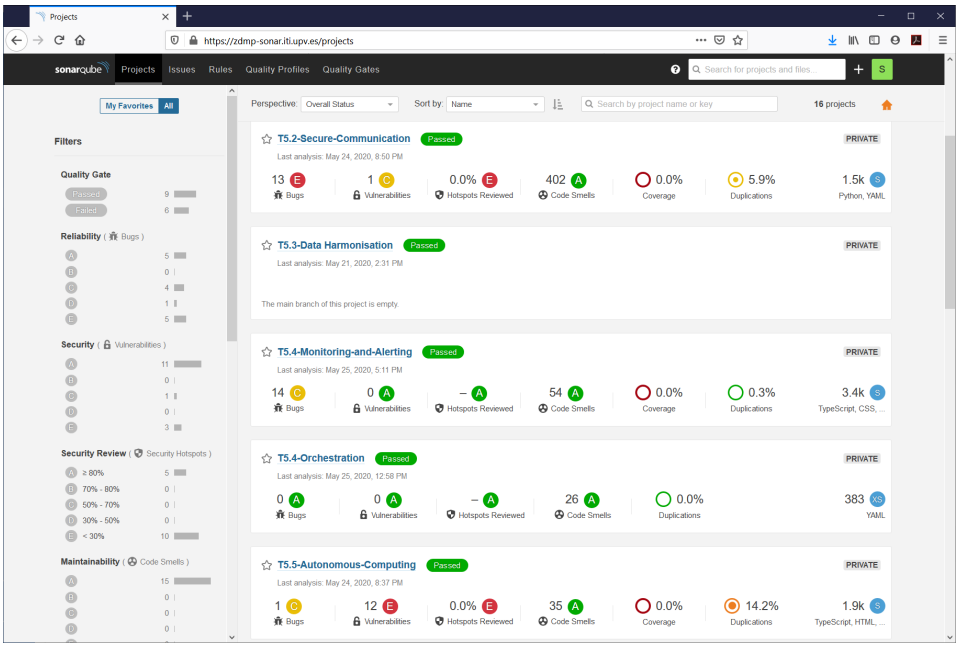


Figure 5: ZDMP Sonar Dashboard

1.3 Reviewer's comments

Figure 6 collects the recommendations and comments from the reviewers in M9 of the project related to technical management. It clarifies the issues and provides pointers where additional information can be found.

Reviewers Comments	Consortium Reaction
The potential success of the hybrid Agile and Waterfall development methodologies. The Project Technical Manager must closely monitor backlog and project plan across different geographic locations working. The Periodic Consortium meetings should have dedicated cross organisational technical meetings where demonstrations should be performed and where integration of components coming from different groups is required a review of integration requirements and development status should be performed. The success of this approach should be documented and presented in the next periodic progress report and described to the evaluators at the next periodic progress review.	This approach was already defined in the proposal and thus evaluated well and included in the DOA. It is used due to its success in other projects as being well-suited to European RIA/IA project. At the beginning of the project ZDMP was following a classical waterfall approach, the development work has completely gone to Agile, with sprints of two weeks and a strict method of reporting via GitLab. This reporting and distribution was necessary due to the high amount of developers involved in the work spread in different geographical locations. The methodology followed has been quite strict on the way of using GitLab. The technical management has elaborated an automatic procedure to extract the progress of tasks, so at any moment it can be monitored the progress in each one of them. Finally, this report, together with the rest of technical documentation, shows the progress and the success of this methodology which has worked well in past projects similar to ZDMP such as vf-OS and CREMA all of which have been ranked well in delivery Specific comments: • The Project Technical Manager must closely monitor backlog and project plan across different geographic locations → This monitoring is being shared between the Technical Manager and the Work Package Leaders. Every two weeks, the Work Package Leader has sprint meetings per task, in which a review of the backlog and task progress is performed. Then, Work Package Leaders and the Technical Manager have regular meetings to share the status of the development and take actions if necessary. In any case, Technical Manager (under request) joins in sprint meetings where clarifications are required. • The Periodic Consortium meetings should have dedicated cross organisational technical meetings where demonstrations should be performed → Every consortium meeting has had specific slots for technical meetings since the start of the project. These slots have been used to discuss the integration of the components and establish the technical foundations of the overall project among other issues. The first demonstrations were performed in Steyr's virtual meeting in April 2020, and that dynamic will be continued in the next meetings. • Integration of components coming from different groups is required a review of integration requirements and development status should be performed → At the early stages of the project, the development teams were focused on generating the first functional versions of ZDMP components, integration requirements were taken into account. The first demonstrations were showcased in Steyr's

	virtual meeting. Since then, several cross-component integration meetings have been held, resulting in preliminary fully functional integrations. Nevertheless, a cloud server infrastructure is being set it up to deploy ZDMP component instances, run the pilots, and start with the integration tests.
The hybrid agile approach may not work as planned due to high complexity, tight constraints and local distribution of the team	The comment in this case is like previous one. The hybrid approach was evaluated by the technical manager, and previous projects, as successful for the project so far, and the results are extensively documented in the technical reports presented in Section 1.1. The waterfall phase has been completed and aspects relating to this were successfully evaluated at the M9 review. The second phase is the agile this if this might not work it would have nothing to do with hybrid-agile only the success of agile which is well known and documented. Agile is known to work well with complex projects and through regular project “show and tells” both to developers/users it gets around the distribution of the team. In fact, one of the main reasons for hybrid agile is to precisely counter the distributed working of EU project activity.
The contemporaneous use of Agile and Waterfall development methods are not entirely defined here. The central Idea of RAD/Scrum/Agile methodologies is contrary to the fixed requirements and long-term planning / scheduling described here.	The project does not mix Agile and waterfall in the development phase. See above
GitLab - It is unclear however how the enforcement of quality procedures takes place when or how	ZDMP is using two tools to ensure the quality in the software development work: • GitLab CI/CD pipelines • Sonar These two are briefly discussed in previous Section 1.2. Results out of quality procedures enforced are shown in Section 3.1.3. The quality procedures are enforced by using Gitlab CI/CD pipelines as an execution environment. A pipeline is composed of a set of jobs (See Section 1.2). These jobs are executed when a new commit to the master branch is created, although this configuration can be adapted depending on needs. Every component may specify its own jobs by using YAML format file, but must have the following minimum mandatory jobs: • Software quality: Gitlab’s CI/CD analyses every commit, and uploads the results to SonarQube where all developers have access. • Software security: Similar to previous case, Gitlab’s CI/CD analyses every commit, and uploads the results to Dependency Track instance. Again, all the partners have access to it and can improve the security of their software.

Figure 6: Reviewers Comments Table

2 Components Integration

ZDMP is a complex platform, which needs to integrate several components to work. On the one hand, the integration is of paramount importance at the initial phases of the development. To ensure, or at least to ease, that integration every component must follow a set of guidelines to fit in the overall ZDMP platform. On the other hand, the software development process encompasses many different activities including the selection of a programming language, validation, documentation, maintenance, enhancement, etc, it is not the intention to strictly guide developers in a single way to carry out the development not least because they come from many backgrounds and each party having their own processes. The developer user experience is important, if development process is too complex, unclear, or confusing, they will not use it. For the above reasons, ZDMP allows several ways to work with, but all have some commons activities, specified interfaces, and way of producing software that are described in the following subsections.

2.1 Commons

Every component repository must have the following mandatory structure:

- Documentation:
  - Public:
    - Img
    - Openapi
    - Readme.md
- Orchestration:
  - docker-compose.yml

In which:

- Documentation → Public → README.md:
  - Contains the public documentation which is automatically published using CI/CD pipelines
- Documentation → Public → img:
  - Contains the images used in README.md
- Documentation → Public → openapi:
  - Contains the Open API specifications of the component
  - These are published automatically in the online public documentation
- Orchestration → docker-compose.yml:
  - Contains the recipe to raise up the component in standalone mode automatically using docker-compose

2.2 Component orchestration

Every component has his own architecture, and every architecture is composed of several subsystems. Figure 7 shows an example of a specific architecture defined for a ZDMP component:

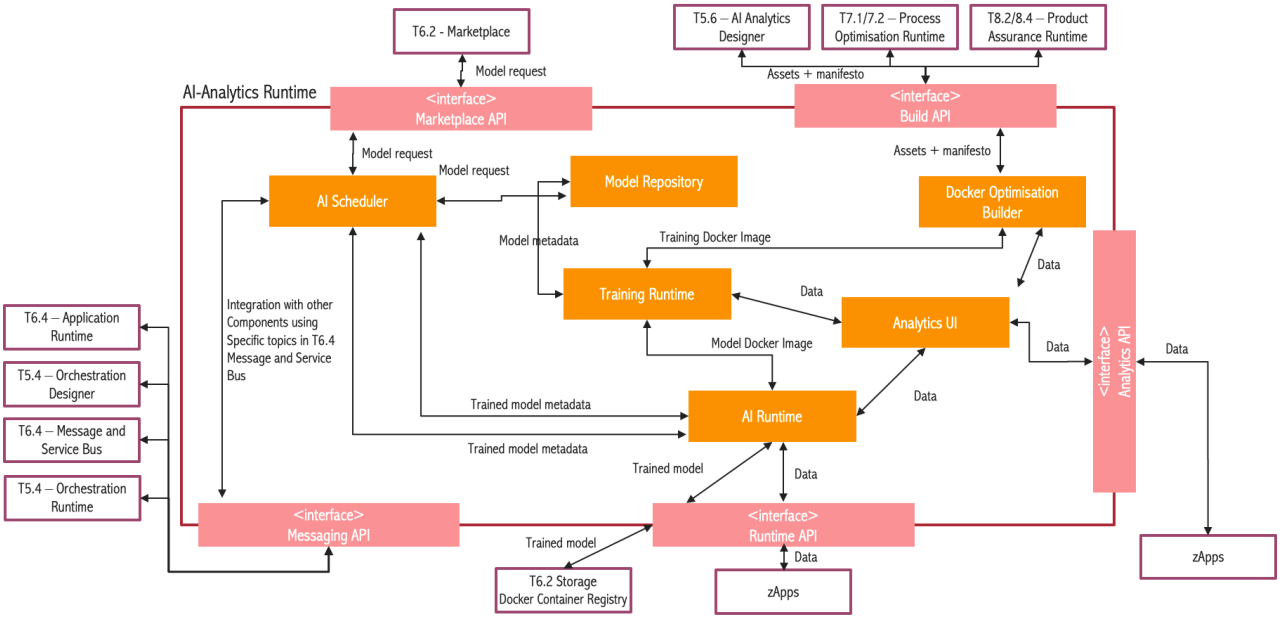


Figure 7: Architecture Diagram Example

As shown in the example, each component is formed by a set of subsystems. Those subsystems can be transformed in APIs, databases, frameworks, engines, etc. The technology used in the component is not important from an integration perspective point of view. Developers are free to use the technology they think best accomplish their targets; the only requirement is that it must run in a docker container.

There is the need to orchestrate all those subsystems, connect and run them easily, and this must be performed without the need of knowing specific technologies and implementations.

Of course, there may be exceptions. In fact, some of the components cannot be Dockerized, because they may be eg desktop or mobile applications; as long as they are client applications that consume ZDMP services there is not an issue.

Every team of developers can build their own component orchestration in several ways - “flavour” in the ZDMP terminology. They can be monorepository, multirepository, proprietary or mixed, and this is explained in the following subsections.

2.2.1 Monorepository flavour

One approach is to use a monorepository flavour. Here the developers put all the code of each subsystem in a folder. The standard folder name is called ‘subsystems’ in ZDMP, but this is only to ease the CI/CD tasks.

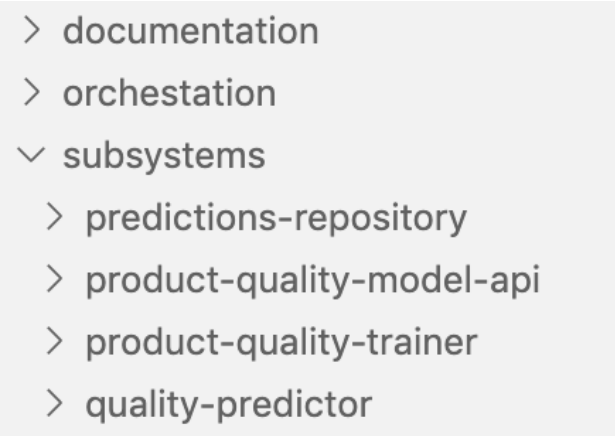


Figure 8: Example of Monorepository Flavour

This approach has several benefits:

- Easy to set up a CI/CD pipeline. It is only needed to configure it in one place
- docker-compose.yml configuration directly over the code, avoiding the need to use another system (package managers, docker registries, external servers, etc)
- All the team can examine the code, and know how works the subsystem
- Unified management: issues, milestones, and code are placed in the same repository

Of course, not all the deployments can fit in this flavour. Proprietary software code may not be uploaded, or simply the technical team may feel uncomfortable with this structure.

2.2.2 Multirepository flavour

Another approach is to use a set of repositories. In this case, and taking the example explained in the monorepository section Figure 8, it may be composed of:

- Four repositories, one per subsystem defined in the architecture (predictions-repository, product-quality-model-api, product-quality-trainer and quality-predictor):
  - Self-organized:
    - Freedom to choose the internal structure
    - Freedom to choose technologies, stacks, or another approach
  - Result of the repository should be a docker image, and be stored in a docker registry
- The component official repository must follow:
  - Commons structure
  - Docker-compose.yml should link the docker image stored in the docker registry

In summary, the multirepository flavour approach should:

- Allow the creation of as many repositories as the developer team needs, without restriction
- Build docker images for every subsystem and store them in a docker container registry:
  - Developers are free to use the container registry they want
- Maintain the docker-compose.yml in the orchestration folder, using the commons structure:

- Docker-compose must link to docker registry instead of source code

2.2.3 Proprietary code or external repository flavour

In some cases, the component is composed of a set of subsystems, some of them proprietary which prevents the source code to be stored in the ZDMP repository. This is the case where the multirepository flavour is used.

For proprietary code, the component must be able to start-up and be used in the ZDMP architecture. So, even in this situation, the developers are responsible to provide a properly built docker-compose.yml file.

2.2.4 Mixed flavour

Finally, all previous flavours can be mixed for a component if needed. It can be easily seen in the following example.

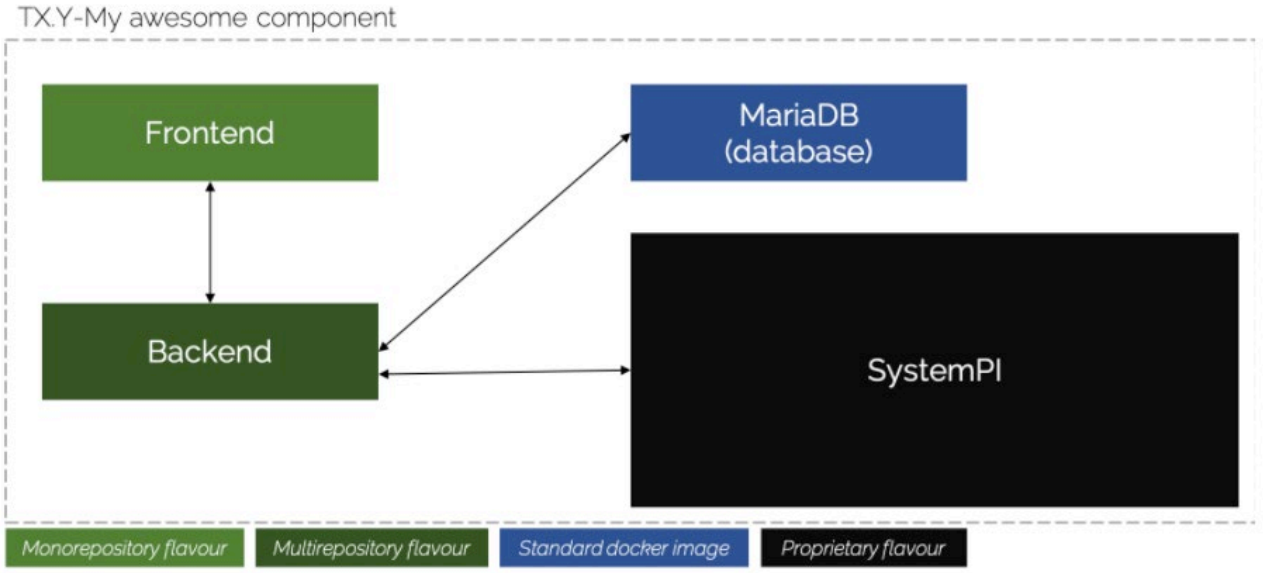


Figure 9: Mixed Flavour Example

As described above, developers must fulfil the commons structure, which consists of:

- Documentation:
  - Public:
    - Img
    - Openapi
    - Readme.md
- Orchestration:
  - docker-compose.yml

Furthermore, the Frontend subsystem follows a Monorepository flavour, and Frontend source code must be placed in the subsystems folder as shown in Figure 10.

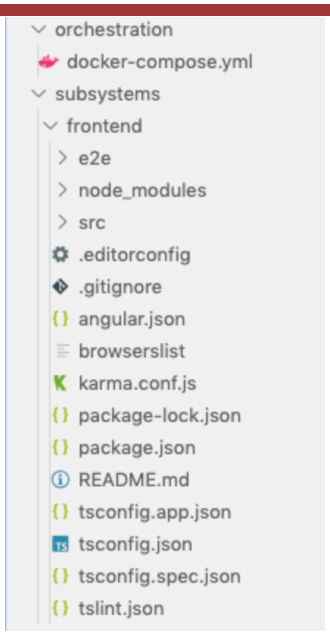


Figure 10: Monorepository Flavour Example Based on Angular

It is important to remember that chosen technology does not really matter. The key is to define the integration in the overall architecture using Docker and Docker Compose.

Then it is necessary to define how to build the Docker image to integrate the frontend subsystem into the Docker Compose. This is achieved by the definition of a Dockerfile, which has the recipe to build the final image. That file should be in the subsystem folder, in this example, the frontend folder. An example of Dockerfile for the frontend subsystem is represented in Figure 11.

```
# base image
FROM node:lts-slim

# set working directory
WORKDIR /app

# install and cache app dependencies
COPY package.json /app/package.json
RUN npm install
RUN npm install -g @angular/cli@7.3.9

# add app
COPY . /app

# start app
CMD ng serve --host 0.0.0.0
```

Figure 11: Dockerfile Example

The Dockerfile explains how a container is created but does not explain how it is orchestrated. That is the mission of the docker-compose.yml, which retrieves the build information of the previous Dockerfile and starts up the subsystem easily. Docker-compose.yml in Figure 12 starts up the frontend subsystem described above.

```
version: '3'
services:
  frontend:
    build: ../subsystems/frontend
    volumes:
      - '../subsystems/frontend:/app'
    ports:
      - '4201:4200'
```

Figure 12: Docker Compose and Monorepository Flavour

Backend subsystem follows the multirepository flavour, meaning that its source code is placed in another Git repository. In this case, source code cannot be linked to the Docker compose. The structure of the multirepository flavour can be defined by the developers working on it, and there is no need to pre-establish it at project level.

In any case, backend subsystem needs to generate a Docker image and upload it to an accessible Docker Registry, for example Docker Hub. The following image shows the backend's docker image uploaded in the ZDMP's container registry.

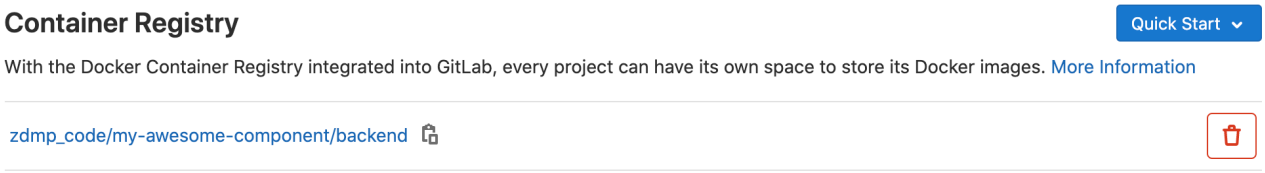


Figure 13: Docker Image in ZDMP's Container Registry

The developers can orchestrate the new backend in the docker compose file once the docker image is uploaded to a container registry. Figure 14 shows an example of integration between frontend (monorepository) and backend (multirepository).

```
version: '3'
services:
  frontend:
    build: ../subsystems/frontend
    container_name: my-awesome-component-frontend
    volumes:
      - '../subsystems/frontend:/app'
    ports:
      - '4201:4200'

  backend:
    image: zdmp_code/my-awesome-component/backend:latest
    container_name: my-awesome-component-backend
    ports:
      - '3001:3000'
```

Figure 14: docker-compose.yml Example Integrating Monorepository and Multirepository Flavours

The integration strategy of the proprietary flavour (eg System Pi in Figure 9) will be the same as the multirepository flavour, the only difference is that source code is not available in the first one. A docker image available in a container registry must be provided to ensure integration with the rest of ZDMP components.

Finally, when all the subsystems of the component are described in the docker-compose.yml file, the component may be started easily, by executing the 'docker-compose up' command.

2.3 Platform orchestration

In this section the orchestration between components is explained. The ZDMP platform can be delivered in several ways:

- As a SaaS InCloud
- OnPremise

The orchestration will vary, depending on the overlaying technologies of the systems in which ZDMP will be deployed. That means that the global integration of components is flexible.

In any case, the architectural reference implementation relies on UNIX servers, and needs a Kubernetes cluster, in which the ZDMP components will be deployed using Helm charts, which is the standard packaging model for Kubernetes applications.

The foundations of Kubernetes are close to Docker Compose defined in the previous section. Of course, the syntax varies, but the underlying concepts are the same.

At this stage, the components are still in development, and for this reason does not yet have any example of integration using Kubernetes. Nevertheless, the components can integrate one each other using Docker Compose, and is the approach followed in the project.

In the next stage of the project, once the components are more mature, the focus will be in the integration tasks. For the moment, some work has been conducted to the infrastructure needed to deploy the reference implementation of ZDMP.

3 Components Overview

3.1 M18 Report

3.1.1 Functional Requirements Status

Tables in Annex D show the functional progress per component mapped with the requirements with the progress represented as a percentage of the total envisioned for the function. Each row gives the following information:

- Functional requirement: Unique identifier coming from D048 Requirements Document and Update
- Description: Brief summary of the scope of the requirement
- Status: Depending on the level of completeness, it can be:
  - Not initiated
  - Working
  - Testing
  - Finished
- Progress: Current percentage of completion
- Comments: Any other consideration that currently affects the requirement

Figure 15 shows a fragment of Data Acquisition component table (see Annex D for complete entries).

Functional requirement	Description	Status	Progress	Comments
T51A001 - Receive asynchronous data from Data Source	Receive data pushed by a data source based on an event configured on it	Working	50%	
T51A002 - Send data for asynchronous data acquisition	Send data to the zApps or other assets through a push mechanism or to the Services and Message Bus Component through pub/sub mechanism	Working	30%	
T51A003 - Receive data from Data Source (via polling process)	Periodically queries data from data sources that do not have a push-based mechanism	Working	70%	

Figure 15: Extract from Data Acquisition component table in Annex D

3.1.2 Activity Statistics in GitLab

In this period, these are the statistics of activity retrieved from the GitLab repository related to:

- Commits
- Jobs
- Pipelines
- Issues

3.1.2.1 Commits

Most of components are making use of the full set of functionalities in GitLab, including the use of “commit” command for source code update/fix, documentation upload and other necessary actions. All commit activity can be retrieved and is shown in this section.

Nevertheless, some components do not make source code commits in ZDMP GitLab as their code is licensed and proprietary, and hence managed in an external repository. Regardless, these components show activity in the statistics since they use commit command for eg upload of documentation, docker image and docker compose specification.

So, although commits do not reflect the entire activity in all components, it is a relevant indicator and gives a good overview of activity. Figure 16 shows the activity per component.

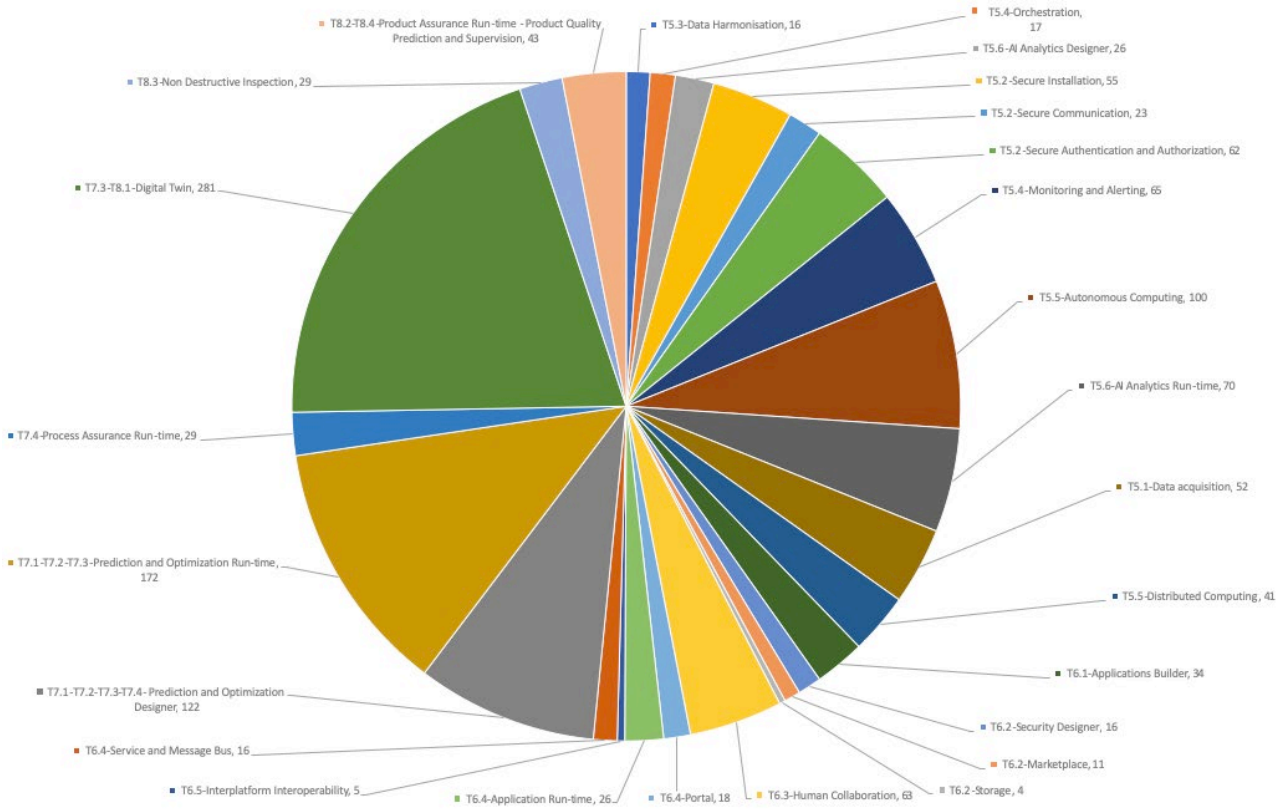


Figure 16: Commits per component

3.1.2.2 Jobs

A job is the smallest unit to run in GitLab, and is often called a “build step”. A single job can contain multiple commands (scripts) to run, and it can be:

- Build or compilation task
- Running unit test(s)
- Code quality check(s)
- Deployment task

Figure 17 shows the number of jobs executed per component. Three are the mandatory number of jobs per commit to provide the minimum expected functionality:

- Code inspection and quality assessment using SonarQube platform (See Section 3.1.4.1)
- Vulnerability and software security assessment using OWASP Dependency Track (See Section 3.1.4.2) and SonarQube platform
- Automatic documentation generation

The number of jobs executed per component depends on total commits performed and the number of jobs defined per commit.

Jobs activity is significantly lower for those components not having their source code in GitLab such as T6.4-Service and Message Bus and T6.2-Security Designer, while higher activity means more jobs are defined per commit like in T5.6-AI Analytics Runtime or T5.1-Data acquisition.

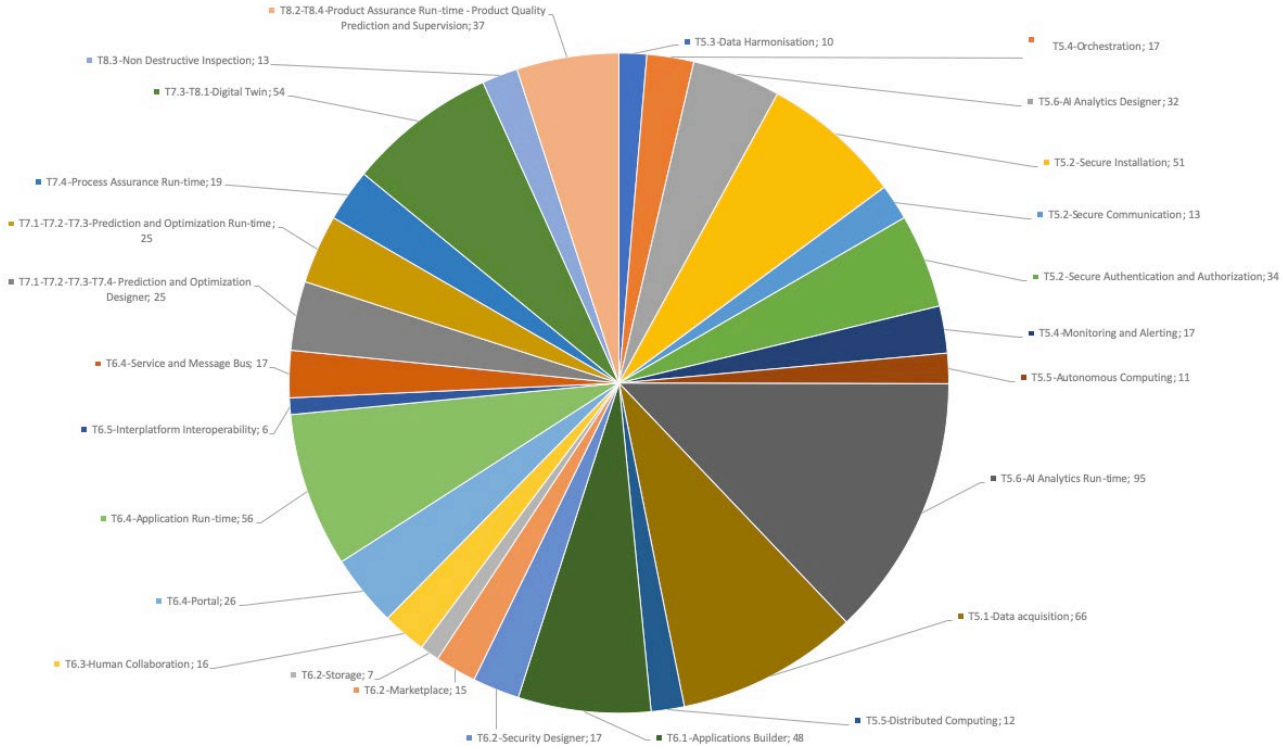


Figure 17: Jobs statistics

3.1.2.3 Pipelines

Pipelines are the top-level component of continuous integration, delivery, and deployment. Pipelines are composed of:

- Jobs, as described in previous section
- Stages, which define when to run the jobs, and can contain zero, one or more jobs to execute

Jobs are executed by “runners”. Multiple jobs in the same stage are executed in parallel if there are enough concurrent runners. If all jobs in a stage succeed, the pipeline moves to the next stage. If any job in a stage fails, the next stage is not executed and the pipeline ends early.

In general, pipelines are executed automatically and require no intervention once created. However, there are also times when the development team can manually interact with a pipeline.

Figure 18 shows the number of pipelines executed by project. Every project can create and configure its own pipelines, but all must define at least these two:

- Grouping the quality and security source code inspection jobs
- Public documentation generation

Thus, the number of pipelines executed per project depends on total commits performed and the number of pipelines defined per commit. Hence activity is significantly lower for proprietary components like T6.2-Security Designer, while number of pipelines executions are higher for those projects which defined more pipelines per commit like is the case of T5.6-AI Analytics Runtime or T7.3-T8.1-Digital Twin.

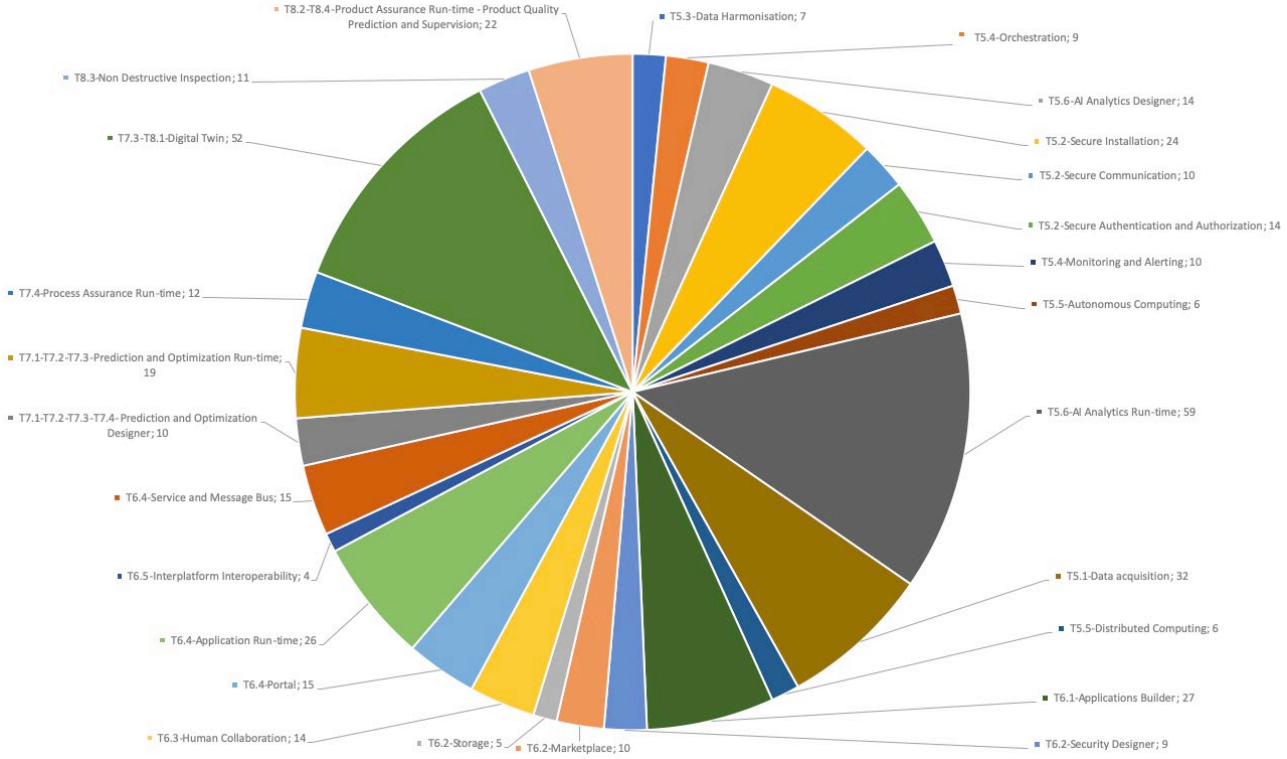


Figure 18: Pipelines statistics

3.1.2.4 Issues

In ZDMP, GitLab Issues are used for issue tracking purposes. Issues represent features, goals, discussions, and enhancements. The criticality, due dates, assigned programmers for an issue etc are managed.

The following chart shows the number of issues created per project. Every project can create and manage their own issues. Issues are monitored and tracked in the periodic sprint meetings. The number of issues depends mainly on the granularity of the issue description, for example a team can define a high level issue such as “Develop Backend and API”, where other can go down to “Add delete button to the frontend”.

T6.5-Interplatform Interoperability does not present many issues since no activity is expected at this initial phase of the project.

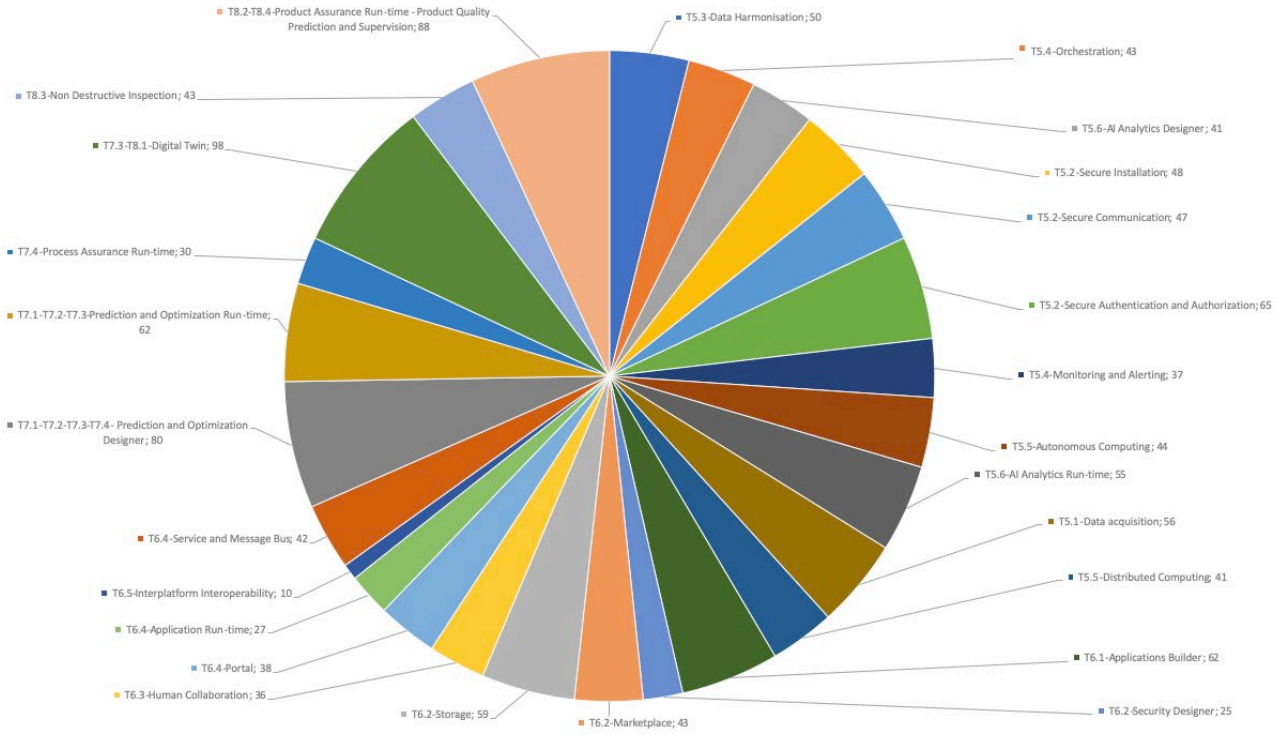


Figure 19: Issues statistics

3.1.3 Software Quality

Software quality can be measured in several ways and using different tools. In ZDMP, SonarQube is used which is an open source automatic code review tool to detect bugs, vulnerabilities and code smells (any characteristic in the source code that possibly indicates a deeper problem) in the code.

There are several ways to integrate SonarQube, but the best practice is to associate with a Continuous Integration / Continuous Deployment pipeline (CI/CD). For this, Gitlab's CI/CD engine was chosen, which allows several phases to be defined and has a good integration with SonarQube.

Thus for every repository, a new stage for the SonarQube integration is defined by using *.gitlab-ci.yml* file. That file contains the configuration to perform the code analysis, and to send the results to a central SonarQube instance, in which the developers can review the results of the static analysis. Figure 20 shows an example of the *.gitlab-ci.yml* file.

```yaml
sonar:
  stage: sonar
  image:
    name: sonarsource/sonar-scanner-cli:latest
    entrypoint: [""]
  variables:
    SONAR_TOKEN: "${SONAR_TOKEN}"
    SONAR_HOST_URL: "${SONAR_HOST_URL}"
    GIT_DEPTH: 0
  script:
    - sonar-scanner -Dsonar.qualitygate.wait=true
        -Dsonar.projectKey=T5.6-AI-Analytics-Run-time
        -Dsonar.sources=. -Dsonar.host.url
        =${SONAR_HOST_URL} -Dsonar.login
        =${SONAR_TOKEN}
  allow_failure: true
  only:
    - master
```

Figure 20: gitlab-ci.yml

The analysis is executed in every commit received in the master branch. This can be changed by every team if they are using another branch for the main development.

Figure 21 shows one example of a SonarQube report; the tool is quite extensive in the results and multiple reports can be generated to help manage the project.

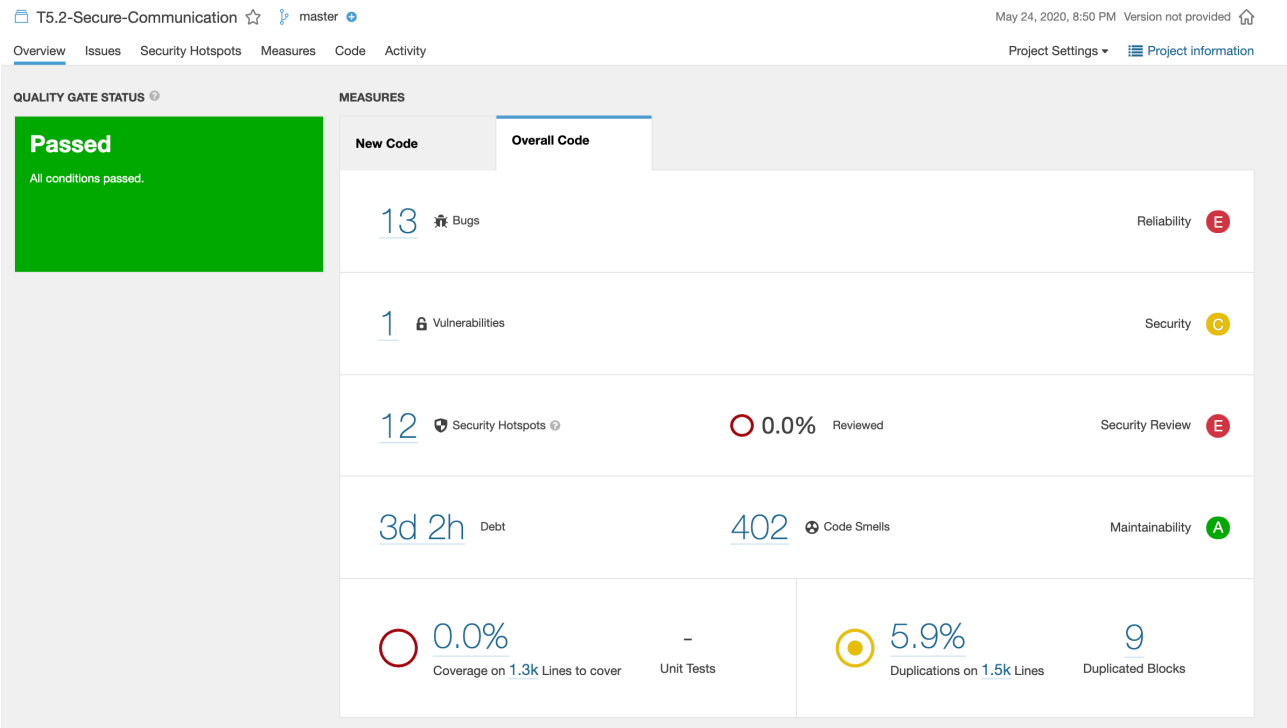


Figure 21: SonarQube report example

There are several ways to define the software quality, but there is a wide consensus about the assessment of the next metrics:

- **Maintainability.** Includes concepts of modularity, understandability, changeability, testability, reusability, and transferability from one development team to another. Poor maintainability is typically the result of multiple minor violations with best practices in documentation, complexity avoidance strategy and basic programming practices that make the difference between clean and easy-to-read code vs. unorganized and difficult-to-read code.
- **Code duplication.** Measures assessing if a sequence of source code occurs more than once, either within a program or across different programs owned or maintained by the same entity. Duplicate code is considered undesirable. A minimum requirement is usually applied to the quantity of code that must appear in a sequence for it to be considered duplicate rather than coincidentally similar
- **Reliability.** The root causes of poor reliability are found in a combination of non-compliance with good architectural and coding practices. This non-compliance can be detected by measuring the static quality attributes of an application. Assessing the static attributes underlying an application's reliability provides an estimate of the level of business risk and the likelihood of potential application failures and defects the application will experience when placed in operation

3.1.3.1 Code duplication

Code duplication is a smell in unit tests. If duplicated code appears, then it will be difficult to perform code refactoring (if needed) because there will be a disproportionate number of tests to update.

Figure 22 represents the overall status of the code duplication in ZDMP. Bubble size indicate the volume of duplicated blocks in the project, and each bubble’s vertical position reflects the volume of lines in those blocks. Small bubbles on the bottom edge are the best. The larger the bubble is, the more duplications in the project/component.

Sometimes, duplication is not in the source code but a third-party dependency. This is the case of the larger ball that corresponds to T6.2-Marketplace in which the duplication blocks are due to Javascript plugins.

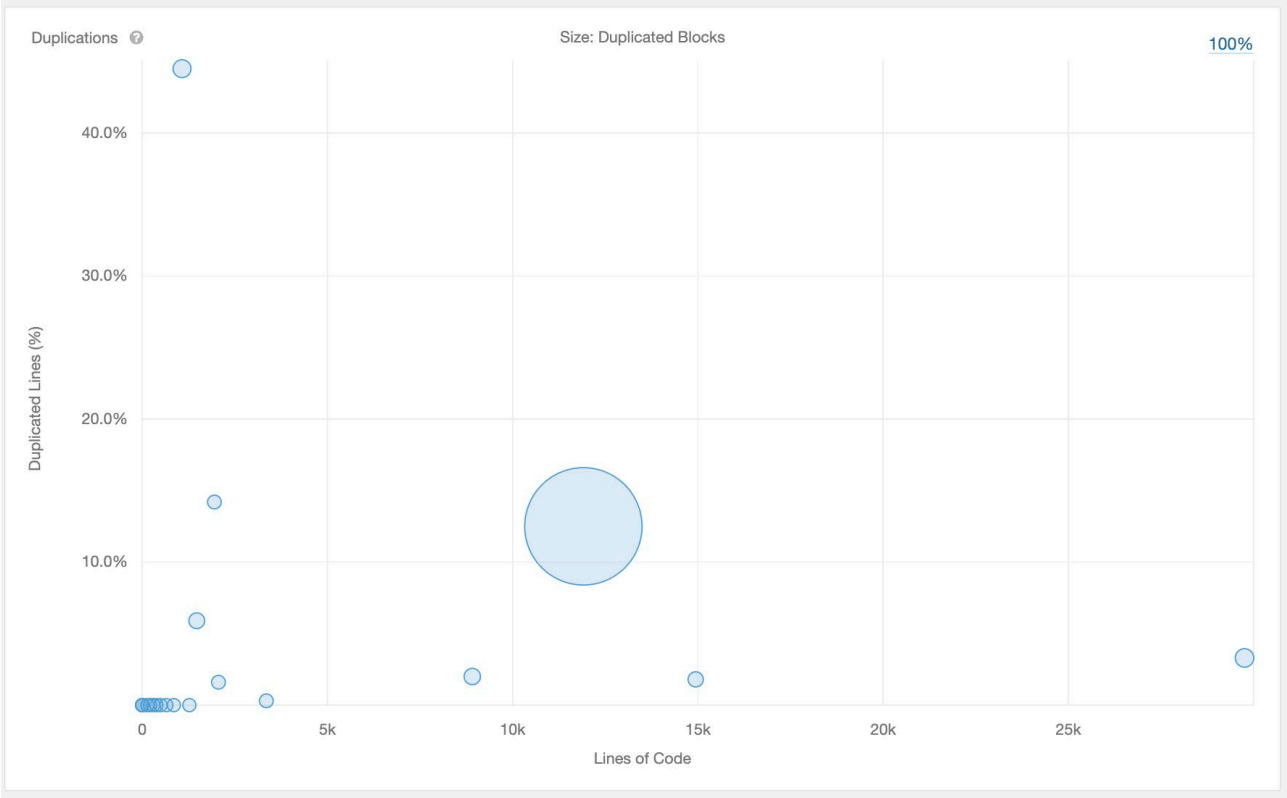


Figure 22: Code Duplication

These are the projects with the best indicators in code duplications:

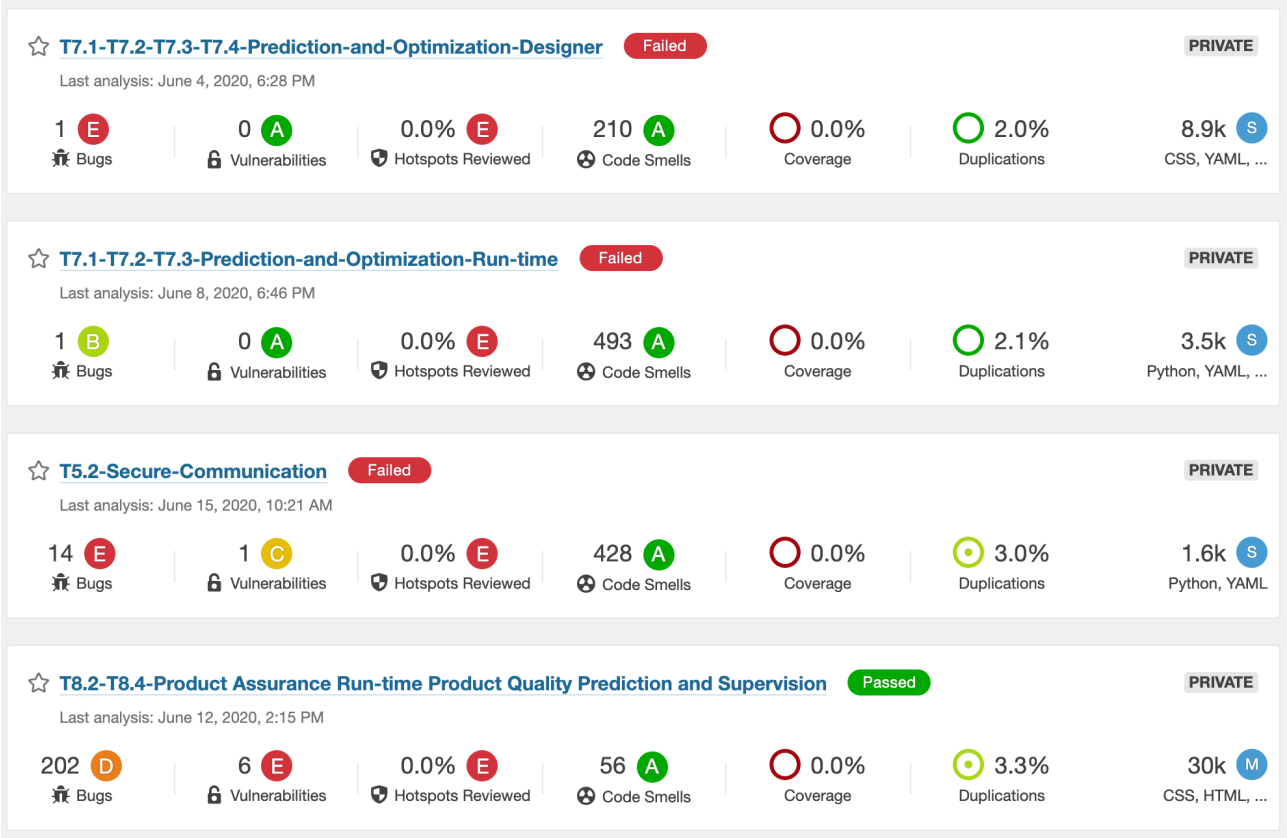


Figure 23: Top Code Duplications Projects

These are the projects with the worst code duplications.

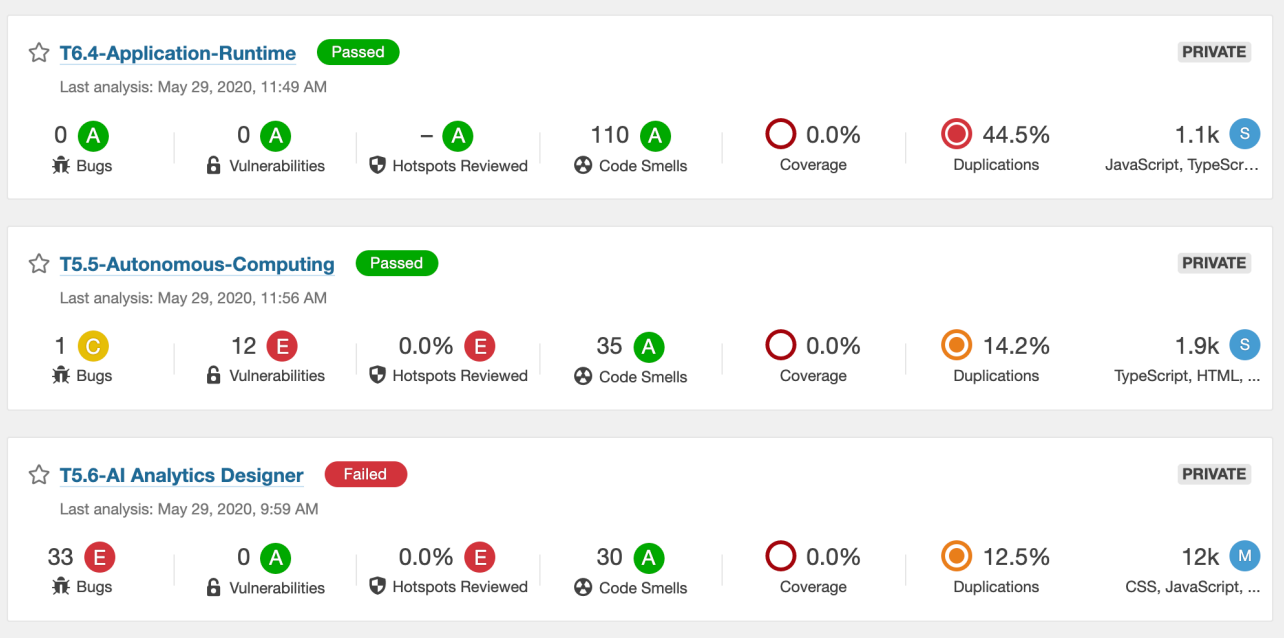


Figure 24: Top Code Duplications Projects

The code duplication in the case of the T6.4-Application Runtime is considered normal because it is based on YAML files in which, by its own nature, forces to the duplication. The lower two entries, although they have the highest code duplication rate, are within the accepted thresholds.

3.1.3.2 Reliability

Figure 25 shows the overall status of reliability in ZDMP. The closer a bubble colour is to red, the more severe the bugs are in the project. Bubble size indicates bug volume in the project, and each bubble vertical position reflects the estimated time to address the bugs in the project. Sometimes, the responsible of the lack of reliability is not the source code, but a third-party dependency. This is the case of the larger ball, which corresponds to T6.2-Marketplace. In this case the recommendation is to replace the third-party software with more trusted libraries. Small green bubbles on the bottom edge are the best.

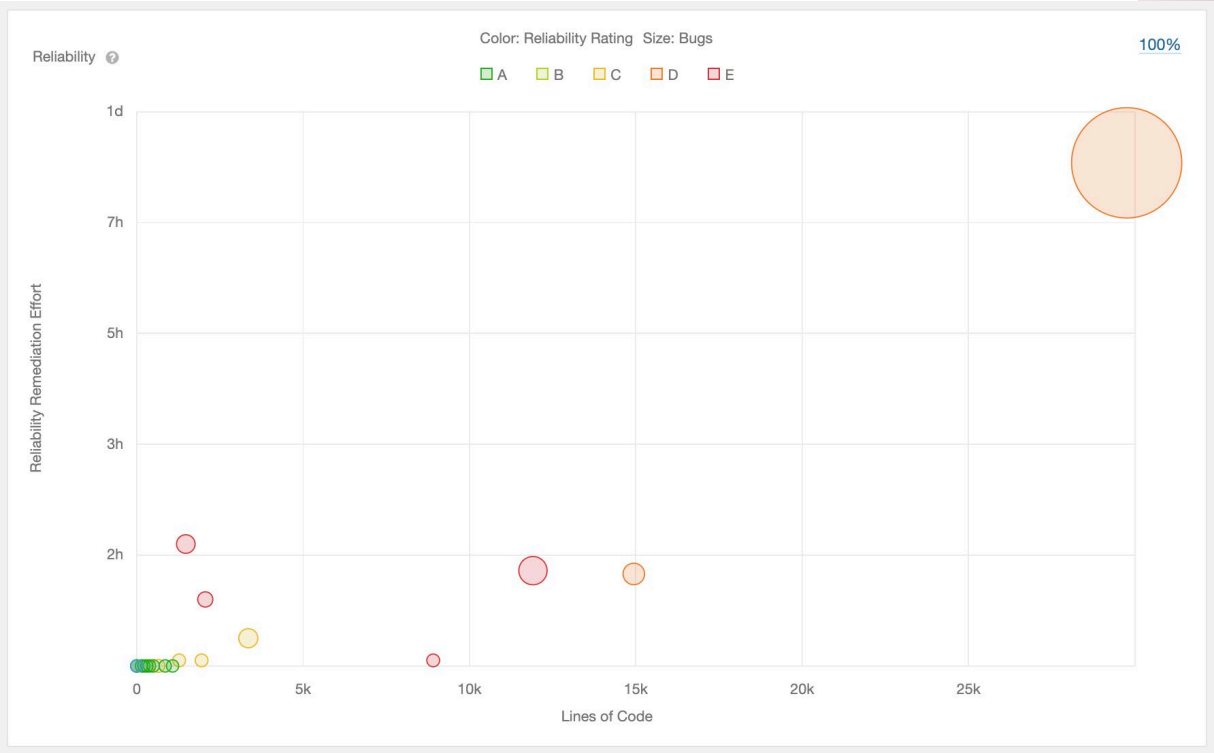


Figure 25: Reliability Overview

Figure 26 represents the worst reliability. In any case, the resolution time stays under " 1 day" in the worst case, which is considered inside the normal parameters.

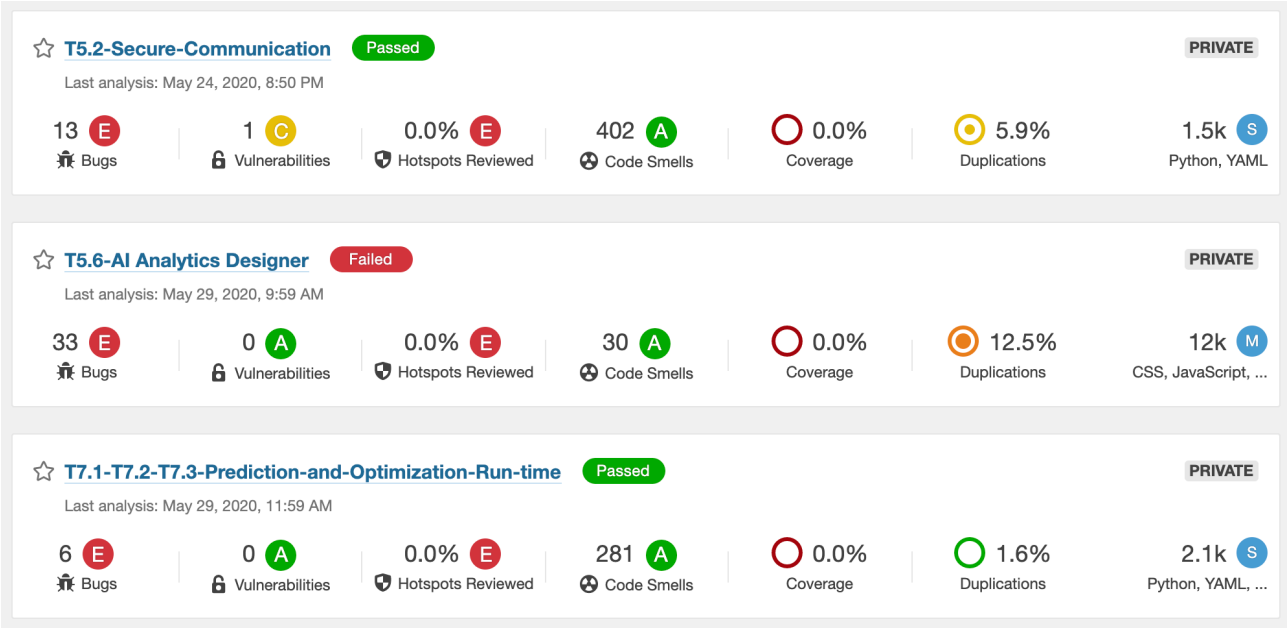


Figure 26: Worst Reliability

3.1.3.3 Maintainability

Figure 27 shows the overall status in terms of maintainability in ZDMP. The closer a bubble colour is to red, the higher the ratio of technical debt to project size. Bubble size indicates "code smell" volume in the project and each bubble vertical position reflects the estimated time to address the code smells in the project. Small green bubbles on the bottom edge are best.

It is noted that all the ZDMP GitLab projects get an A rating, which is considered inside the quality gate (the predefined criteria to be met before proceeding to the next phase).

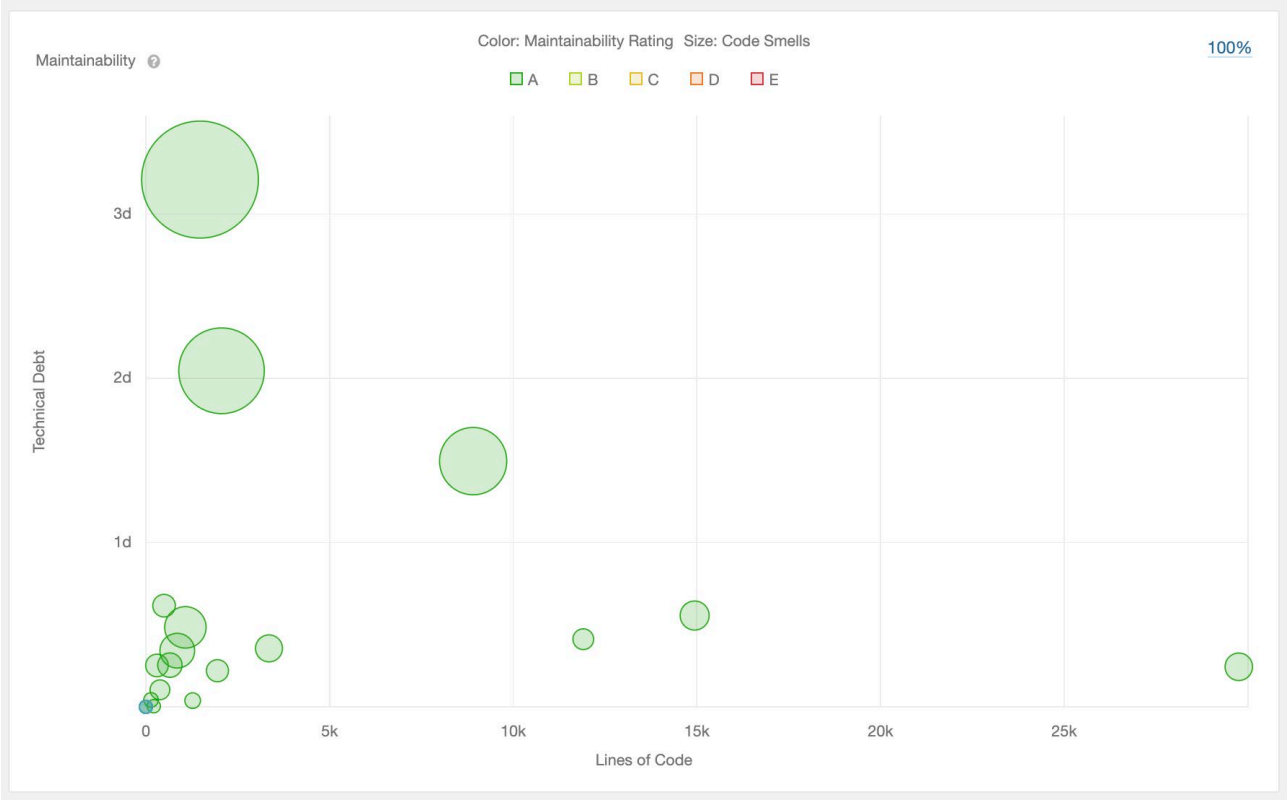


Figure 27: Maintainability overview

3.1.4 Security Assessment

Classically, the security of a platform was focused on the following elements: Systems, networks, operating systems, and configurations - thus leaving the security of the software itself out of the equation. That is not acceptable today, because the overall security of a platform is as strong as the weakest link of the architecture, and the software itself has a needs to respect the security of the users, and their data.

For that reason, new paradigms such as DevSecOps appeared. In this approach, the assessment of the security of the code is made continuously being integrated in the DevOps pipeline now renamed as the “DevSecOps pipeline”.

There are several tools to perform the security assessment, but a focus is on open source tools which can be integrated in the Gitlab’s CI/CD engine. As the ZDMP platform is strongly based on Docker containers, specific checks are made to them taking into consideration specific vulnerabilities related to Docker images.

These are the tools used in the ZDMP DevSecOps pipeline:

- **SonarQube Security.** Measures the code smells related to security, spotlighting the fragments of code which needs to be reviewed. It explains the issue, connecting with the CVE (Common Vulnerabilities and Exposures) code (https://cve.mitre.org/cve/) a universal code of known vulnerabilities. Figure 28 shows an example report
- **OWASP Dependency Tracker.** The Open Web Application Security Project (OWASP) is a non-profit foundation that works to improve the security of software. Modern software development is strongly based on Open Source resources, defined in ZDMP software as dependencies. Those dependencies have their own

vulnerabilities which is not analysed by SonarQube Security. Nonetheless, the responsibility of the security of ZDMP application is still 'the developers', and a discharge of responsibility can be claimed based on software coming from a third-party dependency. Therefore, it is necessary to analyse the dependencies, know its vulnerabilities, and act accordingly. OWASP Dependency Tracker analyses the dependencies of every software project in ZDMP, spotlighting those who have known vulnerabilities, and suggesting corrective actions

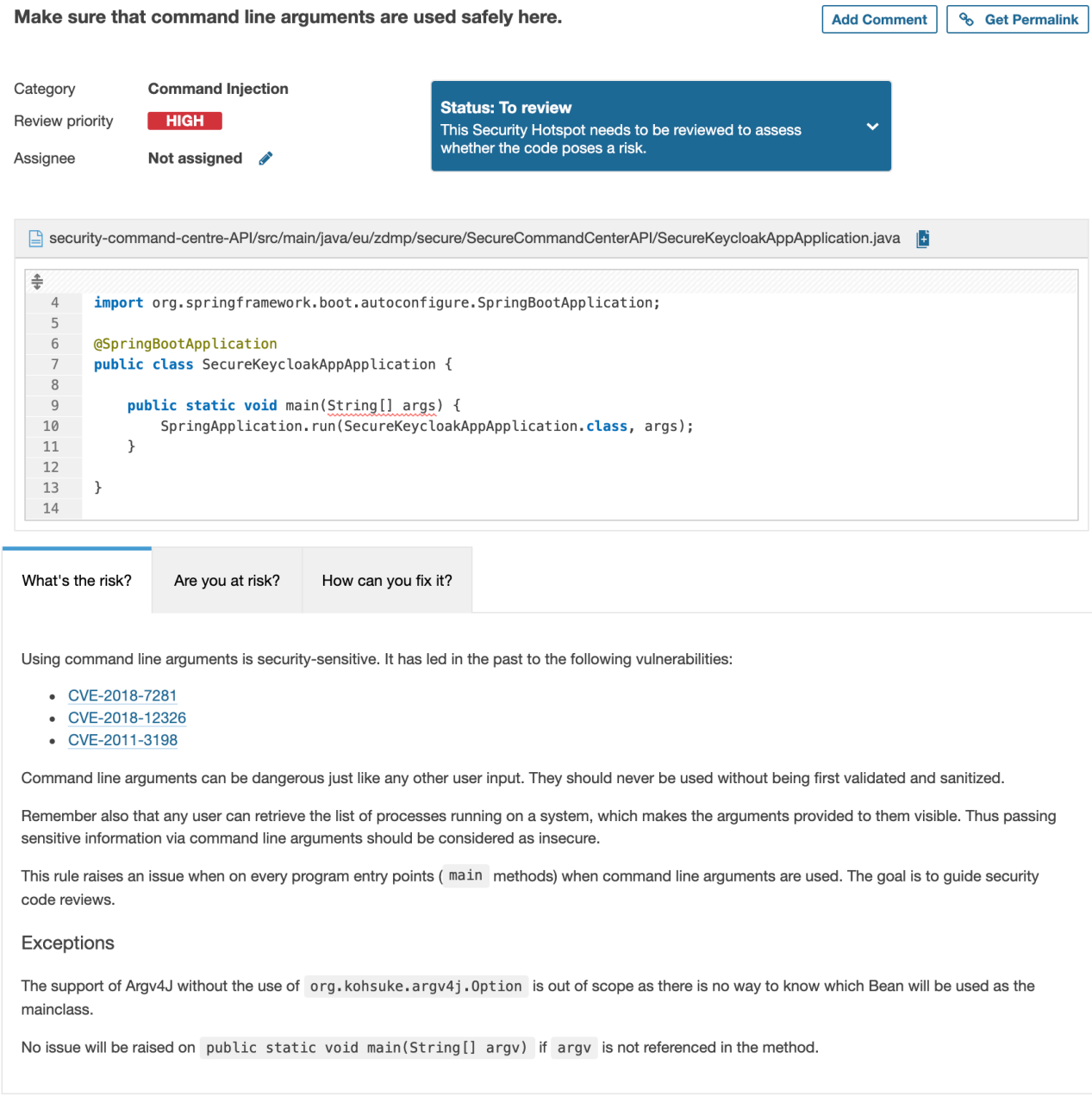


Figure 28: SonarQube Security Report

The same Sonar instance (sonar.zdmp.eu) which performs the quality inspection of the code is also the responsible for the security inspection which is show in the example of Figure 29.

Figure 29: Security report example

The ZDMP OWASP Dependency Track page is available at: dtrack.zdmp.eu (https://zdmp-dtrack.iti.es).

The integration of these tools follows the same strategy than the code quality assessment, using Continuous Integration / Continuous Deployment pipeline (CI/CD) and Gitlab CI/CD engine. For every repository, ZDMP has defined a new stage for the SonarQube integration, using a .gitlab-ci.yml file. That file contains the configuration to perform the security code analysis, and to send the results to a central SonarQube and OWASP Dependency Track instances, in which the developers can review the results of the static analysis. An example of the .gitlab.ci.yml file is shown below:

```
sonar:
  stage: sonar
  image:
    name: sonarsource/sonar-scanner-cli:latest
    entrypoint: [""]
  variables:
    SONAR_TOKEN: "${SONAR_TOKEN}"
    SONAR_HOST_URL: "${SONAR_HOST_URL}"
    GIT_DEPTH: 0
  script:
    - sonar-scanner -Dsonar.qualitygate.wait=true
        -Dsonar.projectKey=T5.6-AI-Analytics-Run-time
        -Dsonar.sources=. -Dsonar.host.url
        =${SONAR_HOST_URL} -Dsonar.login
        =${SONAR_TOKEN}
  allow_failure: true
  only:
    - master
```

```
dependency-check-node:
  stage: sonar
  image: node:12.17
  before_script:
    - npm install -g @cyclonedx/bom
    - npm install -g @fjbarrena/dtrack-cli
  script:
    - cd subsystems/monitoring-and-alerting
    - npm install
    - cyclonedx-bom -o bom.xml
    - dtrack-cli --server ${DTRACK_HOST_URL}
                 --bom-path bom.xml --api-key ${DTRACK_API_KEY}
                 --project-name "T5.4-Monit.Alerting-API"
                 --project-version latest --auto-create true
    - cd ..
    - cd monitoring-and-alerting-ui
    - npm install
    - cyclonedx-bom -o bom.xml
    - dtrack-cli --server ${DTRACK_HOST_URL}
                 --bom-path bom.xml --api-key ${DTRACK_API_KEY}
                 --project-name "T5.4-Monit.Alerting-UI"
                 --project-version latest
                 --auto-create true
  allow_failure: true
  only:
    - master
```

Sonar configuration is the same as the described in the code quality inspection in this case. Another phase in the stage is added, which is responsible for the integration with OWASP Dependency Track tool. The analysis is executed in every commit received in the master branch. This can be changed by every team if they are using another branch for the main development.

3.1.4.1 SonarQube Security

A total of 21 projects were analysed using SonarQube Security Inspection, and only 5 out of them showed vulnerabilities. These are shown in Figure 30.

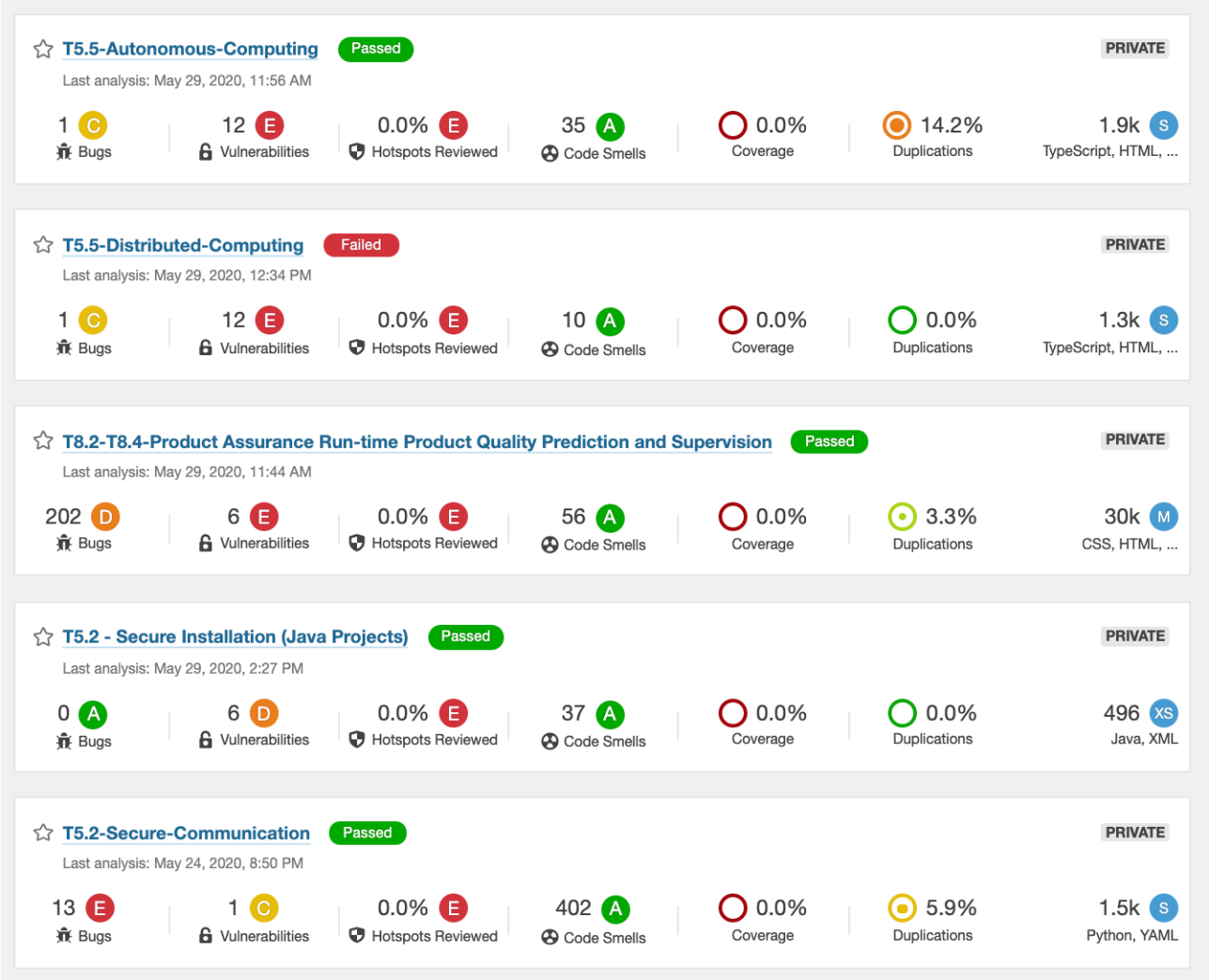


Figure 30: ZDMP Projects with Known Vulnerabilities

At the time of this report, T5.5-Autonomous-computing and T5.5-Distributed-Computing projects are the components with most vulnerabilities (12 each) in ZDMP, whilst the other three projects add up a total of 13 vulnerabilities.

SonarQube also provides a prediction of the necessary time to correct those vulnerabilities, and this is 1 hour and 7 minutes for all of them (25 vulnerabilities). This figure indicates us that these are easy to patch.

In addition, Figure 31 shows the general view of the code security assessment. The 5 large circles correspond to the previous components list although the time estimated to correct the situation is low at a maximum of 35 minutes in one of them.

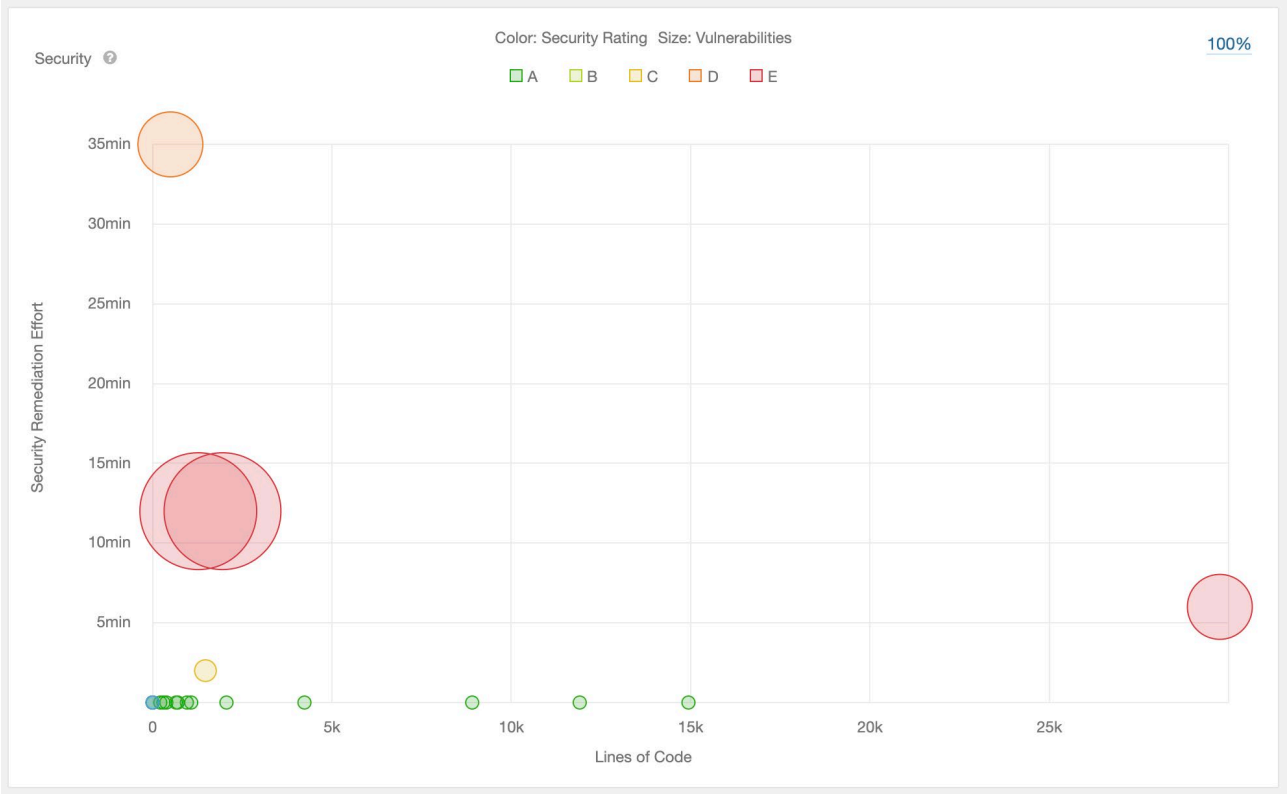


Figure 31: Code security assessment

3.1.4.2 OWASP Dependency Track Report

For M18, 27 projects were analysed using OWASP Dependency Track, of which only 4 projects showed vulnerabilities in their dependencies. This is a normal situation in new developments, since they usually start from nothing and, therefore, use the latest versions of the dependencies, which usually accumulate the latest security patches.

The difference in the number of projects between the SonarQube, and the OWASP Dependency Track is because in SonarQube some projects could be grouped in one, but this is not the case in Dependency Track.

The following dashboards (Figure 32 and Figure 33) shows the general state of the platform.

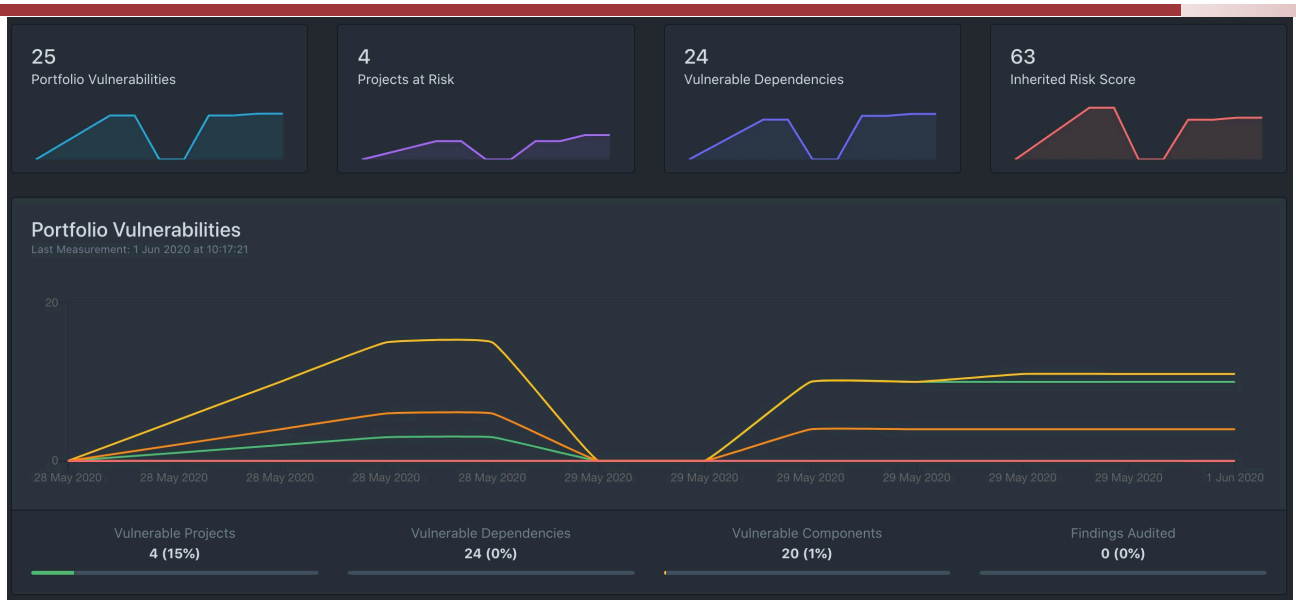


Figure 32: Vulnerabilities Portfolio

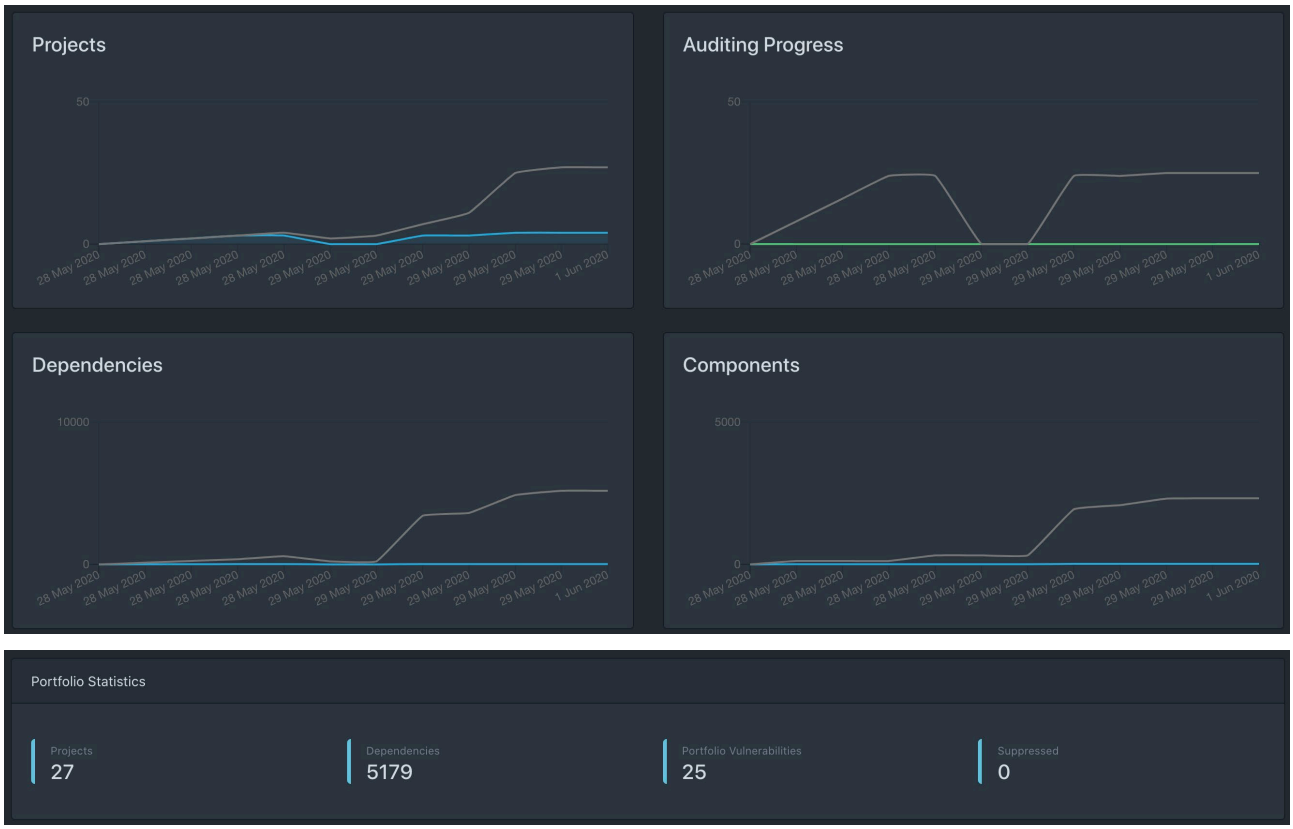
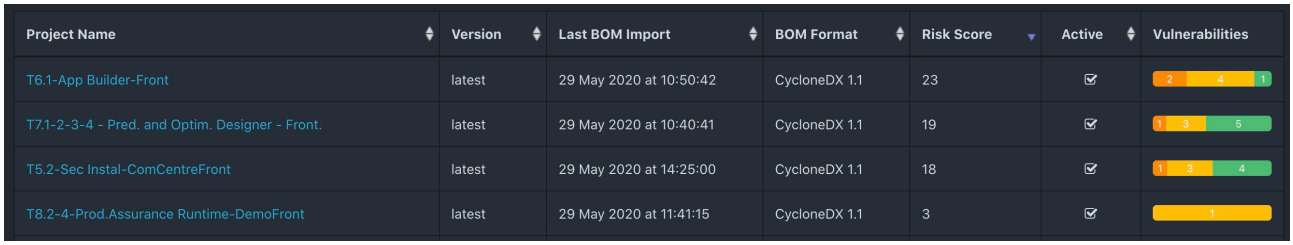


Figure 33: Dependencies and Portfolio Statistics

Figure 34 lists the projects with most dependency's vulnerabilities found. The rest of the projects have no known vulnerabilities.



Project Name	Version	Last BOM Import	BOM Format	Risk Score	Active	Vulnerabilities
T6.1-App Builder-Front	latest	29 May 2020 at 10:50:42	CycloneDX 1.1	23	☑	2 4 1
T7.1-2-3-4 - Pred. and Optim. Designer - Front.	latest	29 May 2020 at 10:40:41	CycloneDX 1.1	19	☑	1 3 5
T5.2-Sec Instal-ComCentreFront	latest	29 May 2020 at 14:25:00	CycloneDX 1.1	18	☑	1 3 4
T8.2-4-Prod.Assurance Runtime-DemoFront	latest	29 May 2020 at 11:41:15	CycloneDX 1.1	3	☑	1

Figure 34: Maintainability overview

25 vulnerabilities are detected in 4 components, of which:

- 4 are considered high-risk
- 11 are medium
- 10 are low

These vulnerabilities are related to the following CWEs (a list of software and hardware weaknesses types):

- CWE-400 Uncontrolled resources consumption
- CWE-471 Modification of assumed-immutable data (MAID)
- CWE-79 Improper neutralization of input during web page generation ('Cross-site Scripting')
- CWE-116 Improper encoding or escaping of output
- CWE-1022 Use of web link to untrusted target with window.opener access

The recommendation is the same for all these vulnerabilities found: Upgrade to a newer version of the dependency.

Dependencies which have known vulnerabilities are identified in Figure 35.

Name	Version	License	Used By	Risk Score	Vulnerabilities
jquery	3.3.1	MIT	1	6	2
http-proxy	1.17.0	MIT	1	5	1
http-proxy	1.18.0	MIT	2	5	1
lodash	4.17.11	MIT	1	5	1
url-parse	1.2.0	MIT	0	5	1
url-parse	1.0.5	MIT	0	5	1
bootstrap	4.1.3	MIT	1	3	1
acorn	6.1.1	MIT	1	3	1
acorn	5.7.3	MIT	3	3	1
quill	1.3.3	BSD-3-Clause	1	3	1
acorn	6.4.0	MIT	2	3	1
acorn	7.1.0	MIT	1	3	1
quill	1.3.6	BSD-3-Clause	0	3	1
extend	3.0.1	MIT	0	3	1
bootstrap	3.3.7	MIT	0	3	1
yargs-parser	11.1.1	ISC	2	1	1
minimist	0.0.8	MIT	2	1	1
yargs-parser	13.1.1	ISC	2	1	1
yargs-parser	5.0.0	ISC	1	1	1
minimist	1.2.0	MIT	3	1	1

Figure 35: ZDMP Dependencies vs. Vulnerabilities

OWASP Dependency Track also explains the vulnerabilities, an example is provided in Figure 36 and Figure 37.

Name	Published	CWE	Severity
NPM 1518	30 Apr 2020	CWE-79 Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	Medium
NPM 796	2 Apr 2019	CWE-471 Modification of Assumed-Immutable Data (MAID)	Medium

Figure 36: jQuery vulnerabilities

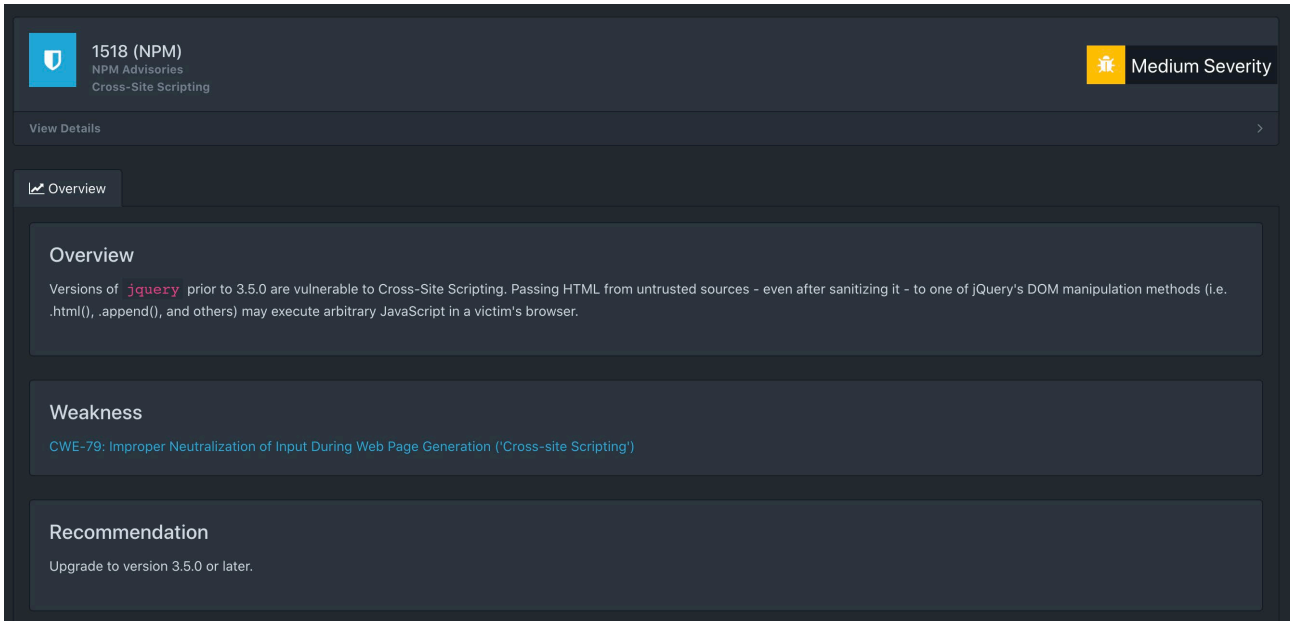


Figure 37: jQuery Vulnerabilities Detail

Every partner has access to these reports in ZDMP, so they can know the vulnerabilities of their dependencies, and the security recommendations to patch them.

3.1.5 Risk and Data Issues

The risk status associated purely with Cross WP Technical Management in the project are as follows. There are multiple other risks records in the D1.4.x series WP5-8 Technical Management Reports:

Risk Matrix

Risk	WP Likelihood % Impact 💣	Critical risks for implementation - Proposed risk-mitigation measures -	Current Status
Delay in the integration of components due to a high complexity	WP: 5/6/7/8 %: Medium 💣: High	Integration is considered as a core issue in the project which needs special attention du e to the complexity of the task. As a countermeasure to overcome this risk, technical meetings between core components must be arranged, and the preparation of a common shared infrastructure for integration testing be set it up (check Section 1.3)	The consortium is working already in the mitigation measures. **Risk future change:** % No change 💣 No change
Lack of development coordination due to COVID-19	WP: 5/6/7/8 %: Medium 💣: High	Physical, face to face meetings are necessary where strong coordination between teams is required, as is the case for the ZDMP development work. COVID-19 has made impossible to meet physicaly and this may have a direct underperforming impact in ZDMP teams. As mitigation measures regular virtual sprints meetings, WP, and plenary meetings must be arranged.	The consortium is working already in the mitigation measures. **Risk future change:** % No change 💣 No change

4 Conclusions

There are several issues which makes ZDMP a very complex project to be Technically Managed; the most prominent are:

- Manage a large group of software developers and analysts coming from very different institutions, and geographically spread over Europe
- Integrate and further develop a considerable number of existing software pieces, quite heterogeneous, both in terms of functionalities and maturity
- Recently to handle the COVID-19 issue which made it not possible to have additional face to face meetings for, eg integration of components

However, a well-defined governance structured for the project was in place since the beginning, so clear roles and responsibilities were defined and that help in the development work.

Furthermore, the basis established in M9 for the software development work, and D021 Methodology Document, has worked well. Of course, the processes are subject to improvement and in fact these are constantly evolving.

ZDMP is making use of state of the art tools (such as GitLab and Sonar) to enforce the procedures established, and to make life easier to developers and integrators working in the project, taking maximum benefit of its resources.

As a result, early prototypes of all the components are offered already in M18 of the project, and they come with a comprehensive documentation, including work performed so far, plan for next periods, and a set of public documentation that will be a strong dissemination tool for the project.

M18 is just the initial formal delivery phase and there will be several more. In addition, so far ZDMP has, as planned, focussed on components. This is just the beginning and in new phases ZDMP will start working on the zApps to be integrated in the pilots and offering its developments to the outside community via the open calls.

Annex A: History

Document History	
Versions	V0.0.1 • First tentative TOC and initial content in section 0. V0.0.2 • Initial content on section 1 V0.0.3 • Content to Sections 3.1.1 and 3.1.4 V0.0.4 • General improvement and new contents in many sections V0.0.5 • Content to section 2 V0.0.6 • Content to section 3 V0.0.7 • General updates and content in many sections V0.0.8 • Revision of sections 2 and 3, content and update in most sections V1.0.0 • Additions and modifications from Project Manager (PM) V1.0.1-4 • Modifications based on PM comments V1.0.5 • PM Final review and submitted version V1.0.5A • PM Final review and submitted version
Contributions	ITI: • Francisco Barrena • Santiago Cáceres ICE: • Stuart Campbell

Annex B: References

None.

Annex C: Security Logging, Traceability and Event Tracking

Purpose and motivation

For security auditing ZDMP assets and users' activity will be monitored and tracked. Both manual actions (eg Login) and component actions/activity need to be recorded. This document presents a first proposal on how logging and event tracking can be managed in ZDMP.

Events and Activity to be tracked

ZDMP will enable the auditing of following activities:

Events related to Authentication and Authorization actions. User actions related to Auth processes (eg creation of new users, clients, successful logins, logouts, creation of new clients to be secured) will be recorded in the auth database (part of the T5.2 Secure Authentication and Authorization). Figure 1 depicts what actions need to be recorded regarding the Authentication and Authorization server.

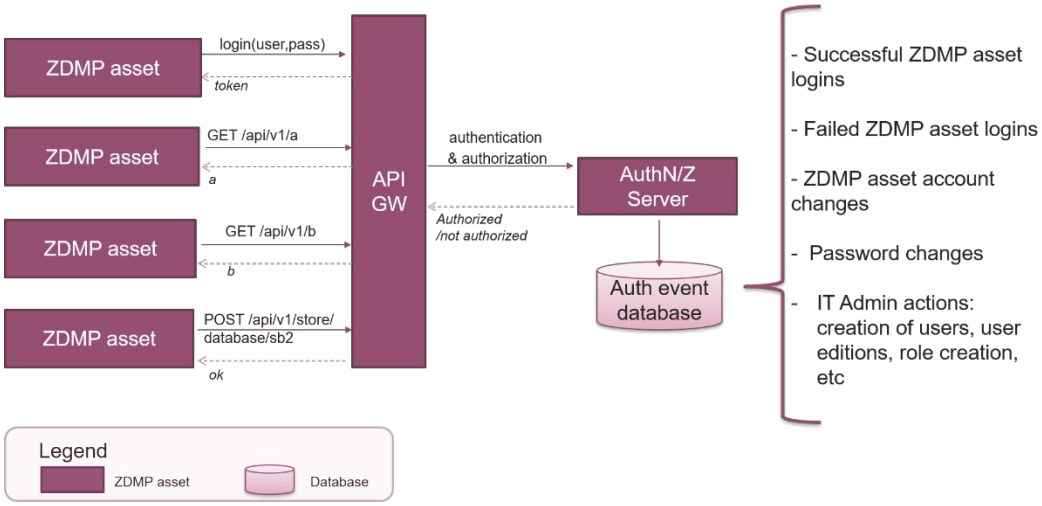


Figure 1. Overview of Authentication and Authorization server events tracking

System logs and activity at host level. ZDMP Assets (ie services) run in a host machine. Monitoring the log files generated by applications running at the host level and creating a historical record of activities and functions, can allow to quick searches for anomalies, and signing an intrusion may have occurred.

This can be achieved by the leveraging modern HIDS (Host Intrusion detection Systems), such as Wazuh [1]. As part of docker images ZDMP Assets would need to install a Wazuh agent [2] in charge of sending the logs to a central log database (ElasticSearch [4] recommended) where later on Kibana can be used for monitoring and alerting.

Hence, all ZDMP components would need to include a Wazuh agent for every system they provide to the ZDMP architecture (servers, docker images, databases, etc). Additionally, the T6.2 Storage could be involved so as to serve the ElasticSearch database where these logs would be held, and T6.4 Monitoring and alerting would deploy Kibana as part of the monitoring of the logs, and alerting of suspicious activity.

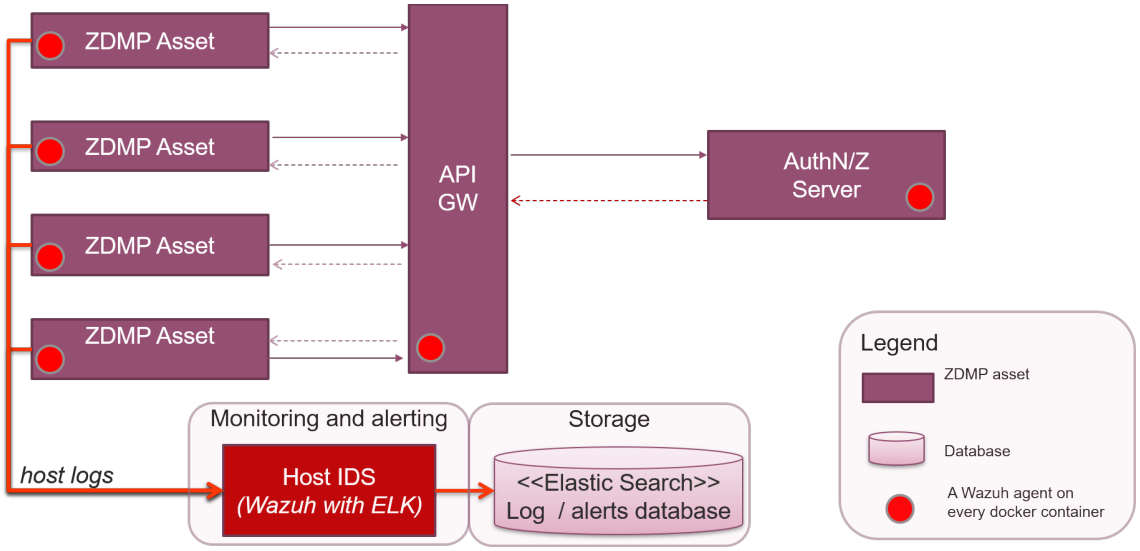


Figure 2. Overview of Host IDS in ZDMP platform

ZDMP assets' activities. In addition to system logs, ZDMP assets can also be tracking events based on users' actions. To record these types of events (eg a ZDMP user rated a given zApp in the Marketplace) the message bus can be leveraged, and enable a specific topic, **"/traceability",** where all ZDMP assets can publish their events. Figure 3 below depicts this scenario.

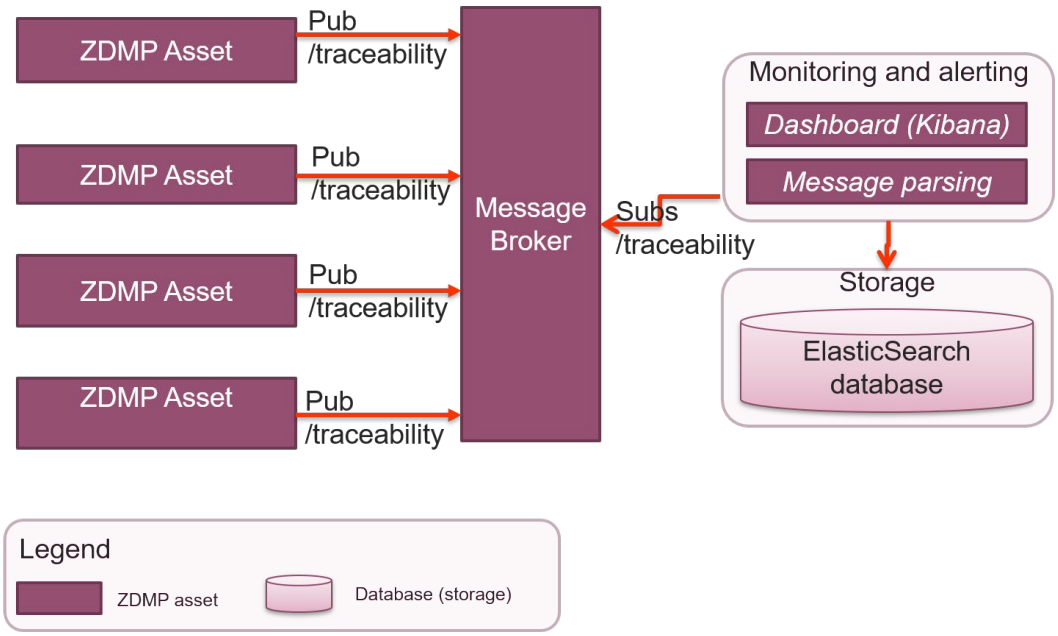


Figure 3. Overview on ZDMP components publishing events via Message Broker, for event tracking/logging

The payload to publish to */traceability* topic can be as follows:

{

"**timestamp**": 1589918550 , *// UTC timestamp in milliseconds since epoch*

"**userId**": "20941-079714*", // the unique ID of the user (if applicable)*

"**clientId**": "marketplace", *// componentID:name*

"**eventType**": "ZAPP_RATING", *// defined by each component*

"**eventDescription**": "Brief description of what means that event", *//to be defined by each component*

"**payload**": { } // defined by each component

}

Once published on this topic, the monitoring and alerting can process this data, and make use of the storage to store the events. ElasticSearch [4] or InfluxDB [5] databases are recommended for this purpose, as these are databases though to hold great amount of time-series data. Additionally, they come with visualization tools such as Kibana [6] and Grafana [7], respectively, that can be used as means of monitoring in real time.

References

[1] Wazuh https://wazuh.com/

[2] Wazuh containers for Dockers https://github.com/wazuh/wazuh-docker

[3] Deploying wazuh on Kubernetes https://github.com/wazuh/wazuh-kubernetes

[4] ElasticSearch https://www.elastic.co

[5] InfluxDB https://www.influxdata.com

[6] Kibana www.elastic.co/kibana

[7] Grafana https://grafana.com/

Annex D: Functional Requirements Status

The following tables show the functional progress per component in the WP5 mapped with the requirements with the progress represented as a % of the total envisioned for the function.

WP5

T5.1 - Data Acquisition

Functional requirement	Description	Status	Progress	Comments
T51A001 - Receive asynchronous data from Data Source	Receive data pushed by a data source based on an event configured on it	Working	50%	
T51A002 - Send data for asynchronous data acquisition	Send data to the zApps or other assets through a push mechanism or to the Services and Message Bus Component through pub/sub mechanism	Working	30%	
T51A003 - Receive data from Data Source (via polling process)	Periodically queries data from data sources that do not have a push-based mechanism	Working	70%	
T51A004 - Receive synchronous data from Data Source	Will query data from data sources that do not have a push-based mechanism	Working	60%	
T51A005 - Send data for synchronous data acquisition	Will provide an API to interact with zApps or other component for synchronous data acquisition through pulling operations	Working	50%	
T51A006 - Send command to Data Source	Will be able to send command on specific private protocols to data sources	Working	60%	
T51A007 - Register Data Source on ZDMP	Set-up an existing data source as a source of information	Working	80%	
T51A008 - Register asynchronous sensors or other IIoT items of Data Source	Register sensors of a given data source as asynchronous capable	Working	50%	
T51A009 - Register synchronous sensors or other IIoT items of Data Source	Register sensors of a given data source as synchronous capable	Working	50%	
T51A010 - Configure Data Source as command receiver	Configure a data source as a command receiver	Working	70%	
T51A011 - List existing Data Sources already configured	List existing registered and configured data sources	Working	60%	
T51A012 - Filter list of Data Sources according to name and type	Filter the configured list of data sources by name and type	Working	50%	

Functional requirement	Description	Status	Progress	Comments
T51A013 - Sort list of Data Sources by columns	Sort the configured list of data sources by any of the columns	Working	40%	
T51A014 - Check Data Source status	Assess the status of the data sources	Working	80%	
T51A015 - Check sensor or other IIoT item status	Assess the status of each data source's sensor or other IIoT item	Working	50%	
T51A016 - List existing Data Source Adapters	List existing installed adapters getting the detail on version, etc	Working	50%	
T51A017 - Filter list of Data Source Adapters by Name	Filter the existing installed adapters by name	Working	50%	
T51A018 Log messages from Data Acquisition component	Send logs of information, warnings, and errors to the Services and Message Bus	Working	60%	
T51A019 - Providing data model	Provide the data model to the Data Harmonization Component	Working	30%	

T5.2 - Secure Installation

Functional requirement	Description	Status	Progress	Comments
T52A001 - Access to roles and policies	The administrator must access user roles and policies of ZDMP assets and users	Working	40%	API endpoints to access to roles is possible, working towards policies. Lack UI
T52A002 - CRUD roles	Perform CRUD operations on roles.	Not initiated	40%	API endpoints available and testing going on, but lacking UI
T52A003 - CRUD user	The administrator must be able to perform CRUD operations on users	Working	50%	API endpoints available and testing going on, but lacking far the command centre API can access users and their data
T52A004 - CRUD security policies	The administrator must be able to perform CRUD operations on security policies	Not initiated	0%	API endpoint and UI to do

Functional requirement	Description	Status	Progress	Comments
T52A005 - Download zApp package	The installation Broker Service must request the download of a zApp package	Not initiated	0%	
T52A006 - Verify zApp signature	The installation Broker Service must check the integrity go the downloaded zApps and ensure the identity of the zApp developer	Not initiated	0%	
T52A007 - Create zApp permissions	The installation Broker Service must request the creation of the different permissions required by the zApp, which are specified in the manifest	Not initiated	20%	Script to process manifest ongoing. Also, remaining is the creation of policies once manifest is processed
T52A008 - Deliver zApp package	The installation Broker Service must be able to send a zApp package to the requesting ZDMP asset and to the Application Runtime component	Not initiated	0%	

T5.2 - Secure Communication

Functional requirement	Description	Status	Progress	Comments
T52A013 - Issue New certificates	New client certificates are created. These certificates include the details that permit the identification of the subject (physical device, gateway or server).	Working	90%	Beta version, requires integration API with security command centre for credentials tokenization
T52A014 - Install Root certificates	New root and intermediate certificates are installed, so that they can be used to sign and verify other (client) certificates	Working	90%	Discussion on root certificates and public CA needed
T52A015 - Revoke certificates	Certificates that are out-of-date or that have been compromised are registered (in Certificate Revocation Lists, CRLs), to guarantee that their use is discontinued.	Working	70%	
T52A016 - Renew certificates	The Security Command Centre can request clients to renew their certificate to avoid revocations.	Working	60%	
T52A017 - Certificate retrieval	Where stored certificates are retrieved based on their identifying details (eg certificate subject).	Working	90%	Discussion on details available for WP6.2 needed

T5.2 - Secure Authentication and Authorisation

Functional requirement	Description	Status	Progress	Comments
T52A009 - Create a new account	A ZDMP asset must be able to create a new account associated with a ZDMP asset	Working	70%	Accounts can be of two types: organizational account, and a ZDMP asset within an organization account. the latter is ongoing, and the former completed
T52A010 - Authentication	A ZDMP asset must be authenticated thorough this module and get associated access token	Testing	85%	May need additional claims to be embedded in the token, but the server is issuing them correctly. Efforts also to connect with the API Gateway are ongoing
T52A011 - Resource access authorisation	A ZDMP must be get authorized to a protected resource it can access	Working	60%	Tested with security command centre client and secure authentication module.
T52A012 - React to a suspicious activity	The Detection Intrusion service must detect, log, and alert any suspicious activity	Working	40%	The identity module logs user's activity (eg login), can detect user changed context

T5.3 Data Harmonisation Designer and Run-time

Functional requirement	Description	Status	Progress	Comments
T53A001 Connect to Database	Open filesystem and access the schema of the data source (both for source and target schemas) So that the schema can be loaded and, thus, the mappings can be performed	Finished	100%	
T53A002 Read XML	interprets eg XML schema files So that the schema can be loaded and, thus, the mappings can be performed	Finished	100%	

Functional requirement	Description	Status	Progress	Comments
T53A003 Read CSV	Interprets CSV schema files So that the Transformation engine can read the transformation steps determined in the Map	Finished	100%	
T53A004 Read JSON	Interprets JSON schema files So that the Transformation engine can read the transformation steps determined in the Map	Finished	100%	
T53A005 Read TXT	Interprets plain text schema files So that the Transformation engine can read the transformation steps determined in the Map	Waiting	90%	
T53A006 Read XLS	Interprets XLS schema files So that the Transformation engine can read the transformation steps determined in the Map	Finished	100%	
T53A007 Read MySQL	Interprets MySQL schema files So that the Transformation engine can read the transformation steps determined in the Map	Finished	100%	
T53A008 Display UI	Show Mapping UI so that the source schema can be displayed and, thus, the mappings can be performed	Finished	100%	
T53A009 Load Source Schema	Loads source schema So that the source schema can be accessed	Finished	100%	
T53A010 Display Source Schema	Display source schema So that the source schema can be viewed	Finished	100%	
T53A011 Analyse Source Schema	Analyses source schema So that the source schema can be manipulated	Testing	75%	
T53A012 Connect to ontology	Connects to domain (or ZDMP) ontology So that alternative concepts can be suggested for the concepts present in the source schema	Testing	75%	
T53A013 Suggest Semantic Concepts for Source Schema	Suggests alternative (or linked) concepts to the concepts present in the source schema So that a crowdsourced domain (or ZDMP) ontology can be populated	Testing	75%	
T53A014 Display UI	Show Mapping UI So that the target schema can be displayed and, thus, the mappings can be performed	Finished	100%	
T53A015 Load Target Schema	Loads target schema So that the target schema can be accessed	Finished	100%	
T53A016 Display Target Schema	Display target schema So that the target schema can be viewed	Finished	100%	
T53A017 Analyse Target Schema	Analyses target schema So that the target schema can be manipulated	Testing	75%	
T53A018 Connect to Ontology	Connects to domain (or ZDMP) ontology So that alternative concepts can be suggested for the concepts present in the target schema	Testing	75%	
T53A019 Suggest Semantic Concepts for Target Schema	Suggests alternative (or linked) concepts to the concepts present in the target schema So that a crowdsourced domain (or ZDMP) ontology can be populated	Testing	75%	

Functional requirement	Description	Status	Progress	Comments
T53A020 Connect to the Storage	Connect to the Storage with the credentials as directed by the T5.2 Secure Authentication/Authorisation component So that the Maps can be read, searched, and filtered after save	Working	50%	
T53A021 Read file	Read file So that the Maps can be loaded into the Data Harmonisation Designer component	Working	50%	
T53A022 Search Map	Search map So that the Maps can be searched into the Storage component	Working	50%	
T53A023 Filtering in the Storage	Apply filters So that the Maps stored in the Data Storage can be filtered according to user specified criteria	Working	50%	
T53A024 Preview Map	Preview map So that the Maps can be graphically previewed before loading into the Data Harmonisation Designer component	Working	50%	
T53A025 Annotate Map	Annotate map with metadata So that the Maps can be searched and filtered with this metadata as parameters	Working	50%	
T53A026 Publish Map	The map is published to a computer readable format So that the Maps can be wrapped and become an executable service	Working	50%	
T53A027 Persist Map	Save the map in the Storage So that the Maps can be re-used, retrieved, removed, searched, and filtered	Working	50%	
T53A028 Search Map	Search map So that the Maps can be searched into the Storage component	Working	50%	
T53A029 Delete Map	Remove map Checks that the persisted Maps in the Storage can be removed from it	Working	50%	
T53A030 Annotate Service	Annotate map for publication So that the to-be deployed Map can be easily found in the Storage	Working	50%	
T53A031 Create Service	Create self-executing service So that the deployed Map can be executed as a stand-alone service	Testing	90%	
T53A032 Deploy Service	Create Docker package So that the deployed Map can be executed and scalable if necessary, as a stand-alone service	Testing	90%	
T53A033 Connect to the Storage	Connect to Marketplace with the credentials as directed by the T5.2 Secure Authentication/Authorisation component So that the Transformation Services (ie the deployed maps) can be published	Working	50%	
T53A034 Publish Deployed Map	Publish deployed map (ie transformation service) So that the Transformation Service can be sold to ZDMP Users and re-used within eg the Process Orchestration	Working	50%	
T53A035 Get Related Data	Obtain data with internal relationships to a given concept passed as parameter So that the Data Harmonisation Designer can make use	Working	50%	

Functional requirement	Description	Status	Progress	Comments
	of the “wisdom of the crowd” for developing the maps			
T53A036 Get Routine	Obtain the internal relationships between a set of given concepts passed as parameters So that the Data Harmonisation Designer can make use of the “wisdom of the crowd” for developing the maps	Working	50%	
T53A037 Add Related Data	Add a new concept that is linked to a given concept passed as parameter So that the Data Harmonisation Designer can provide feedback to the “wisdom of the crowd” for developing future maps	Working	50%	
T53A038 Add Routine	Add a new link between two given concepts, passed as parameters So that the Data Harmonisation Designer can provide feedback to the “wisdom of the crowd” for developing future maps	Working	50%	
T53A039 Update Related Data	Update a concept that is linked to a given concept passed as parameter So that the Data Harmonisation Designer can provide feedback to the “wisdom of the crowd” for developing future maps	Working	50%	
T53A040 Update Routine	Update a given link between two concepts, passed as parameters So that the Data Harmonisation Designer can provide feedback to the “wisdom of the crowd” for developing future maps	Working	50%	
T53A041 Get Concept	Get concepts from the ontology So that the Data Harmonisation Designer can make use of the ZDMP Ontology for developing the maps	Working	50%	
T53A042 Add Concept	Add concepts to the ontology So that the Data Harmonisation Designer can make use of the ZDMP Ontology for developing the maps	Working	50%	
T53A043 Update Concept	Update concepts in the ontology So that the Data Harmonisation Designer can make use of the ZDMP Ontology for developing the maps	Working	50%	
T53A044 Delete Concept	Remove concepts from the ontology So that the Data Harmonisation Designer can update the ZDMP Ontology for future developing the maps	Working	50%	
T53A045 Reasoning	Reason So that the Data Harmonisation Designer can make use of the ZDMP Ontology for developing the maps	Working	90%	
T53B001 Get invocation	Get invocation request from Service Call So that the transformation engine can execute the right transformation service	Working	50%	

Functional requirement	Description	Status	Progress	Comments
T53B002 Connect to Store	Connect to ZDMP Store with the credentials as directed by the T5.2 Secure Authentication/Authorisation component So that the transformed data can be stored	Working	50%	
T53B003 Unpack Routines	Unpack a mapping routine set So that the Transformation engine can read the transformation steps determined in the Map	Working	50%	
T53B004 Read Data	Reads source data as input parameter from the invocation So that the input data can be transformed by executing the transformation services	Working	50%	
T53B005 Transform	Transforms the data So that the routines of the mapping are executed	Finished	100%	
T53B006 Push Transformed Data	Pushes transformed data back to the calling zApp So that the routines of the mapping are executed	Working	50%	
T53B007 Store Transformed Data	Store the transformed data So that the transformed data is accessible without having to re-execute the transformation service again	Working	50%	
T53B008 Submit Usage Data	Submit usage data So that the platform can make use of this data for monitoring purposes	Working	50%	

T5.4 - Orchestration Designer and run-time

Functional requirement	Description	Status	Progress	Comments
T54A001 Process Explorer	Open Process Explorer to browse available processes To provide already created processes and services to the orchestration designer	Finished	100%	
T54A002 Search / Filter process models	Search/filter process models So the user can find a process to be added to the orchestrator	Finished	100%	
T54A003 Create / Open existing process	Create a new process/Open existing process To access previously created processes or create new ones	Finished	100%	
T54A004 Save process model	Save process model changes to local storage or platform storage Automatically save changes made by the user to the current process model to limit the loss of work	Finished	100%	Auto save to local docker MongoDB instance
T54A005 Diagram: connect figures	Drag and drop elements and connect them together Allow the user to design the BPMN process model	Finished	100%	
T54A006 Update element common properties	Update common element properties by using properties panel To allow customisation of the connections to components or zApps	Working	80%	
T54A007 Marketplace explorer	Allow the user to add a service task from the marketplace to the current diagram Use a service/asset from the marketplace as part of the process model being designed	Finished	100%	Connected to WASP marketplace

Functional requirement	Description	Status	Progress	Comments
T54A008 Validate process	Provides the list of errors/attributes that must be set by the user Ensure the process model and all its elements have all required input	Working	50%	Skeleton validators already in place
T54A009 Make valid connections	Allow only certain toolbox elements to be connected. Following the BPMN specification, certain elements cannot be connected, such as an end event and a message start.	Finished	100%	
Requirements Fields Cross App Functionality – see Additional Issues T54A010	Gateway condition designer Allows the user to create condition expressions that will be evaluated at runtime	Finished	100%	
Create conditions that will affect the flow in runtime Requirements Fields Cross App Functionality – see Additional Issues	T54A011 Message pub/sub designer	Not started	0%	
Allows the user to send a message through the service bus (throw), or wait for a specific one (catch) Provide message support as part of the BPMN specification	Requirements Fields Cross App Functionality – see Additional Issues T54A012	Not started	0%	
Remove selected element Remove the selected element	Standard editing function in designers Requirements Fields Cross App Functionality – see Additional Issues	Finished	100%	
T54A013 Delete process model	Delete process model Allow user to delete process models	Not started	0%	
Requirements Fields Cross App Functionality – see Additional Issues T54A014	Deploy process model Deploy process model to process engine in the platform on to the marketplace	Not started	0%	
The engine can be executed on the process by any other privileged user Requirements Fields Cross App Functionality – see Additional Issues	T54A015 Get list of process instances	Working	50%	
Get the list of process instances Enable a user to assess which	Requirements Fields Cross App Functionality – see Additional Issues	Working	50%	

Functional requirement	Description	Status	Progress	Comments
processes instances are available				
T54B001 Deploy to Camunda Engine	Connect to Process Execution Manager Prepare deployment for process model	Working	50%	
T54B002 New Process Instance	Prepare process model for execution So that process can be executed	Working	50%	
T54B003 Request Services	Connect to the Service Manager Get metadata information for each service used in the process model for further execution	Finished	100%	Connected to WASP marketplace
T54B004 Launch and start process (and get metrics)	Launches the process and start getting metrics A new instance is created, and metrics start being collected	Finished	100%	
T54B005 Manage process instances runtime values	Manage process instances and get/set process instance variable values So, run-time values can be read and write	Not started	0%	
T54B006 Manage process instances runtime values	Compile external service metadata so they can be called as part of the process execution So that a process is effectively helped by the execution of 3rd party services	Not started	0%	
T54B007 Start process instance	Start a new instance from an existing process model Manually trigger instance start from process engine	Not started	0%	
T5BA008 Stop / Suspend / Resume process instance	Stop/Suspend/Resume process instances User can interact with process instances	Working	50%	

T5.4 - Monitoring and Alerting

Functional requirement	Description	Status	Progress	Comments
T54C001 - CRUD KPI	Allow the user to define a KPI and its filtering expressions.	Working	20%	
T54C002 - Get KPI List	Lists existing KPIs, and its filtering expression.	Working	20%	
T54C003 - Monitor KPI data	Monitor incoming KPI data from Message Bus and Orchestration Run-time.	Working	45%	
T54C004 - CRUD Alert	Create, read, update, and delete alerts, its conditions and the users and components that should be alerted.	Working	15%	
T54C005 - Alert users	Alert users and components in case a KPI value is not in the desired range.	Working	10%	
T54C006 - React on Alert	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.	Not started	0%	

T5.5 - Autonomous Computing

Functional requirement	Description	Status	Progress	Comments
T55A001 - Subscribe to KPI	Subscribe to an existing KPI to react on it and / or push the data to Storage component for historic data collection and analytics reasons	Working	25%	
T55A002 - Unsubscribe to KPI	Remove subscription to a KPI previously subscribed	Working	25%	
T55A003 - Get KPI Data	Get KPI data, such as values over time, starting processes, etc.	Working	10%	
T55A004 - CRUD Autonomous Process for KPI	Define / Update / Remove conditions and actions	Working	30%	
T55A005 - Get Autonomous Process for KPI	Get the defined Autonomous Processes for a KPI	Working	20%	
T55A006 - Monitor KPI's values and Start Actions	Receive the KPI data/values from the message bus and apply the rules to see if any process must be started.	Working	65%	

T5.5 - Distributed Computing

Functional requirement	Description	Status	Progress	Comments
T55B001 - Compute Task	Compute assigned Task to optimize performance, reducing overall processing time, using Cloud, Mist, Edge and Fog computing.	Working	5%	
T55B002 - CRUD Resources Location	Define / Update / Remove resources location in a scope of Cloud, Mist, Edge, and Fog Computing.	Working	20%	
T55B003 - Get Map of Resources Location	Get the Map of resources location to adhere to the spatial tagging of computing resources, to execute tasks in the proper environment.	Working	5%	
T55B004 - Get Resource Location	Get the defined location of a resource to Adhere to the spatial tagging of computing resources, to execute tasks in the proper environment.	Working	5%	

T5.6 - AI Analytics Designer

Functional requirement	Description	Status	Progress	Comments
T56A001 Connect to data source	Opens filesystem of the data source and access the database connections, so that the schema can be loaded and, thus, the data extraction can be performed	Working	60%	
T56A002 Read SQL database	So that the Historic API can read the records from database	Working	60%	
T56A003 Read JSON	The JSON schema is successfully interpreted and retrieved	Finished	100%	
T56A004 Read XML	Interprets XML schema files so that the schema can be loaded and, thus, the data extraction can be performed	Finished	100%	

Functional requirement	Description	Status	Progress	Comments
T56A005 Read CSV	Interprets CSV schema files, so that the Historic API can read the data	Working	60%	
T56A006 Transform in K/V	Extracts values from records and pair them with keys	Finished	100%	
T56A007 Display UI	Displays data retrieved and K/V strings created	Finished	100%	
T56A008 Write K/V	Stores in memory structures with K/V strings	Working	60%	
T56A009 Build Supervised Algorithm	Creates supervised algorithms	Finished	100%	
T56A010 Train Supervised Algorithm	Runs supervised algorithms with historic data.	Finished	100%	
T56A011 Validate Model	Runs the trained machine learning model to estimate the performance.	Finished	100%	
T56A012 Build Unsupervised Algorithm	Creates unsupervised algorithms	Finished	100%	
T56A014 Display Model Validation	Displays graphical representations of data validated by the machine learning model	Not started	0%	
T56A015 Save Model	Saves the binary machine learning model in the repository	Working	80%	
T56A016 Load Model	Loads the binary machine learning model from the repository	Not started	0%	
T56A017 Convert to POJO	Transforms a machine learning model in a Plain Old Java Object (POJO)	Working	50%	
T56A018 Convert to MOJO	Transforms a machine learning model in a Model Object, Optimised (MOJO)	Working	50%	
T56A019 Deploy model	Deploys POJO or MOJO objects using Deploy API	Not started	0%	This task will be managed by T5.6 AI Analytics Runtime
T56A020 Upload model	Uploads a machine learning model to Marketplace, using Marketplace API	Not started	0%	This task will be managed by T5.6 AI Analytics Runtime
T56A021 Upload Script	Uploads a Python script to Marketplace, using Marketplace API	Not started	0%	This task will be managed by T5.6 AI Analytics Runtime
T56A022 Upload Library	Uploads a library to Marketplace, using Marketplace API	Not started	0%	This task will be managed by T5.6 AI Analytics Runtime

T5.6 – AI Analytics Runtime

Functional requirement	Description	Status	Progress	Comments
T56B001 - Connect to marketplace	Access Marketplace and open machine learning models page	Waiting	0%	Waiting for Marketplace component
T56B002 - Import model	Find machine learning model and import into Deployer module	Working	50%	Models based on PKL and ONNX ready. Working on H20.ai based models
T56B003 - Schedule model	Select a model and program it for running at a specific date and time	Working	70%	Boilerplate and architecture finished. Coding a task framework based on Cron Expressions
T56B004 - Connect to zApps Data source	Find a proper zApps and connect to it for access their data source	Waiting	20%	Waiting for integration with other ZDMP's components. Proof of concept using smoke component.
T56B005 - Connect to storage data source	Connect to Storage to a specific data source	Waiting	10%	Waiting for integration with Storage component. Connected to local databases to continue the development
T56B006 - Run model	Run a machine learning model against real production data	Waiting	80%	Waiting for integration with other ZDMP's components. The model is deployed into production, waiting to link it with Message and Service Bus and Storage component
T56B007- Model control UI	Displays data, status and anomalies detected on a running model	Working	20%	Minimalistic UI
T56B008 - Notification issue	Send notifications other component about a running model	Waiting	0%	Waiting for Monitoring

Functional requirement	Description	Status	Progress	Comments
				and Alerting component
T56B009 - Analytics queries	Creates analytics queries on data processed by the machine learning models	Working	20%	
T56B010 - Display graphs	Display graphs based on analytics queries	Not started	0%	
T56B011 - Analytics storage	Stores the data computed with analytics functions	Working	60%	Using MinIO repositories
T56B012 - Automatic API wrapping	Wraps an AI's model inside a standard API	Finished	100%	New feature added
T56B013 - Cluster management and administration UI	Shows activity and status of the cluster which runs the AI models	Finished	100%	New feature added

WP6

T6.1 - Application Builder

Functional requirement	Description	Status	Progress	Comments
T61A001 Get List of Stored Configurations	Browse existing configurations to model the SDK	Not started	0%	
T61A002 Get zApps APIs	Retrieves the APIs used by the selected zApp	Not started	0%	
T61A003 Get ZDMP Asset APIs and Manifests	Retrieves the API and manifest files for the selected ZDMP asset	Not started	0%	
T61A004 Get Data Model definitions	Retrieves APIs and manifest files for the selected data model / design pattern	Working	20%	
T61A005 Get Configuration	Retrieves data for the selected configuration file stored in the Data Storage	Not started	0%	
T61A006 Retrieve Composition data	To provide all information needed for the zApp build	Working	60%	
T61A007 Validate Dependencies	Analyses received data corresponding to the development of a zApp and check the dependencies of the various involved modules	Not started	0%	
T61A008 Structure the Build Manifest	To ensure the process steps needed for performing the build make sense and are listed in an appropriate way	Finished	100%	
T61A009 Create Error Report	Uncaught exceptions are intercepted and afterwards transformed in a readable format	Finished	100%	
T61A010 Send Error Report Automatically	Errors are submitted to the Secure Business Cloud Backend to inform the developer about such errors	Finished	100%	
T61A011 Send Error Report Manually	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	Finished	100%	
T61A012 Forward Error Report	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	Finished	100%	
T61A013 Provide a Register form for new Users	Provides a form to register a new user in ZDMP	Finished	100%	
T61A014 Provide User Login Form	To give a common login mask including a behaviour template for authorised only actions	Finished	100%	
T61A015 Provide a button to reset a password	Provides a button which forwards to a form to reset the password	Finished	100%	
T61A016 Provide a form to reset a password	Provides a form to fill in only the email address of the user, which is already registered	Finished	100%	

Functional requirement	Description	Status	Progress	Comments
T61A017 Translate text	Provides an option to translate the current language in any other language	Finished	100%	
T61A018 Switch language	The user can choose between several (if provided) languages and select the intended language to use for this ZDMP Asset	Finished	100%	
T61A019 Provide UI Elements	Developers can take UI elements from the UI repository to use them in their ZDMP Assets, which are under development	Finished	100%	
T61A020 Show Notification	Notification will be shown on a display, in case of a triggered event.	Finished	100%	
T61A021 Dismiss Notification	Developers can add a dismiss functionality to make notifications disappear	Finished	100%	
T61A022 Remind Later	Developers can set a timer to appear and a notification after the timer is elapsed	Finished	100%	
T61A023 Manifest reading	Returns a manifest file from an installed driver	Not started	0%	

T6.2 - Security Designer

Functional requirement	Description	Status	Progress	Comments
T62A001 - Provide User Login Form	The login page is activated either by clicking on ‘Sign In’ link in the dropdown menu or by clicking on the Login button.	Working	60%	The software has an existing user management system which is being replaced with one backed by KeyCloak to be able to integrate with Enterprise authentication systems and SSO systems
T62A002 - Provide registration form for new users		Working	60%	Relates to T62A001
T62A003 - Create new model		Finished	100%	
T62A004 - List Models	A list of existing models is provided so that a user can choose which one to open.	Working	70%	The list of models exists but usability is poor. Working to improve the usability by making it appear more as a list of files with last-

Functional requirement	Description	Status	Progress	Comments
				modified date, etc.
T62A005 - Import Model	Import a model that the user has saved locally.	Finished	100%	
T62A006 - Export Model	Save a model locally.	Finished	100%	
T62A007 - Validate Security Model	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.	Finished	100%	
T62A008 - Generate Security Report	This function generates a detailed description of the threats and risks to the system.	Working	50%	Will provide human- and machine-readable reports. The format and content of the reports is in development.
T62A009 - 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.	Testing	80%	
T62A010 - 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.	Waiting	30%	The progress on this task depends strongly on the maturity of other components. Actively discussing with partners in order to finalise the integration of Security Designer and other ZDMP components.
T62A011 - Model construction	Selecting and adding assets to the canvas, adding relationships between assets, deleting assets/relations, and renaming assets.	Working	40%	The functionality works but usability is poor. Working to improve the usability.
T62A012 - Threat management	Threats may be resolved by selecting one or more controls. There are various panels provided to help show the user where to best place controls.	testing	80	

T6.2 - Marketplace

Functional requirement	Description	Status	Progress	Comments
T62A001 Connect to Marketplace	Connect to Marketplace and get user access rights set via T5.2 Security Run-time component, so that the user can access the UI and functionalities of the Marketplace	Not started	0%	
T62A002 User Negotiation	Users and application providers negotiate with each other about possible new ZD Assets and the conditions of the development, so that the users can request applications based on their specifications	Working	20%	Acceptance Criteria: Communication between users and developers facilitated.
T62A003 Search items	Browse and search for specific ZDMP assets, so that the user can examine assets and their individual information such as screenshots, usage fees, and reviews or ratings of other users.	Working	80%	Acceptance Criteria: Asset successfully found
T62A004 Upload items	Administrators and providers upload ZDMP assets, so that the users can search for specific assets.	Testing	20%	Acceptance Criteria: Asset successfully uploaded.
T62A005 Place Order	The user can make an order for a specific asset and save it in the Order Database, so that the user can buy the specific asset	Working	80%	
T62A006 Issue Invoice	The Invoicing module creates an invoice for the order saved, so that the user can pay for the chosen asset.	Working	70%	This would also call the license manager that an invoice was paid for so that the license manager can transfer from trial to paid licenses.
T62A007 Notification	User receives information regarding the items, so that the user can be informed about a specific item.	Not started	0%	
T62A008 Payment	User pays the invoice for the chosen asset, so that the buying process can be processed.	Not started	0%	Acceptance criteria: Payment successfully made
T62A009 Record Usage Data	Collect and store usage data for pay-per-use items, so that the pay-per-use items can be invoiced.	Not started	0%	
T64BN10 Product Information Management System	A new feature since the creation of the functional specification, a simple content management system to manage the static information about software (zApp) products. This is a backend UI to easily manage the content behind this, showing of the content will be done in the marketplace frontend.	Working	10%	

Functional requirement	Description	Status	Progress	Comments
T64BN11 License Managing System	A new feature since the creation of the functional specification, a license managing tool which can combine the information of the applications (see T64BN10) with information of the users and assign a license to this. Inherently no UI is needed, but could contain a simple web UI to allow manual changes for the ZDMP marketplace admin.	Working	10%	
T64BN12 Referral Affiliate Managing System	A new feature since the creation of the functional specification, a small web tool to manage generation of URLs to particular zApps purchase pages. If the user uses these URLs to purchase a zApp, points will be awarded to the referring user. The user can login into the referral affiliate manager to see, which of his links gained how many points and how many points the user collected so far. There is also a button which triggers a sell-out, where the marketplace shop system will pay the referral user based on the points he generated. The payment might be in form of Amazon coupons or other means of value.	Working	10%	

T6.2 - Storage

Functional requirement	Description	Status	Progress	Comments
T62B001 Connect to file repository	Connect to file system and opens filesystem of the data source	Working	50%	
T62B002 Connect to SQL database	Connect to SQL database for any database operation	Not started	0%	
T62B003 Connect to NoSQL database	Connect to SQL database for any database operation.	Not started	0%	
T62B004 Receive file command	Receive a valid file transfer command in order to a file operation can be perform.	Working	20%	
T62B005 Receive SQL command	Receive a valid SQL command in order to a database operation can be executed	Not started	0%	
T62B006 Receive NoSQL command	Receive a valid NoSQL command in order to a database command can be performed.	Not started	0%	
T62B007 Send binary data	Perform a file transfer command.	Working	50%	
T62B008 Send database records	Perform a database SQL command in order to get the result of the user's request.	Not started	0%	
T62B009 Send structured data	Perform a NoSQL command in order to get the result of the user's request.	Not started	0%	

Functional requirement	Description	Status	Progress	Comments
T62B010 Receive Management command	Receive a valid management command for performing a management operation.	Not started	0%	
T62B011 Send Management action	Execute a management command for performing an administration task.	Not started	0%	

T6.3 - Human Collaboration

Functional requirement	Description	Status	Progress	Comments
T63A001 Connect to Marketplace	Connect to Marketplace and open zApps page	Not started	0%	
T63A002 Search content	Browse and search for specific zApps from the Marketplace	Not started	0%	
T63A003 Download content	Download a specific zApp from the Marketplace	Not started	0%	
T63A004 Send information	Send specific information regarding an asset	Not started	0%	
T63A005 Receive information	Receive specific information on request	Not started	0%	
T63A006 Upload images	Upload specific image regarding an asset	Working	15%	
T63A007 Download images	Download specific image	Working	10%	
T63A008 Upload video	Store a video using Stream Manager and open a link for users	Working	80%	
T63A009 Join video conference	Create/join video conference	Working	80%	
T63A010 Get location	Connect to asset and get its location	Not started	0%	

T6.4 - Portal

Functional requirement	Description	Status	Progress	Comments
T64D01 Manage users	Users can login and manage their profiles and anything they have permission to.	Working	20%	
T64D02 Show appropriate content for a user's role	Users can show appropriate content based on their role. So, a developer receives different content compared to factory worker or to a factory manager.	Not started	0%	
T64D03 Getting a zApp UI and content	Users can access the UI and content of their zApps.	Not started	0%	
T64D04 Managing zApps	Administrators can manage (request install or uninstall) of zApps.	Not started	0%	
T64D05 Getting marketplace content	Users can can purchase zApps from the marketplace.	Not started	0%	

T6.4 - Service and Message Bus

Functional requirement	Description	Status	Progress	Comments
T64A001 Management of APIs	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	Finished	100%	
T64A002 API access tokens	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	Working	30%	The tokens are not stored in the Services API Management. Instead, the T5.2 Secure Authentication & Authorisation component will be integrated as an external authentication & authorisation server
T64A003 API function calls	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	Working	60%	APIs can be called using basic authentication via user-name/password or an API key. The integration with the T5.2 Secure Authentication & Authorisation component including the authorisation part is work in progress
T64A04 API logging	The T6.4 Services API Management logs each API call including the following information: • Timestamps • Calling entity • Accessed function • Encountered errors	Testing	60%	Logging functionality is available but not yet fully tested
T64A005 API documentation	The T6.4 Services API Management provides the functionality to enter documentation for each API	Testing	80%	
T64A006 (De-)Activation of APIs	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	Finished	100%	

Functional requirement	Description	Status	Progress	Comments
T64A007 Versioning of APIs	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	Testing	60%	Base functionality is available but not yet fully tested
T64A008 Mocking of APIs	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	Finished	100%	
T64B001 Message Bus access tokens	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	Not started	0%	
T64B002 Message Bus Topic Management	The T6.4 Message Bus provides the functionality to define and delete topics. Each topic must have a unique name, eg no duplicates are allowed. A topic is represented as a text string	Working	20%	
T64B003 Publishing messages on topics and subscribing to topics	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	Testing	60%	
T64C001 Optional integration of Enterprise Tier, Platform Tier, and Edge Tier components	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	Waiting	0%	Feature not yet needed

T6.5 - Inter-platform interoperability

Functional requirement	Description	Status	Progress	Comments
T65A001 Interconnection of ZDMP with external platforms	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	Working	10%	
T65B001 ZDMP and eFactory	The eFactory and ZDMP Marketplaces can be accessed from either platform and allow buying and selling of zApps	Working	10%	

Functional requirement	Description	Status	Progress	Comments
Marketplace Integration	as well as other 3rd party apps across the system			
T65C001 Exchange of device- and sub-device Information	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	Working	10%	
T65C002 Exchange of supported operations per device	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	Not started	0%	
T65C003 Exchange of historical data of devices	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	Not started	0%	
T65C004 Exchange of end user/owner information	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	Not started	0%	

WP7

T7.1, T7.2, T7.3, T7.4 - Prediction and Optimisation Designer

Functional requirement	Description	Status	Progress	Comments
PE001 - Create Model Project	Creates a new instance of an optimisation/prediction project	Working	30%	Will use the first component samples to create the generator
PE002 - Select Optimisation Objective	User selects the optimisation objective from an available set of options	Finished	100%	Taxonomy of problems and objectives ready
PE003 - Set Time Performance	User selects the time performance from an available set of options	Waiting	0%	Waiting to finalise taxonomy of solutions
PE004 - Set Accuracy	User selects the accuracy from an available set of options	Waiting	0%	Waiting to finalise taxonomy of solutions
PE005 - Get Algorithms	User selects the optimisation algorithm from an available set of options	Finished	100%	First version already available
PE006 - Set Algorithm	User confirms the selected configuration and creates the model	Finished	100%	First version already available
PE007 - Load Model	User loads the created model	Working	10%	On-going definition of detailed technical specifications
PE008 - Select training data source	User selects the data sources to train the model	Waiting	0%	Evaluation of Gitlab as backend for generator and training
PE009 - Train Model	User trains the model	Waiting	0%	Evaluation of Gitlab as backend for generator and training
PE010 - Select test data source	User selects the data sources to evaluate the model	Waiting	0%	Evaluation of Gitlab as backend for generator and training
PE011 - Test Model	User evaluates the model	Waiting	0%	Evaluation of Gitlab as backend for generator and training

Functional requirement	Description	Status	Progress	Comments
PE012 - Select production data sources	User selects the data sources to run the model	Waiting	0%	Evaluation of Gitlab as backend for generator and training
PE013 - Execute model	User confirms the execution of the model	Waiting	0%	Evaluation of Gitlab as backend for generator and training
PE014 - Visualise Output Data	User visualises results of executing the model	Waiting	0%	Evaluation of Gitlab as backend for generator and training

T7.1, T7.2, T7.3 Prediction and Optimisation Runtime

Functional requirement	Description	Status	Progress	Comments
T7123A001 - Select the source	Select the source of input to be whether Message bus or Storage.	Working	25%	Local storage and local database implemented. message bus (local and ZDMP) and storage (ZDMP) remaining
A7123A002 - Specify the input channels	Specifies the name of the sensors, parameters etc, for the source.	Working	40%	Database and file partly implemented. Message bus not started
A7123A003 - Perform optimisation	Calls the ‘optimiser’ with necessary input.	Working	70%	
A7123A004 - Perform prediction	Calls the ‘predictor’ with necessary input.	Working	60%	Needs to be combined with new compute unit concept
A7123A005 - Configure	Configures the run-time relevant parameters of the component.	Working	10%	Model dependent. So should be implemented for each Layer separately
A7123A006 - Check for updates	Checks the status of updates to the component on Marketplace.	Waiting	0%	Waiting until Marketplace component is ready to integrate

T7.4 - Process Assurance Runtime

Functional requirement	Description	Status	Progress	Comments
T74A001 Select data for quality prediction	The user selects which data he wants to use for quality prediction	Waiting	0%	
T74A002 Select data for quality analysis	The user selects which data wants to use for quality analysis	Waiting	0%	
T74A003 Select data for quality process optimisation	The user selects which data wants to use for process quality optimisation	Waiting	0%	
T74A004 Visualise data for quality prediction	The user selects the kind of graphs to be shown in the interface	Waiting	0%	
T74A005 Visualise data for quality analysis	The user selects the kind of graphs to be shown in the interface	Waiting	0%	
T74A006 Visualise data for process quality optimisation	The user selects the kind of graphs to be shown in the interface	Waiting	0%	
T74A007 Set Algorithm and its properties	The user selects the optimisation algorithm from an available set of options	Waiting	0%	
T74A008 Predict	The user requests a prediction on the quality of the process	Waiting	50%	Connects to a process optimisation runtime instance and send a request. needs to complete the request with more config data.
T74A009 Classify	The user requests a classification on the quality of the process	Waiting	0%	
T74A010 Optimise	The user requests an optimisation of the parameters of the process	Waiting	0%	
T74A011 Save model	The user saves the created model	Waiting	0%	
T74A012 Load model	The user loads the created model	Waiting	0%	
T74A013 Show quality prediction results	The user obtains the results of the prediction results	Working	90%	
T74A014 Show quality analysis results	The user obtains the results of the analysis results	Waiting	0%	
T74A015 Show process quality	The user obtains the results of the optimisation results	Waiting	0%	

Functional requirement	Description	Status	Progress	Comments
optimisation results				

WP8

T8.1, T7.3 - Digital Twin

Functional requirement	Description	Status	Progress	Comments
DT001 - Product configuration	User selects the parameters of the product to be executed or simulated. Objective: To configure the product model according to the provided configuration	Working	20%	
DT002 - Process configuration	User selects the parameters of the process to be executed or simulated. Objective: To configure the process model according to the provided configuration	Working	20%	
DT003 - Load Model	User loads the model. Objective: To load/update a new model instance with the provided configuration	Working	25%	
DT004 - Execute model	User confirms the execution of the model. Objective: To execute the digital representation of the model according to the provided configuration	Working	25%	
DT005 - Simulate model	User confirms the simulation of the model. Objective: To execute the simulation of the model according to the provided configuration	Working	20%	
DT006 - Visualize Output Data	User visualizes the model/simulation. Objective: To visualize the data inferred from the real and simulated model	Working	20%	

T8.2, T8.4 - Product Assurance Runtime

Functional requirement	Description	Status	Progress	Comments
T82A001 - Processing Data Instance Generation	Creates an instance for data processing. Objective: To generate the necessary data to feed the Product Quality and Supervision Trainers	Working	20%	Interfaces defined. Working with smoke component until other components involved are ready like Data Acquisition or Message Bus
T82A002 - Create Product Quality Model	Create an instance of product quality model trainer in run-time. Objective: To start the process of training the model as data arrives	Working	20%	

Functional requirement	Description	Status	Progress	Comments
T82A003 - Create Supervision Model	Create an instance of supervision model trainer in run-time. Objective: To start the process of training the model as data arrives	Working	80%	Testing phase
T82A004 - Model Container Deployment	To deploy instances of the data processor and a Product Quality Model or a Supervision model as docker containers using the Application Runtime. Objective: Models must be executed at runtime	Working	50%	Integrating with AI Analytics Runtime using manifesto
T82A005 - Establish Data Processing Link	To connect/route the data processor to the specific trainer and prediction modules. Objective: To send the required processed data to the trainer and prediction modules to be used	Working	20%	Interfaces prepared. Using smoke components waiting for other components like Data Acquisition and Message Bus
T82A006 - Store Model Instances	This component logs the different models created by the zApps and their status (running, stopped, etc). Objective: Store model instances previously created and made them available to other zApps	Working	50%	Integrating with AI Analytics Runtime which provides that service
T82B001 - Create Data Subscription	Receive product information from the Message Bus on a continuous basis. Objective: To send the data to the prediction and trainer modules	Working	20%	Interfaces prepared. Using smoke components
T82B002 - Authentication for data subscription	Product data could be confidential, requiring authenticating the data user. Objective: To ensure security and privacy of the product data	Waiting	0%	Waiting for a more stable release of secure components to integrate
T82B003 - Processing Rules Definition	Define a processing rule for a variable of the product data. Objective: Data could be received following a different format or temporal alignment than the expected by the models	Not started	0%	
T82B004 - Data Processing	Processing rules are applied to the received data. Objective: To transform data and aggregate them according to a specific temporal window	Working	15%	
T82B005 - Processed Data Storage	Processed data according to the defined rules is stored. Objective: Processed data must be available for generating an historical set or for performing further analysis	Working	0%	

Functional requirement	Description	Status	Progress	Comments
T82B006 - Processed Data Publishing	To send the processed data to the subscribed modules. Objective: Different submodules, mainly the trainer and predictor, used the same data	Working	0%	
T82D001 - Predictions Storage	This repository stores the predictions generated by the Quality Predictor. Objective: To retrieve historical information in a specific time range and compare the accuracy among	Working	0%	
T82D002 - Predictions query	The Predictions Repository retrieves the predictions requested in a specific period. Objective: To get information regarding the predictions generated	Working	0%	
T82F001 - Load Model	Load from Model Deployment Manager the model selected by the end user at configuration time of zApp. Objective: To perform model training once processed data are available	Working	50%	Integrating with AI Analytics Runtime
T82F002 - Load Training Parameters	Configure the training according to the specific configuration passed by the Model Deployment Manager. Objective: To perform a model training once processed data are available	Working	60%	
T82F003 - Load Training Dataset	Loads the processed training dataset passed through by the Data Processor. Objective: To perform model re-training once new processed data are available	Working	70%	Data is loaded through a smoke API. Waiting for integration with other components
T82F004 - Train Model with Dataset	Start training with a dataset supplied by Data Processor. Objective: To improve anomaly detection capabilities using an extended or more recent dataset	Working	90%	Training phase is practically completed. Testing and integration with AI Analytics Runtime is pending
T82F005 - Training KPI status	Training status (running, target reached/unreached) and other meaningful training KPIs are available to other modules. Objective: To let data analyst monitor the performance and accuracy of the training process	Working	40%	KPIs calculated but not shared between other components yet
T82F006 - Send Trained Model	Send freshly trained model to Quality Predictor. Objective: To let Quality Predictor update the model	Working	50%	Tested locally

Functional requirement	Description	Status	Progress	Comments
	under execution and save it in the storage			
T82F007 – Stop & Start Training	Stop/Start training the model. Objective: 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	Finished	100%	API to manage the training is provided
T82G001 - Execute trained anomaly detector	A previous trained anomaly detector model using the Supervision Model trainer is executed in runtime with the received product data. Objective: To get anomalies, which include variable abnormal values, error statistics, possible alerts, and variables responsible of such alerts.	Working	80%	In testing
T82G002 - Anomaly notification	Detected anomalies in specific variable must be made available to the rest of the components as soon as they are generated. Objective: zApps and ZDMP components will use the detected anomalies to react accordingly and present notifications to end users	Working	10%	Designing interfaces and made a proof of concept locally
T82G003 - Anomaly detector update	An anomaly detector model is could be retrained by the Supervision Model Trainer to improve accuracy. Objective: To guarantee the accuracy of the detected anomalies, the new model will replace the previous one	Working	70%	Retraining is available. Researching about online learning, to train the model continuously
T82G004 - Anomaly thresholds reporting	For each variable, an upper-bottom normal condition threshold is automatically calculated by the trainer. Objective: These calculated thresholds are updated in real-time and are useful to report what is considered and anomaly	Working	80%	
T82G005 - Anomaly contributing variables	When an anomaly is detected it should be explained which variables are generating such anomaly. Objective: To report which specific variables must be considered to solve the detected anomaly	Working	40%	Variables are retrieved, but not shared yet. A model to describe the variables consistently is being designed
T82G006 - Recent anomalies	A historical of set of the most recent anomalies. Objective: To receive information of the previous generated anomalies (for instance, if a component was not previously subscribed)	Working	90%	

Functional requirement	Description	Status	Progress	Comments
T82C001 - Load Model	Load from Model Deployment Manager the model selected by the end user. Objective: To perform model training once processed data are available	Not started	0%	Role of Model Deployment Manager still to be completed
T82C002 - Load Training Parameters	Configure the training according to the specific configuration passed by the Model Deployment Manager. Objective: To perform a model training once processed data are available	Not started	0%	Role of Model Deployment Manager still to be completed
T82C003 - Load Training Dataset	Loads the processed training dataset passed through by the Data Processor. Objective: To perform model re-training once new processed data are available	Not started	0%	interface with Data Processor still to be completed
T82C004 - Train Model with Dataset	Start training with a dataset supplied by Data Processor. Objective: To improve prediction capabilities using an extended or more recent dataset	Working	40%	dataset is to be considered available as a static csv file for the moment
T82C005 - Training KPI status	Training status (running, target reached/unreached) and other meaningful training KPIs are available to other modules. Objective: To let data analyst monitor the performance and accuracy of the training process	Working	10%	Schema to define training status availability still to be fully implemented
T82C006 - Send Trained Model	Send freshly trained model to Quality Predictor. Objective: To let Quality Predictor update the model under execution and save it in the storage	Working	40%	Proposed Schema of interaction with AI analytics runtime foresees the use of trained model for on-line prediction
T82C007 - Stop & Start Training	Stop/Start training the model. Objective: 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	Not started	0%	
T82E001 - Execute trained predictor	A previous trained model using the Product Quality Model trainer is executed in runtime with a subset of the product data. Objective: To get predictions regarding product	Working	40%	Proposed Schema of interaction with AI analytics runtime

Functional requirement	Description	Status	Progress	Comments
	quality as processed data is received form the data processor			foresees the use of trained model for on-line prediction
T82E002 - Data batch aggregation	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). Objective: To get predictions with small subsets of the data and reducing the required time to execute a model	Not started	0%	
T82E003 - Prediction publishing configuration	Predictions are published with a predefined time frequency (every 5 seconds, for instance). Objective: To provide predictions to and external component in a suitable frequency to visualize the data	Working	20%	Waiting for the scheduling API provided by the AI Analytics Runtime
T82E004 - Predictions broadcast	Predictions must be published and made available to rest of the components as soon as they are generated. Objective: zApps and components will use the predictions to gain insights regarding the quality of the manufacturing data	Working	20%	Proof of concept. Waiting for Message Bus integration
T82E005 - Model update	Use a new product quality model because of a retraining. Objective: To guarantee the accuracy of the generated predictions, models should be updated considering the most recent data	Not started	0%	
T82E007 - Predictions query	The Quality Predictor or external modules will query the predictions repository in an specific time-range for a specific quality variable. Objective: To get information regarding the predictions generated	deprecated	0%	Predictions should be stored when available, then predictions query for statistical and comparison reason should be available/implemented in other zAsset, most probably zApps
T82E008 - Optimization Launch	The module returns the set of process variables values that optimize the outcome of a target variable. Objective: To maximize or	Working	20%	

Functional requirement	Description	Status	Progress	Comments
	minimize the value of a selected target variable			
T82E009 - Optimization	The Quality Predictor executes several times the trained model with different parameters following an algorithm. Objective: To get the optimal set of values for each of the process variables involved	Working	30%	First tests performed

T8.3 - Non-Destructive Inspection

Functional requirement	Description	Status	Progress	Comments
T83A001 - Select analysis type	Select the processing algorithm to be used to process data for quality inspection. Objective: To select the needed processing algorithm	Not started	0%	
T83A002 - Select streaming data source	Specifies the input source needed for quality inspection, by subscribing a specific topic on the Message Bus. Objective: To provide the source to the processing algorithm	Working	40%	
T83A003 - Specify historical data source	Specifies the input data needed for quality inspection from the Storage. Objective: To provide the proper input to the processing algorithm	Not started	0%	
T83A004 - Select streaming data destination	Select the output/results destination by indicating a specific topic to be published on the Message Bus. Objective: To provide output broadcast	Working	40%	
T83A005 - Specify historical data destination	Specifies the output data destination on the Storage. Objective: To store output including results	Not started	0%	
T83A006 - Configure	Configures the run-time relevant parameters of the component. When dealing with AI based algorithms, this implies selecting the trained model to be used. Objective: needed for the component to function properly as desired	Working	20%	
T83A007 - Execute the analysis	Performs the quality analysis. Objective: To have the quality results as output of the component	Working	80%	
T83A002 - Send analysis results	Sends the quality analysis results. Objective: To allow prompt access to the quality results as output of the component	Working	40%	
T83B001 - Select AI Engine	End user selects specific AI processing engine (AI Image Classifier, AI Image Labeller …) to train a model associated to it. Objective: To obtain a list of associated models	Not started	0%	

Functional requirement	Description	Status	Progress	Comments
T83B002 - Select AI model	End user selects the AI model to be trained. Objective: To improve quality analysis performance, AI models need to be trained	Not started	0%	
T83B003 - Select training dataset	End user selects the training dataset from available historical data. Objective: To train selected AI model with selected dataset	Working	20%	
T83B004 - Select training parameters	End user selects training parameters. Objective: To configure training process	Not started	0%	
T83B005 - Train Model	Load selected model, dataset, configuration and start training. Objective: To improve inspection capabilities	Working	60%	
T83B006 - Training KPI status	Training status (running, target reached/unreached) and other meaningful training KPIs are available during training. Objective: To let data analyst monitor training performance	Working	20%	
T83B007 - StopTraining	Stop training the model. Objective: 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	Not started	0%	
T83B008 - Store Trained Model	Stores trained model to AI Model Storage. Objective: To let Quality Inspection Engine update the model to be executed	Working	10%	
T83F101 - 3DScan Create docker files	Create docker file and test container	Finished	100%	Subject to changes
T83F102 - 3DScan Backend skeleton	Set up Meshlabs in server mode and manage to run it from python	Finished	100%	
T83F103 - 3DScan Conversion engine	create Meshlab script for conversion from cloud of points to surface-only, run it and modified it when needed	Finished	100%	Not mandatory, just for clarifications
T83F104 - 3DScan Simplification engine	create parametrizable Meshlab script for simplification, run it and modified it when needed	Finished	100%	
T83F002 - 3DScan Reference object	Place 3D model in Euclidean space from certain reference	Waiting	0%	Waiting for more realistic data
T83F002 - 3DScan Backend skeleton	Set up Meshlabs in server mode and manage to run it from python	Finished	100%	Not mandatory, just for clarifications

ZERO DEFECTS

Manufacturing Platform

ZDMP

www.zdmp.eu