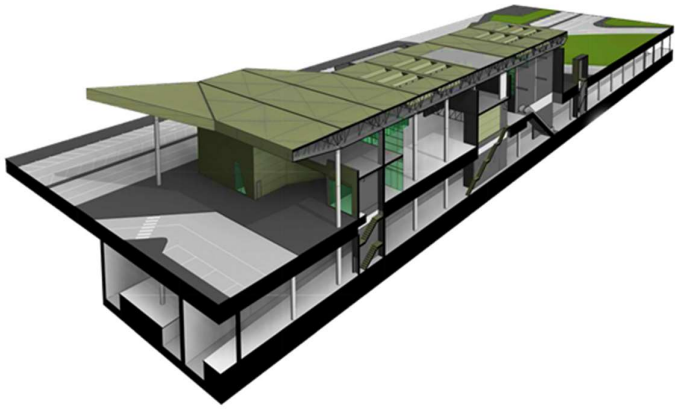


COGITO

CONSTRUCTION PHASE DIGITAL TWIN MODEL

cogito-project.eu

D2.5 – COGITO System Architecture v2

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958310

D2.5 – COGITO System Architecture v2

Dissemination Level:	Public
Deliverable Type:	Report
Lead Partner:	Hypertech
Contributing Partners:	UCL, AU, UEDIN, CERTH, UPM, BOC-AG, QUE, NT
Due date:	30-04-2022
Actual submission date:	30-04-2022

Authors

Name	Beneficiary	Email
Giannakis, Giorgos	Hypertech	g.giannakis@hypertech.gr
Papafragkakis Apostolos	Hypertech	a.papafragkakis@hypertech.gr
Vassiliadis, Michalis	Hypertech	m.vassiliadis@hypertech.gr
Kakarantzas Pavlos	Hypertech	p.kakarantzas@hypertech.gr
Malavazos, Christos	Hypertech	c.malavazos@hypertech.gr
Rovas, Dimitrios	UCL	d.rovas@ucl.ac.uk
Lilis, Georgios	UCL	g.lilis@ucl.ac.uk
Katsigarakis, Kyriakos	UCL	k.katsigarakis@ucl.ac.uk
Teizer, Jochen	AU	teizer@eng.au.dk
Esterle, Lukas	AU	lukas.esterle@eng.au.dk
Chronopoulos, Christos	AU	chrichr@cae.au.dk
Bosché, Frédéric	UEDIN	f.bosche@ed.ac.uk
Bueno Esposito, Martín	UEDIN	martin.bueno@ed.ac.uk
Robertson, Gail	UEDIN	gail.robertson@ed.ac.uk
Wilson, Amy	UEDIN	Amy.L.Wilson@ed.ac.uk
Kaheh, Zohreh	UEDIN	zkaheh@exseed.ed.ac.uk
Tsakiris, Thanos	CERTH	atsakir@iti.gr
Gounaridou, Apostolia	CERTH	agounaridou@iti.gr
Karkanis, Vasilios	CERTH	vkarkanis@iti.gr
Chatzakis, Michael	CERTH	mchatzak@iti.gr
García-Castro, Raúl	UPM	rgarcia@fi.upm.es
González-Gerpe, Salvador	UPM	salvador.gonzalez.gerpe@upm.es
Bernardos, Socorro	UPM	sbernardos@fi.upm.es
Woitsch, Robert	BOC-AG	robert.woitsch@boc-eu.com
Falcioni, Damiano	BOC-AG	damiano.falcioni@boc-eu.com
Andriopoulos, Panos	QUE	panos@que-tech.com
Zografou, Chara	QUE	c.zografou@que-tech.com
Moraitis, Panagiotis	QUE	p.moraitis@que-tech.com
Trantafillou, Yannis	QUE	y.triantafillou@que-tech.com
Belej, Bohuš	NT	belej@novitechgroup.sk
Straka, Martin	NT	straka@novitechgroup.sk
Baňas, Vladislav	NT	banas@novitechgroup.sk
Fedor, Jozef	NT	fedor@novitechgroup.sk
Lofaj, Stanislav	NT	lofaj@novitechgroup.sk

Reviewers

Name	Beneficiary	Email
Bernardos, Socorro	UPM	sbernardos@fi.upm.es
Moraitis, Panagiotis	QUE	p.moraitis@que-tech.com

Version History

Version	Editors	Date	Comment
0.1	All contributing partners	18.11.2021	Concept. Architecture – Final Revision
0.2	Hypertech	29.11.2021	Updates in Sections 1 and 2
0.3	Hypertech	04.02.2022	Updates in UCs sequence diagrams
0.4	All contributing partners	25.02.2022	Updated UCs sequence diagrams review
0.5	Hypertech	21.03.2022	Updates in Section 3
0.6	All contributing partners	30.03.2022	Updates in Req. and Interfaces tables
0.7	Hypertech	08.04.2022	Updates in Component Diagrams
0.8	All contributing partners	15.04.2022	Updates in Sections 4, 5 and 6
0.9	UPM, QUE	29.04.2022	Deliverable internal review
1.0	Hypertech	30.04.2022	Submission to the EC

Disclaimer

Executive Summary

This deliverable constitutes the outcome of the work carried out to design and deliver the final version of the COGITO ecosystem architecture. The present documentation of the COGITO system architecture is oriented towards (1) the definition of the Conceptual Architecture, (2) the development of the Use Cases Sequence Diagrams, (3) the elicitation of each component's Requirements and Specifications, and (4) the internal design of individual components.

The Conceptual Architecture constitutes a high-level illustration of the COGITO solution with the various components involved, as have been reviewed and refined in the deliverable "D2.1 – Stakeholder requirements for the COGITO system" [1]. It also provides a high-level description of the components with their main functionalities that complements the conceptual architecture design. UML sequence diagrams have been developed for each COGITO Use Cases (UCs) to convey the high-level information about the relationships among the different components. We gave special attention to identifying and defining the relationships of each component with the Digital Twin Platform – the core interoperability component. A high-level sketch of dependencies among different parts of the COGITO system (e.g., individual components interfaces, etc.) has also been delivered, including information about the functional and non-functional requirements, the constraints of the different elements in terms of software and hardware resources, compatibility with standards, etc. UML component diagrams have also been developed providing additional information about each COGITO component's sub-components. Those UML component diagrams are further detailed in the corresponding demonstrator type deliverables.

The first version of the COGITO system architecture was released in M9, and it has been documented in the deliverable "D2.4 – COGITO system architecture v1", providing a solid basis for the individual components' prototyping. With the COGITO components prototypes and the COGITO data models & ontologies development being progressing, the first version of the overall architecture has undergone updates, based on the feedback received from the technical developers of the COGITO components during the technical workshops that were organised within the M10-M18 period. For the sake of completeness, the outcomes of D2.4 are replicated, accommodating the latest updates to the Use Cases Sequence Diagrams, the Components requirements and specifications tables, and the internal design of individual components.

Table of contents

List of Figures

List of Tables

List of Acronyms

Term	Description
AI	Artificial Intelligence
API	Application Programming Interface
AR	Augmented Reality
BCSC	BlockChain Smart Contract
BIM	Building Information Model
BPMN	Business Process Model and Notation
COGITO	Construction Phase diGItal Twin mOdel
DB	DataBase
DCC	Digital Command Centre
DigiTAR	Digital Twin visualisation with AR
DoA	Description of Action
DT	Digital Twin
GDPR	General Data Protection Regulation
H&S	Health and Safety
HMD	Head Mounted Display
HSE	Health, Safety and Environment
HSEM	Health, Safety and Environment Manager
HSES	Health, Safety and Environment Supervisor
HSET	Health, Safety and Environment Trainer
IoT	Internet of Things
KPI	Key Performance Indicator
POPD	Protection Of Personal Data
PM	Project Manager
PMS	Process Modelling and Simulation
QC	Quality Control
QM	Quality Manager
QS	Quantity Surveyor
REST	Representational State Transfer
SDK	Software Development Kit
SLAs	Service Level Agreements
SLAM	Service Level Agreements Manager
SM	Site Manager
T	Task
UAV	Unmanned Aerial Vehicle
UC	Use Case
UDI	User-Driven Innovation
UI	User Interface
UML	Unified Modelling Language
UR	User Requirement

VR	Virtual Reality
WODM	Work Order Definition and Monitoring tool
WOEA	Work Order Execution Assistance tool
WP	Work package

1 Introduction

As per DoA, COGITO aims to materialise the digitalisation benefits for the construction industry by harmonising Digital Twins with the Building Information Model concept and to establish a digital Construction 4.0 toolbox. The COGITO toolbox consists of on-site data acquisition tools to collect data from site operations, work planning and management tools that optimise the schedule of work and the corresponding allocation of resources, health & safety tools to prevent hazards, and quality control tools to systematically detect geometric and visual defects. COGITO also deploys graphical user interfaces and apps to interact with the construction stakeholders and issue alerts to crews on site. Methods & technologies to ensure interoperability among the different components comprising the COGITO ecosystem are applied within the Digital Twin platform, the core of the entire ecosystem, lying in the middle of the majority of data exchanges among different components. The COGITO Digital Twin platform relies on harmonised interfaces, standardised data structures (such as IFC), ontologies, communication protocols and data formats (like REST, MQTT, JSON and RDF), delivering a re-usable and extensible construction digital twin.

The integration and functioning of such a complex ecosystem require a diligently designed and delivered system architecture, agreed among all the technical developers of individual COGITO components. To this direction, two iterations of the COGITO system architecture's documentation have been released in the context of "T2.4 – COGITO System Architecture Design". The first version of the COGITO system architecture was released in M9, and it has been documented in the deliverable "D2.4 – COGITO system architecture v1", providing a solid basis for the individual components' prototyping.

The COGITO partners followed a specific methodology to design the software architecture that will act as the backbone of all the subsequent developments foreseen in the project. The methodology has been based on agile principles with multiple iterations and continuous and interactive communication with the COGITO partners, as illustrated in Figure 1. The process consists of six (6) steps towards delivering the first version of the architecture, as defined below:

- **Step 1:** Revision of the DoA conceptual architecture updating the COGITO components to align with the most recent work detailed in the D2.1 "Stakeholder requirements for the COGITO system";
- **Step 2:** Definition of the conceptual data flow and high-level building blocks' dependencies;
- **Step 3:** Creation of UML sequence diagrams of all the Use Cases (UCs) defined in the D2.1;
- **Step 4:** Identification of components requirements in terms of software / hardware, programming language (s), etc. as well as functional & non-functional specifications and internal/external dependencies; a clear definition of interfaces required for their operation (input/output data, format, method, endpoint, and protocol) – see Annex A for the template created to support this activity;
- **Step 5**: Creation of UML component diagrams aligned with the components' requirements and interfaces table (step 4) and the UML sequence diagrams (step 3);
- **Step 6**: Finalisation of the conceptual architecture and drafting the current document, including the most recent version of all UML diagrams and the respective components' tables.

With the COGITO components prototypes and the COGITO data models & ontologies development being progressing according to the plan, the first version of the overall architecture has undergone small updates, based on the feedback received from the technical developers of the COGITO components during the technical workshops that were organised within the M10-M18 period. To elicit the latest updates that are reported in this document the aforementioned Steps 2, 3, 4, 5 and 6 were repeated having as the basis the findings of D2.4.

Online collaborative tools involving all partners were extensively used to ensure alignment, consistency, and shared understanding among the consortium members. Key meetings where critical decisions were taken regarding the definition of the architecture were:

- The **Work Package (WP) targeted online meetings**, organised with the participation of all involved partners and aimed at defining components requirements aligned with the WPs / Tasks

description. Inter-dependencies, as regards the work and time plan were also identified during these telcos;

- Four (4) **technical workshops**, organised with the participation of all partners and aimed at discussing, fine-tuning, and agreeing on the UML sequence diagrams of all the COGITO UCs.

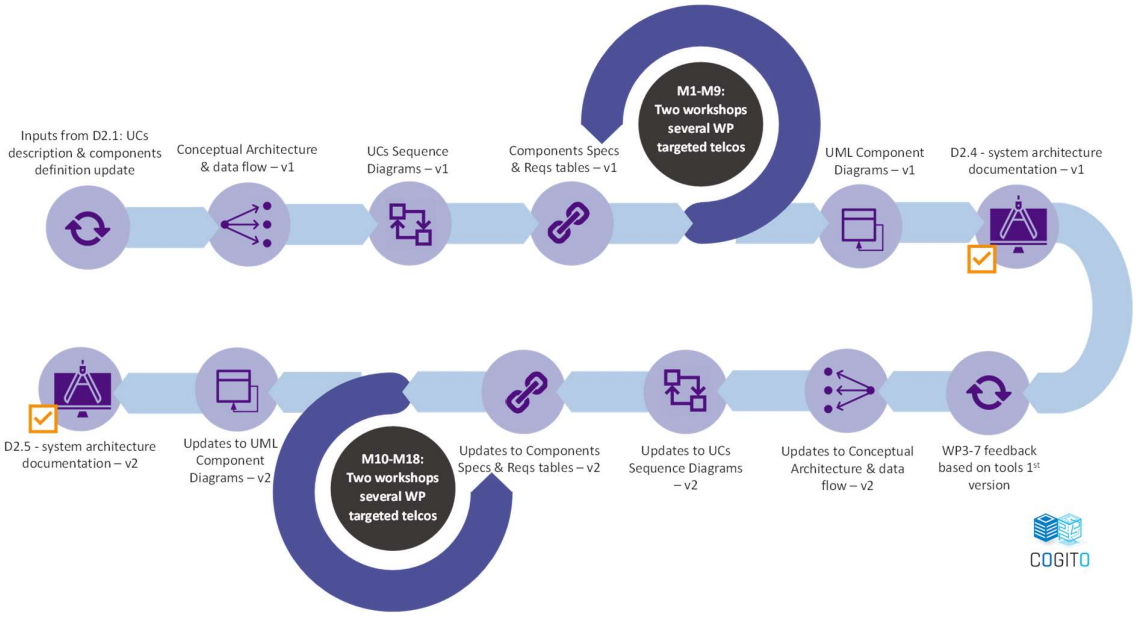


Figure 1 – Methodology for the definition COGITO system architecture - versions 1 and 2

1.1 Scope and Objectives of the Deliverable

Based on the COGITO DoA, the scope of the current document is to deliver the overall architecture of the COGITO solution and the specifications of the key components along with their functionalities. The objective is to provide an appropriate decomposition of the COGITO solution without detailing the interface specifications, to be elaborated in "*WP3 – COGITO Data Model and Reality Capture Data Tools*". To this end, critical architectural elements were identified, including components and their relationships, as well as relevant architectural mechanisms to address cross-cutting relationships (i.e., those not localized within a single component but affecting the design and operation of other parts of the architecture). One of the core outcomes of this work is the updated conceptual architecture diagram, presented in this document, that identifies the system components and interconnections (data flows) amongst them.

Aspects pertaining to deployment and the implementation technologies/platforms of the various COGITO tools are also investigated as part of this deliverable. This is to ensure operational capacity in the various environments where the tools will be deployed – from cloud-based ICT tools to Augmented Reality (AR) glasses and construction site Internet of Things (IoT) components. Finally, the mechanisms defined as part of COGITO architecture design that will guarantee data protection and privacy throughout the COGITO system are discussed in alignment with the "*D10.2 – POPD - Requirement No. 2*" [2] and the "*D1.2 – Data Management Plan*" [3].

1.2 Relation to other Tasks and Deliverables

This document is the second tangible outcome of the "*T2.4 – COGITO System Architecture Design*", which – based on the DoA – falls under the activities of "*WP2 – Stakeholders Requirements, Evaluation Planning and Architecture Design*". The work performed strongly depends on various tasks and deliverables within the WP2 and beyond as depicted in Figure 2. More specifically:

- Architecture design is primarily based on the work performed in "*T2.1 – Elicitation of Stakeholder Requirements*" and its primary outcome i.e., "*D2.1 – Analysis of digital tools market and prevailing*

regulatory frameworks". More specifically, the COGITO components and their involvement in the various UCs were essential for developing this version of COGITO system architecture.

- Another important input was the work performed within "*T3.1 – Survey of Existing Data Models & Ontologies & Associated Standardization Efforts*" and its deliverable "*D3.1 – Survey of Existing Models & Ontologies & Associated Standardization Efforts*" [4] including a comprehensive review of the relevant data models, ontologies, and standardisation initiatives.
- "*D2.3 – COGITO evaluation methodology*" (output of "*T2.3 – Development of an Evaluation Methodology for the Impact of COGITO Tools*") provided the methodology for the evaluation of the COGITO solution and tools, from both functional and usability viewpoints supporting the consortium to ensure alignment with the overall solution design;
- From a market viewpoint, "*T2.2 – Analysis of Regulations & Markets for Digitalization in Construction Industry*" provided the analysis of the regulatory and market conditions within which the COGITO digital twin tools will be called upon to make an impact. This supported consortium members to guide architectural decisions, taking into account business opportunities as well.

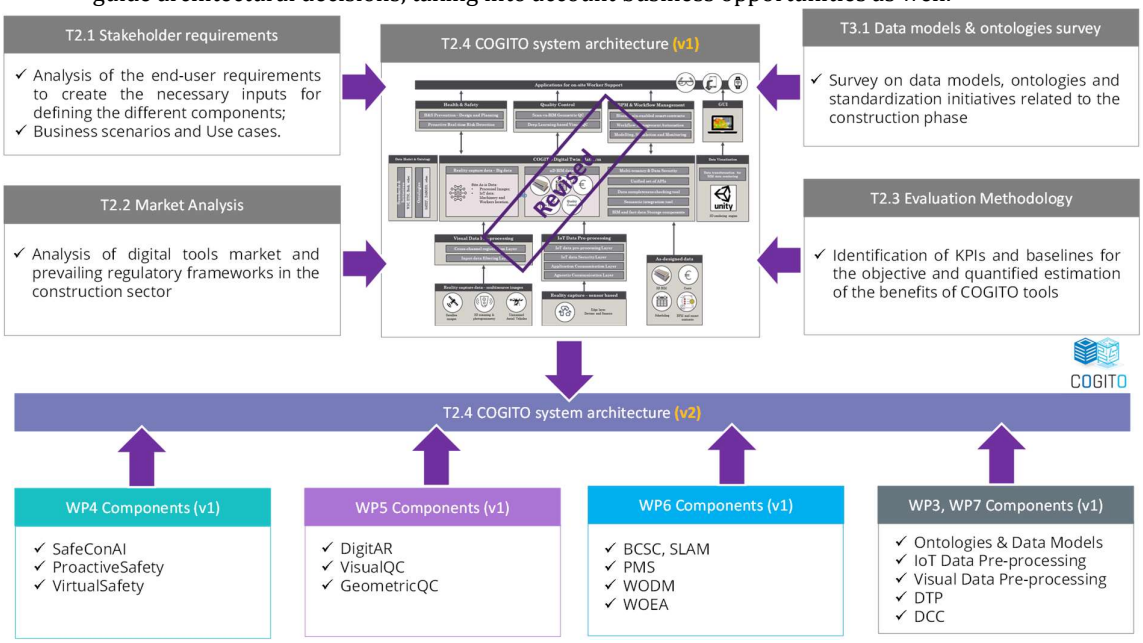


Figure 2 – Dependencies on other tasks of the COGITO work plan

Apart from the abovementioned tasks/deliverables, for the design and delivery of the final version of the COGITO system architecture, valuable inputs have been received by the following WPs:

- "*WP3 – COGITO Data Model and Reality Capture Data Tools*";
- "*WP4 – On-site Workers' Health & Safety Assurance Tools";*
- "*WP5 – Geometric and Visual Quality Control Tools*";
- "*WP6 – Adaptive Workflow Modelling and Workflow Management Automation Tools";* and
- "*WP7 – COGITO Digital Twin Platform*".

This document will act as the backbone of all integration activities foreseen in the project. Thus, D2.5 will guide the activities to be performed in "*T8.1 – End-to-end ICT System Integration, Testing and Refinement*".

1.3 Structure of the Deliverable

The deliverable is structured as follows:

- Section 1 introduces the document;

- Section 2 describes the COGITO conceptual architecture, the updates made as regards the relevant diagram provided in the COGITO DoA and the core building blocks of the solution;
- Section 3 provides the UML sequence diagrams of all COGITO UCs along with a brief description of the interactions that are taking place in each of them;
- Section 4 provides the requirements and specifications of all the components of COGITO architecture following the template in Annex A;
- Section 5 provides the UML component diagrams of all the components of COGITO architecture accompanied by a description of their main sub-elements. In the same section the main technical data protection mechanisms already identified are presented; and
- Finally, Section 6 concludes the document.

1.4 Updates to the first version of the COGITO System Architecture

The first version of the COGITO System Architecture was presented in the deliverable D2.4. Based on the outcomes of several, targeted bilateral meetings among technical partners and workshops that took place from M10 to M18, the first version has undergone slight modifications that are listed below:

- Section 1 – Refinements in the dependencies of this deliverable to other COGITO tasks and deliverables that mainly concern the contribution of the technical WPs first outcomes to the evolution of the final COGITO system architecture;
- Section 2 – Rephrasing of the IoT Data Pre-processing and the Visual Data Pre-processing components to be more consistent with their actual development, considering that since the D2.4 submission, both components have released their second version; changes in the conceptual architecture and information flow to be aligned with the updates to specific Use Cases Sequence Diagrams;
- Section 3 – Updates to the UC-1.1, UC-1.2, UC-2.1, UC-2.2, and UC-4.2 Sequence Diagrams to accommodate the outcomes of the latest discussions towards the respective use cases realisation;
- Sections 4 and 5 – Updates to the Requirements & Specification tables and the UML component diagrams of the following components: IoT Data Pre-processing; Visual Data Pre-processing, DT Platform, GeometricQC, VisualQC, WODM, PMS, and DigiTAR.

2 COGITO Conceptual Architecture

This section provides information about the role of each high-level key component, as appeared in the initial conceptual architecture diagram of the project, introduced in the Description of Action (DoA) [5]. The COGITO system architecture comprises:

- the COGITO Digital Twin (DT) Platform, a data integration middleware that supports the complex requirements of each of the applications;
- the Multi-source Visual and IoT Data Pre-processing components to pre-process raw visual and IoT data respectively;
- the Health & Safety (H&S) components to generate rules for hazard detection based on design & planning, and issue preventive warnings or alerts to construction workers and operating suggestions for equipment operators to mitigate dangerous situations leveraging real-time information from the site;
- the Quality Control components to retrieve as-designed and as-is data from the DT Platform and detect defects and areas out of tolerance using advanced algorithms;
- the Workflow modelling, simulation and management components to monitor and optimise the construction processes in terms of cost and time; and
- applications (apps) for AEC stakeholders, that retrieve metrics and messages populated by the health & safety, Quality Control and workflow management services from the DT Platform to support off-site and on-site crew activities and training.

Common ground of all above is the COGITO data models and ontologies, which organise data elements and standardise their relations, to the extent possible.

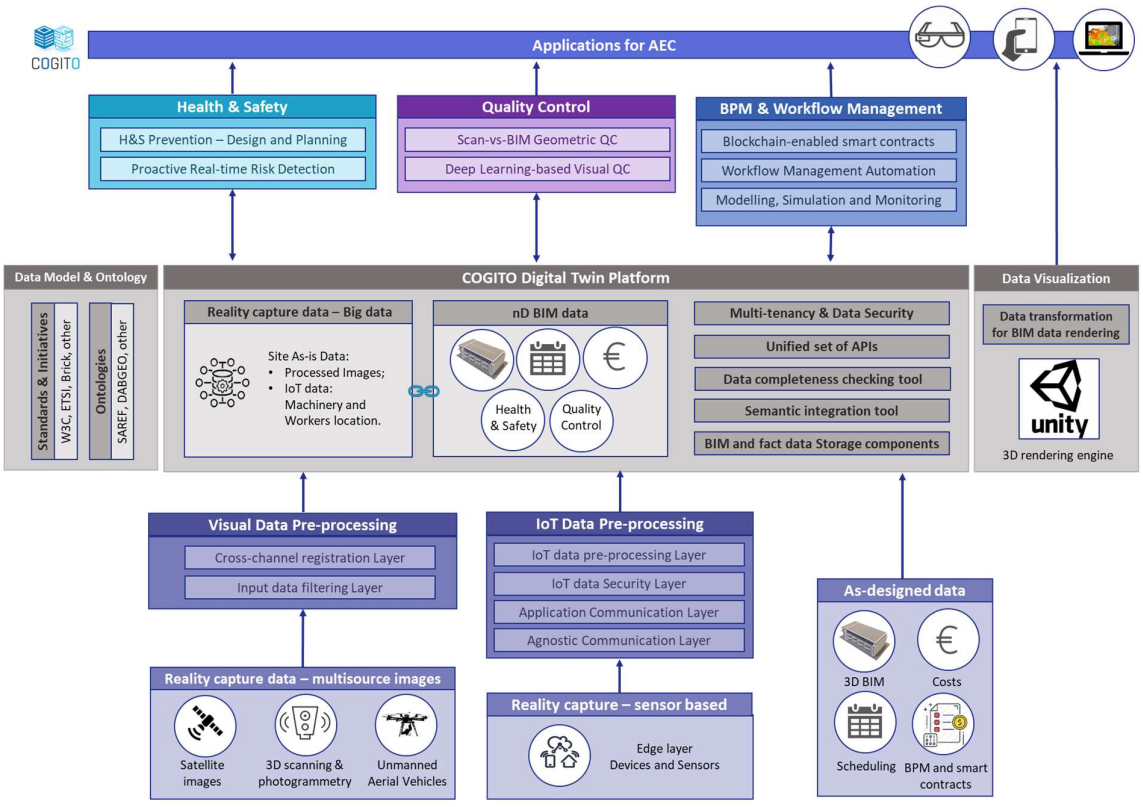


Figure 3 – COGITO system architecture [5]

Before illustrating the final conceptual architecture and the information flow among the individual components of the COGITO ecosystem, these component groups, that were reviewed and refined in the context of "*T2.1 – Elicitation of the Stakeholders requirements*" activities [1], are briefly introduced below.

2.1 Multi-Source Data Pre-Processing

The Visual and IoT Data Pre-processing components is responsible for filtering and processing raw data acquired on-site and serving data requested for real-time resources tracking (personnel and heavy machinery) and objects detection and recognition.

The **Visual Data Pre-processing** component, designed, developed, and delivered in "*T3.5 – Visual Data Pre-processing Module*", manages the visual data acquired onsite. It is in charge of receiving and handling the point clouds and the 2D images sent by laser scanners and static cameras, mobile phones, Augmented Reality (AR) goggles respectively. It provides a Graphical User Interface (GUI) to allow its user to interact with the application, add fields and new information, as well as link the IFC components with a capturing device and specific visual data.

To enable the capture and transmission of construction site as-is data, the suitability of commercially available sensors and relevant communication protocols are elaborately investigated in "*T3.3 - IoT solution in Construction Phase*". The resulting survey drives the selection of the IoT sensor network/system to serve as a back-end, feeding the **IoT Data Pre-processing** component with real-time data; the latter is developed in the context of "*T3.4 - IoT Data Pre-Processing Module*". The IoT Data Pre-processing component acts as a middleware, allowing ingestion and fusion of data from multiple IoT sources installed at the construction field; pre-processing operations shall be available depending on the specific site requirements, e.g., filtering, smoothing, aggregation. Ultimately, consistent, error-free timeseries data to be stored in the COGITO DT Platform are generated. A seamless communication with the DT Platform is achieved using RESTful Application Programming Interfaces (APIs) as well as real-time streaming protocols (e.g., Kafka).

2.2 Digital Twin Platform

The Digital Twin platform (**DT Platform**), to be designed and specified in "*T7.1 – Digital Twin Platform Design & Interface Specification*" and developed within "*T7.2 – Extraction, Transformation & Loading Tools and Model Checking*", is a semantically-enabled data integration middleware. The core functionalities to be provided by the DT platform include Master Data Management services, which include operational support for storing, versioning, routing and consistent updating of the data that comprise the virtual representation of the construction site.

The DT Platform supports the COGITO services and applications by providing a central repository for all types of data available before construction (e.g., 3D BIM models, schedules, and resources), during construction (point clouds, images, sensor tracking data and videos), and derived quantities like performance data Visual and Geometric Quality Control results, or even Health and Safety Rules.

The proposed central repository is supported by an ontology framework capable to capture all data requirements for the digital representation of the COGITO environment, to integrate the data provided by the different COGITO components and to respond efficiently to various data needs of the COGITO services and applications. The COGITO ontology framework is also aligned with well-known standards such as SAREF[1], which is supported by the European Telecommunications Standards Institute, and W3C Thing Description[2].

2.3 Adaptive Process Modelling and Workflow Management

For the Adaptive Process Modelling and Workflow Management, five components are deployed.

1. The Process Modelling and Simulation (**PMS**) tool, developed in "*T6.2 – Adaptive Processes/Workflow Modelling and Simulation-based Optimisation*", is used in the planning phase (before the actual construction starts) to develop process and workflow models of all interactions between the various tasks, building components and resources that a construction project entails. In the construction phase, the PMS provides functionalities for (a) simulation and formal verification of the process of the designed construction project to allow the project managers to

[1] https://saref.etsi.org/
[2] https://www.w3.org/2019/wot/td

identify process steps or interactions that are critical for the successful implementation of the project, (b) optimisation opportunities to minimise time and/or cost.

2. The Work Order Definition and Monitoring (**WODM**) tool, developed in "*T6.3 – Adaptive Workflow Management and Automation*", is used to define work order templates, generate work orders and executing/monitoring the defined workflow. Adaptiveness is necessary to account for unexpected effects leading to updates of the work orders and assignments to personnel, such as: weather patterns that may prohibit specific works, equipment availability shortness, etc. The WODM User Interface (UI) is designed and delivered in the context of "*T6.4 – Personalised On-site Works Support and Relevant Apps Development*".
3. The Work Order Execution Assistance (**WOEA**) is an application for on-site workers, that provides functionalities to assist them in reporting work progress and alert them for hazardous components and areas. This component is another outcome of T6.4 activities.
4. Blockchain-enabled smart contracts interacts with the WODM tool to provide trusted means to verify the completion of construction tasks. To this direction two main components are designed, developed and delivered within "T6.1 – Blockchain & Smart Contracts on the Workflow Modelling and Management", the Service Level Agreements Manager (**SLAM**) and the BlockChain-enabled Smart Contracts (**BCSC**) platform.

2.4 Quality Control

For the as-built Quality Control management, within COGITO, two main components are being developed: the Scan-vs-BIM–based Geometric Quality Control (**GeometricQC**) and the Deep Learning-based Visual Quality Control (**VisualQC**) components.

GeometricQC, to be designed, developed, and delivered within "*T5.1 – Scan-vs-BIM Geometric Quality Control*" and "*T5.3 – BIM-based Standard Test Methods for Geometric Quality Control*", aims to automatically control the geometric quality of the executed works against the specified geometric dimensions and tolerances, given the as-built 3D data acquired on-site. The as-built 3D data consist of (dense laser scanned) point clouds acquired on-site. The specified dimensions are obtained from the as-planned 4D BIM data, whereas the specified tolerances are obtained from ISO/CEN standards used by industry (and translated digitally to enable the automated process). The QC results are modelled and semantically linked to the BIM/DT model.

The VisualQC tool automatically detects common visible defects of constructed/erected concrete components and their severity in colour images (visual spectrum). The QC results are modelled and semantically linked to the BIM/DT data models. The VisualQC results from "*T5.2 – Deep Learning Image Processing for Visual Quality Control*".

2.5 Health and Safety

Health and safety measures can be taken during the design and planning phase to prevent accidents on the construction site. In this direction, within COGITO, the **SafeConAI** component is being developed in "*T4.1 – Health & Safety Prevention through Design and Planning*". SafeConAI aims to process the as planned 4D BIM data of a construction project and based on a health and safety rule checking prototype, propose mitigation measures, enhancing the 4D BIM data with relevant information.

ProActiveSafety is the solution to accidents prevention in the construction phase. This component, main outcome of "*T4.2 – Proactive Real-time Risk Monitoring and Detection*" and "*T4.3 – Tools for Personalized Alerts, Prediction and Feedback*" utilises behavioural data of resources (equipment and personnel) on the construction site to avoid close-calls, accidents, and collateral damage. The location data, acquired by the IoT Data Pre-processing component is utilised to analyse and predict trajectories of resources with a focus on four primary areas that reflect detection, avoidance, tracking, prediction, and learning. Estimated paths are cross-checked with potential hazards based on previous experiences/observations, rules, and the probability of hazards given the dynamic nature of the work environment. Functionalities for performing timely data processing using cloud-based artificial intelligence and issuing preventive warnings or alerts to construction workers are supported by this component.

2.6 On-site and Off-Site Data Visualisation

The Digital Command Centre (**DCC**) is the off-site data visualisation solution, that helps the Project Manager to monitor through visualisation the construction progress, detected QC defects and H&S issues. It is used to visualise, navigate and walkthrough the 3D BIM model, focusing on the geometric data, construction resources data and other data and annotations generated by the QC, H&S and Workflow tools (available through the DT platform). The DCC is the main output of "*T7.3 – Data Transformation for 3D BIM Rendering*" and "*T7.4 – 3D Mesh Data Quality and Consistency Checker and 3D Data Transformation Testing*".

The Digital Twin visualisation with Augmented Reality (**DigiTAR**) tool, a software package for commercial AR head mounted displays, prototyped, developed and delivered in "*T5.4 – User Interface for Construction Quality Control*", provides on-site visualisation of useful information such as geometric and visual quality control results (defects) as well as safety hazards using AR/mobile device.

Apart from the on-site and off-site data visualisation solutions mentioned above and intended to be used in the construction phase, **VirtualSafety** is another application planned for Health and Safety educational and training purposes. It is developed within "*T4.4 – Personalised Safety Education and Learning*".

2.7 Revised Conceptual Architecture and Information Flow

Having concluded a list of fifteen COGITO main components, introduced above, the conceptual COGITO system architecture of Figure 3 has been reviewed and refined to illustrate the high-level interactions and information flow among these components. To reflect the information flow on the system architecture and improve its readability, we decided to split the overall solution into two diagrams. The first diagram depicts the revised conceptual architecture that involves all COGITO components that are used in the planning phase, before the actual construction works start (see Figure 4). In contrast, the second diagram represents the information flow between the COGITO components that are applied in the construction phase (see Figure 5). In both Figures, information about the partner leading each component development is provided.

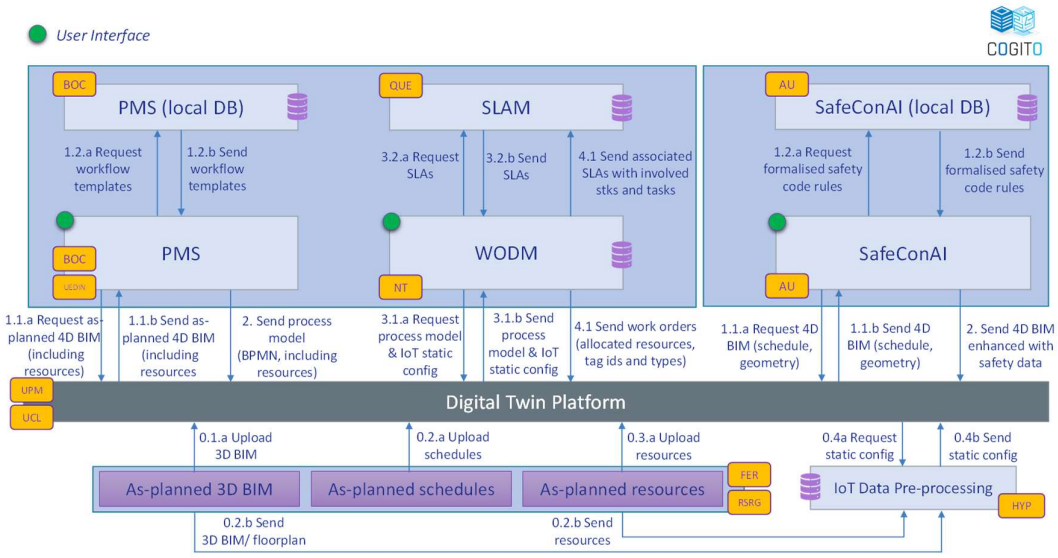


Figure 4 – Revised Conceptual Architecture: Before Construction Starts

As illustrated in Figure 4, before the construction works of a project start, the as-planned data of the project are uploaded to the DT platform. As-planned data includes the 3D BIM model, scheduling data that could be the fourth dimension of the BIM model, and the resources (workers and their roles and heavy machinery). Then, the information flows as follows:

1. The PMS tool requests and receives a subset of the 4D BIM data (as-planned schedule including construction elements IDs, including human and equipment resources allocated for each task) from the DT Platform. Using process and workflow templates, that have been designed and store in its

local Database (DB), it generates an optimised business process and workflow model that is sent to the DT Platform. In parallel, SafeConAI requests and receives (a) the 4D BIM data from the DT Platform, and (b) formalised safety code rules from its local DB, to enhance the 4D BIM data with health and safety information. The enhanced 4D BIM model is forwarded to the DT Platform.

2. The WODM tool requests and receives (a) optimised business process (provided by PMS) and the IoT static configuration data from the DT Platform (provided by the IoT Data Pre-processing component) and (b) the SLAs templates from SLAM, and after the user input to define the work orders and their assignments to personnel, it sends the associated SLAs along with involved personnel and tasks back to SLAM, while work orders linked with resources and IoT static configuration data are sent to the DT Platform.

In the construction phase (see Figure 5), groups of components are collaborating to provide support for the Worflow, QC and H&S management.

1. The IoT and Visual Data Pre-processing component feed the DT Platform with as-built data acquired on-site.
2. With regards to the worfklow management, the WODM tool sends notifications to workers about tasks that have been assigned to them through the WOEA application. This application is also used by the workers to report their tasks progress back to WODM. The PMS tool requests and receives resources tracking data from the DT platform, combines it with the reported tasks progress data, runs scenarios simulation and optimisation, resulting to updates of the process and workflow model. The updated model is received by WODM, which in turn sends that tasks progress and the updated work orders (workflow) to the DT Platform.
3. Concerning the prevention of the construction on-site accidents, ProactiveSafety receives the resources tracking data from the DT platform and processes it in combination with the 4D BIM model enhanced with health and safety information, to generate alarms for H&S risks that are sent as notifications to WOEA. Further information is populated by ProactiveSafety to update the H&S parameters and rules of the SafeConAI's local DB.
4. For the Quality Control checking, the GeometricQC tool requests and receives the 4D BIM model and point cloud data from the DT platform to produce the geometric QC results that are forwarded to the DT Platform. In a similar manner, the VisualQC tool requests and receives the 4D BIM model and visual (imagery) data from the DT platform to produce the visual QC results that are forwarded to the DT Platform.
5. DCC, DigiTAR and VirtualSafety are mainly used for visualisation purposes, as described in Section 2.6, however DigiTAR also serves as a mean of acquiring images on site.

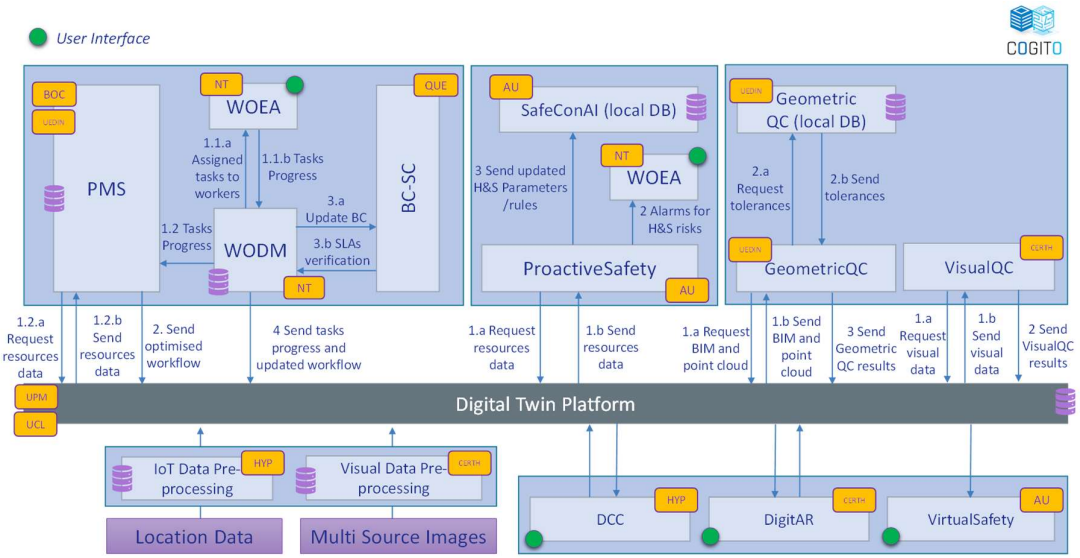


Figure 5 – Revised Conceptual Architecture: Construction Phase

As mentioned above, each of the steps entails significant complexity that a high-level conceptual architecture cannot capture. In the following sections, delving deeper into the sequence diagrams of the UCs, the components requirements & specifications, and diagrams (including all sub-components), these steps become more evident.

3 COGITO Use Cases – Sequence Diagrams

In this section, high-level information about the relationships among the different components, required for the realisation of the COGITO UCs, is provided in the form of UML sequence diagrams. Although a detailed description of all the UCs can be found in the D2.1, a concise summary is provided in each of the following sections for clarity and consistency purposes.

3.1 UC-1.1 – Efficient and Detailed project workflow planning using the project's construction schedule and as-planned BIM model

In this UC, a detailed construction project workflow is derived from the as-planned 4D BIM model using a data-driven approach. In particular, as shown in Figure 6, the UC is initiated by its stakeholder (actor).

After the login is processed (credentials provided centrally by the DT platform), the simulation part of the UC to extract the BMPN file is followed. In order to guarantee the execution of this part, the following steps/communications between components should take place:

1. The actor (PM/SM) logs in the PMS UI (the credentials are provided centrally by the DT Platform) and selects the project that will be further processed;
2. PMS requests and receives from the DT platform the as-planned 4D BIM data of the project, including as-planned resources (geometric representation of elements is not required);
3. Pre-defined workflow templates are loaded and visualised in the PMS UI;
4. The actor (PM/SM) defines the workflows that are required for the specific project and associates them with the already received as-planned data;
5. The PMS start the processing of productivity / climate / etc. historical data (already available in a local database) to be used as input to the simulations;
6. The actor (PM/SM) triggers the simulation initiation;
7. The PMS runs the simulations and visualise the simulated workflows for the project;
8. The actor (PM/SM) selects the simulated workflows that s/he prefers;
9. The selected workflow in store in PMS local database in BPMN format;
10. The PMS sends the generated BPMN (process model including the as-planned resources) to the DT Platform to be received by WODM for further processing.

Having the simulation part finalised, the workflow process for defining the relevant work orders can be initiated. The workflow is secured using blockchain technology. For this to be executed, the following steps/communications between components should take place (Figure 6):

1. The actor (PM/SM) logs in the WODM UI (the credentials are provided centrally by the DT Platform) and selects the project that will be further processed;
2. Upon project selection, WODM requests and receives the generated BPMN (process model) of the project from the DT platform along with the IoT static configuration data (tag ids and types);
3. WODM also requests and receives relevant SLAs and KPIs from the SLAM;
4. The process model along with the IoT static configuration data, the SLAs and KPIs are visualised in the WODM UI;
5. The actor (PM/SM) defines the work orders and associates them with SLAs and KPIs; resources to be tracked are also associated with IoT static configuration data by the actor;
6. WODM sends the work orders associated with SLAs and KPIs to SLAM;
7. SLAM sends the work orders with associated SLAs and KPIs to BCSC;
8. BCSC creates a decentralised network for enabling blockchain realisation;
9. WODM sends the work orders, including allocated resources and their connection to specific tags ids and types (IoT devices that are used for specific resources location tracking) to the DT Platform where they are centrally stored.

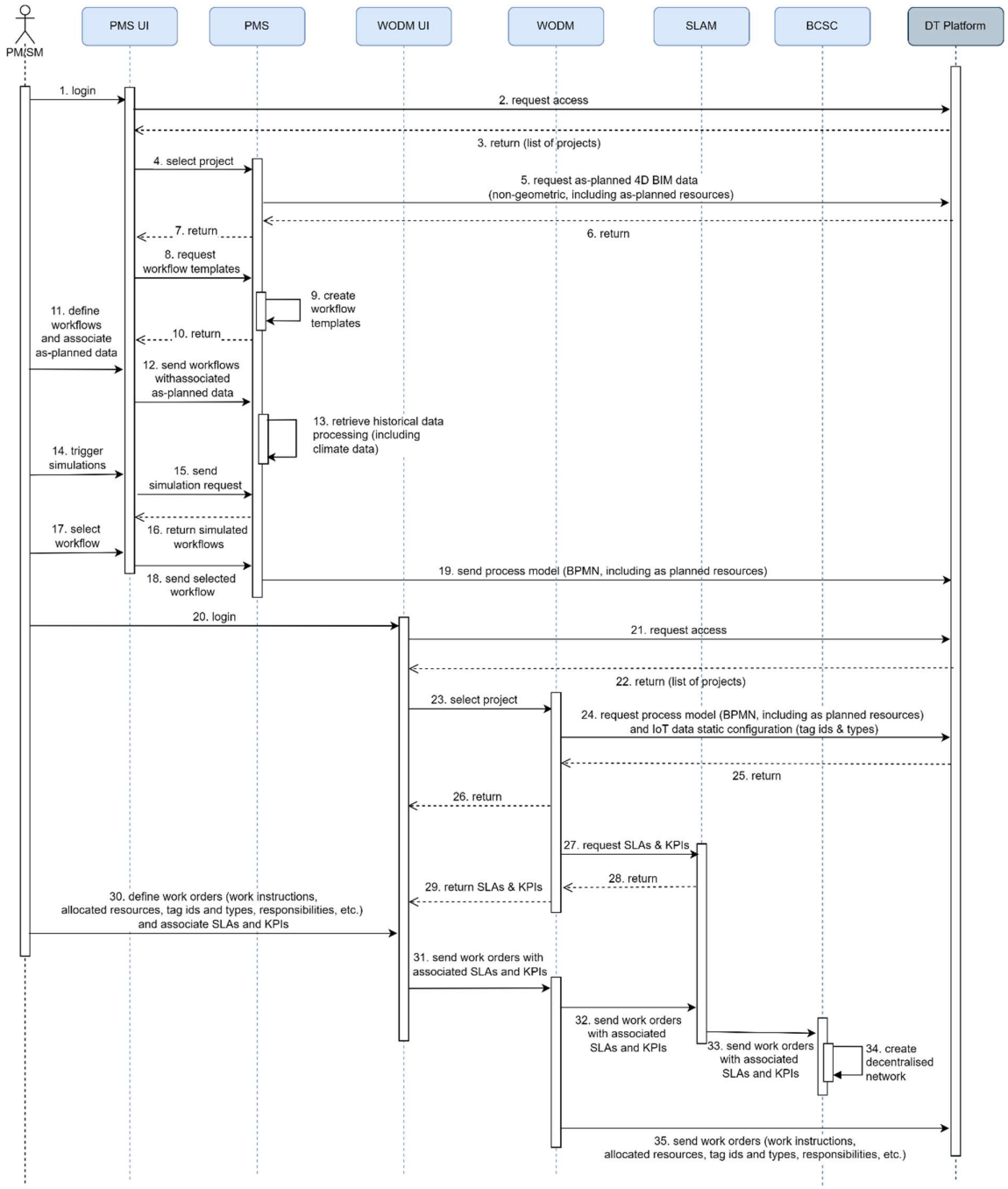


Figure 6 – Sequence diagram of UC-1.1

3.2 UC-1.2 – Systematic and secure execution, monitoring and updating of the project workflow

In this UC, it is assessed whether the project is executed according to the planned workflow (see UC-1.1) and is continuously monitored, keeping the workflow updated. As shown in Figure 7, for this to be executed, the following steps / communications between components should take place:

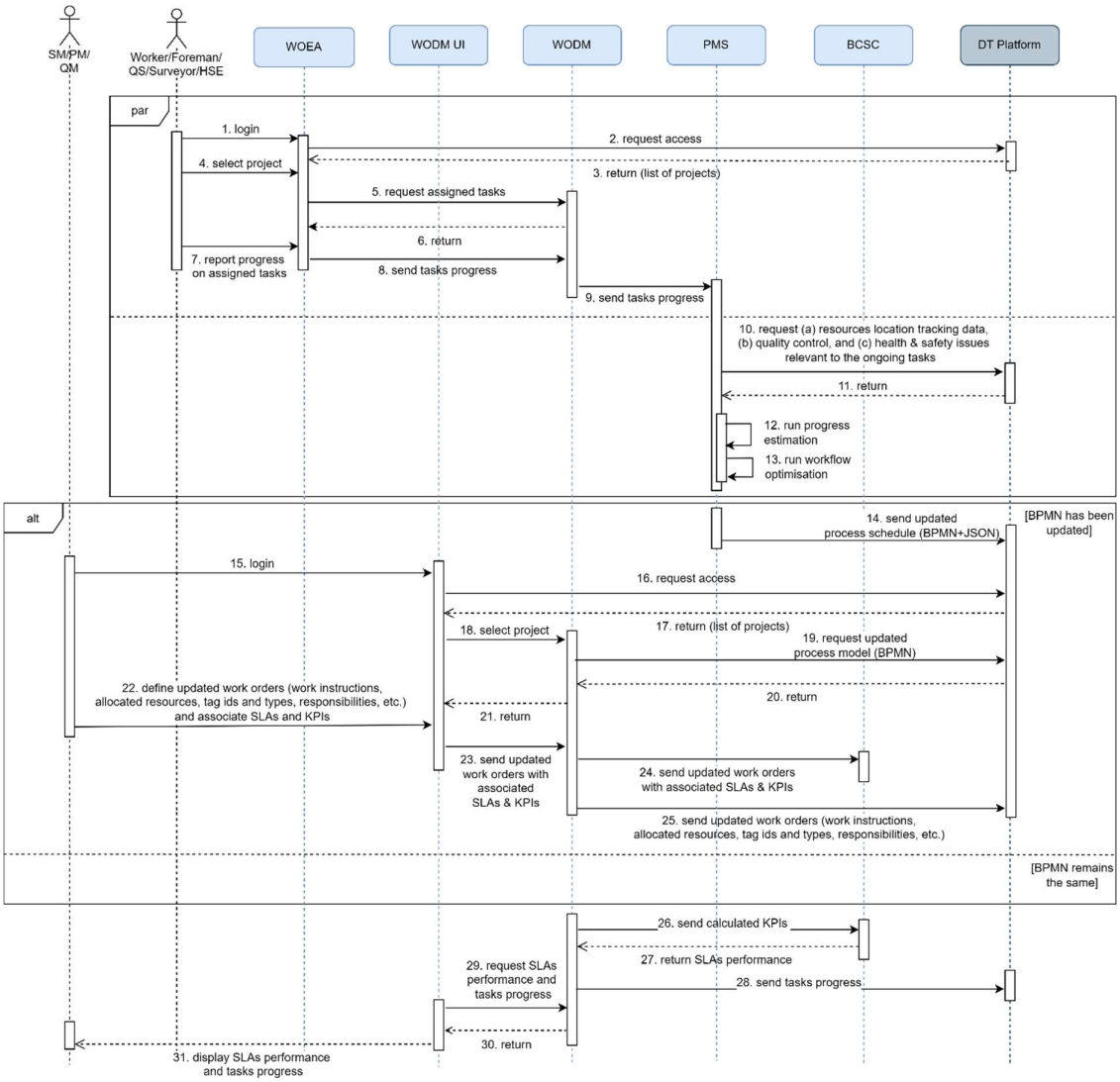


Figure 7 – Sequence diagram of UC-1.2

1. The actor (Worker/Foreman/QS/Surveyor/HSE) logs in WOEA (the credentials are provided by the DT platform) and selects the relevant project;
2. The actor (Worker/Foreman/QS/Surveyor/HSE) requests through WOEA the assigned tasks that s/he wants to be displayed;
3. WOEA requests and receives the assigned tasks from WODM and displays them;
4. The actor (Worker/Foreman/QS/Surveyor/HSE) reports through WOEA the progress made on the assigned tasks;
5. WOEA sends the tasks' progress to WODM;
6. WODM sends the tasks' progress to PMS;
7. PMS requests and receives from the DT platform the (a) location tracking data of the involved resources, (b) quality control and (c) health and safety issues relevant to the ongoing tasks;
8. PMS estimates the progress already made and runs an optimisation for optimal workflow management;
9. If there are updates to the existing process model, PMS extracts the updated process model and sends it to the DT Platform;
10. The actor (PM/SM) log ins WODM and select the relevant project (credentials are provided by the DT Platform) to receive the updated process model from the DT Platform;

11. The actor (PM/SM) defines the updated work orders and associates them with SLAs and KPIs; new resources to be tracked are also associated with IoT static configuration data by the actor;
12. WODM sends the updated work orders, along with allocated resources, associated SLAs and KPIs to the SLAM;
13. WODM sends the updated work orders, including the newly allocated resources and their connection to specific tags ids and types (IoT devices that are used for specific resources location tracking) to the DT Platform;
14. WODM updates the relevant progress-related KPIs and sends them to BCSC;
15. BCSC assesses the performance of the relevant SLAs and sends this information back to WODM;
16. The SLAs performance, along with the tasks progress, is displayed in the WODM UI.

3.3 UC-2.1 – Automated geometric tolerance compliance checking in 3D point cloud data and allocation to DT building component

In this UC, accurate geometric data are acquired, and the geometric tolerance specifications are checked automatically by matching that data to the BIM model and apply standard tolerance control methods. As shown in Figure 8, for this to be executed, the following steps/communications between components should take place:

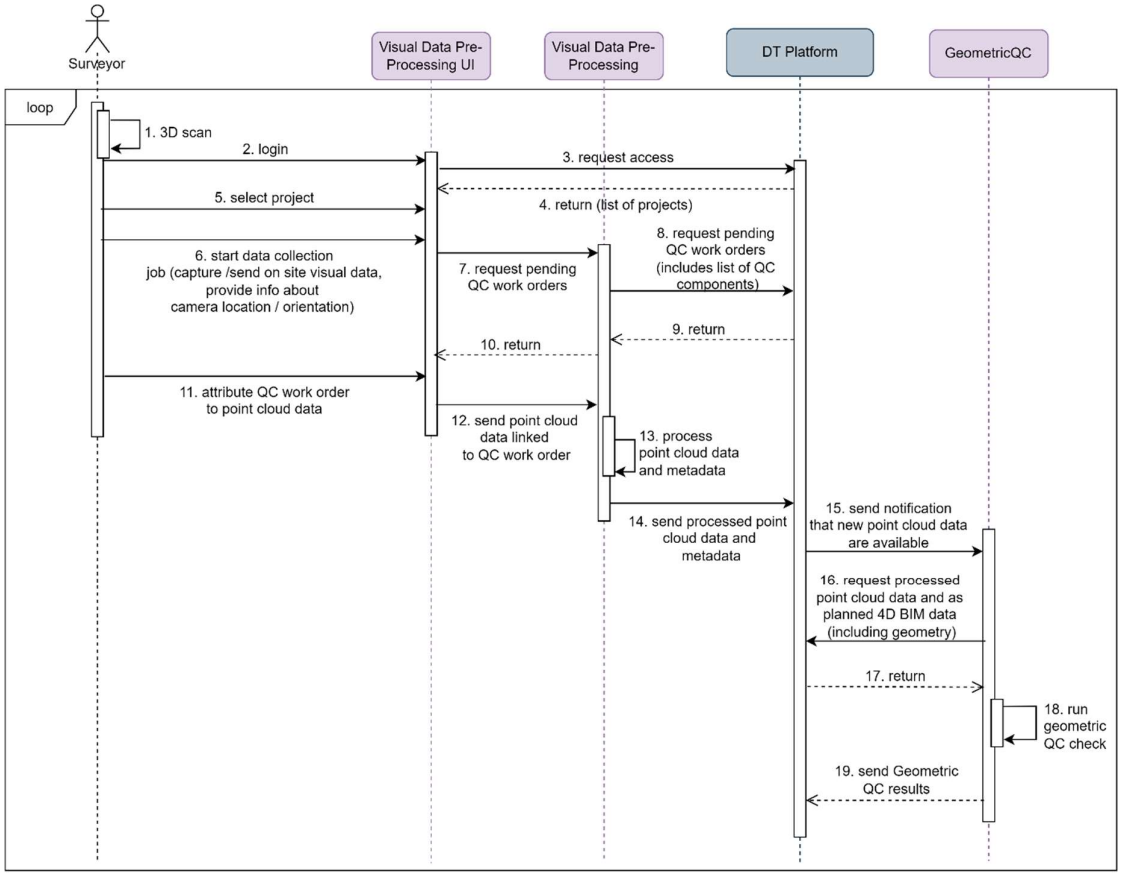


Figure 8 – Sequence diagram of UC-2.1

1. The actor (Surveyor) performs a 3D scanning of the site of interest to acquire a 3D point cloud;
2. The actor (Surveyor) logs in the Visual Data Pre-processing UI (the credentials are provided centrally by the DT platform);
3. The actor (Surveyor) starts the visual data collection job (incl. capture/send on-site visual data, provide information about camera orientation/location);

4. The Visual Data Pre-processing requests and receives from the DT platform pending QC work orders along with the list of QC components attached to each QC work order;
5. The actor (Surveyor) attributes components/tasks to the 3D point cloud data;
6. The Visual Data Pre-processing processes the 3D point cloud data linked with components/tasks;
7. The processed 3D point cloud data along with its metadata are sent and centrally stored in the DT platform;
8. The processed 3D point cloud data along with its metadata are sent to the GeometricQC;
9. The GeometricQC performs the geometric QC check;
10. The QC results and relevant metadata are sent to and centrally stored in the DT platform.

3.4 UC-2.2 – (Semi-)Automated detection of construction defects from visual input captured using AR and drones

In this UC, visual data from the site are captured and regions of risk in infrastructure (i.e., concrete defects, cracks, material displacements) are detected while their severity is estimated.

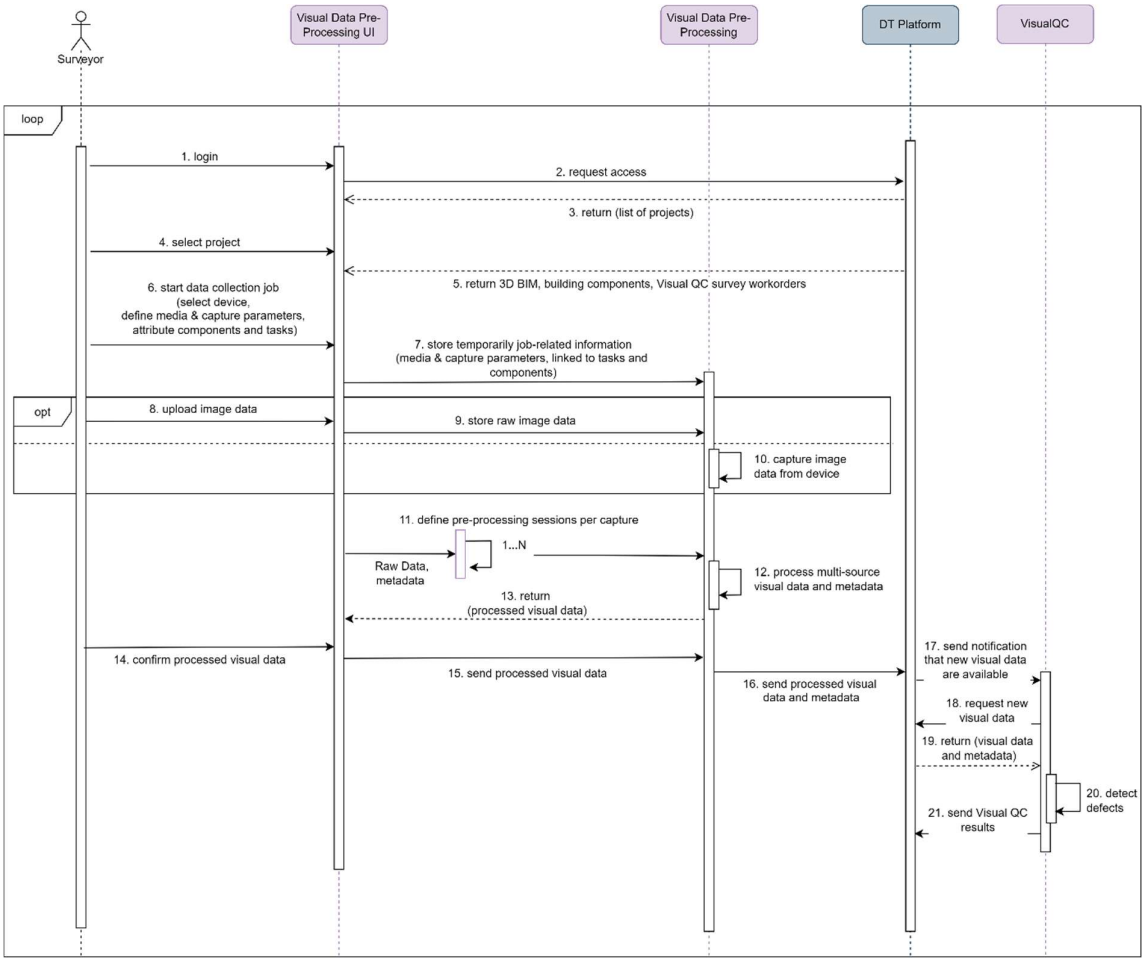


Figure 9 – Sequence diagram of UC-2.2

As shown in Figure 9, for this to be executed, the following steps / communications between components should take place:

1. The actor (Surveyor) logs in the Visual Data Pre-processing UI (the credentials are provided centrally by the DT platform) and selects the relevant project;
2. Upon the project selection, the Visual Data Pre-processing receives the 3D BIM model, the building components, and the Visual QC survey workorders;

3. The actor (Surveyor) starts a visual data collection job; the initiation of a collection job includes the device selection, the definition of the media/capture parameters, and the attribution of components and tasks; job-related information is temporarily stored in the local database of the Visual Data Pre-processing component;
4. The actor either uploads image data captured by any device to the Visual Data Pre-processing using its UI or captures image data using the Visual Date Pre-process operation mode of DigiTAR;
5. Based on a predefined number of pre-processing sessions per capture, the Visual Data Pre-processing transform image data to processed visual data;
6. The processed visual data are displayed in the Visual Data Pre-processing UI for the actor confirmation;
7. The confirmed, processed visual data are sent to the Visual Data Pre-processing backend to be forwarded, along with relevant metadata, to the DT Platform;
8. The DT Platform sends a notification to the VisualQC that new visual data are available;
9. VisualQC requests and receives from the DT platform the new visual data along with its metadata;
10. VisualQC processes the new visual data and detects defects;
11. VisualQC sends the Visual QC results (detected defects) to the DT Platform to be centrally stored.

3.5 UC-3.1 – BIM-based safety planning and hazard prevention before construction starts

In this UC, the regions of the construction site where specific hazards exist are identified and mitigation measures are proposed and linked to the BIM model. As shown in Figure 10, for this to be executed, the following steps/communications between components should take place:

1. The actor (HSE Manager/HSE Supervisor) logs in the SafeConAI UI (the credentials are provided centrally by the DT platform) and selects the relevant project;
2. Upon the project selection, SafeConAI requests and receives the as-planned 4D BIM data;
3. The actor (HSE Manager/HSE Supervisor) selects or inserts the safety code rules to be applied using the SafeConAI UI;
4. SafeConAI analyses the 4D BIM as opposed to the selected safety code rules and produces a 4D BIM model enhanced with safety information;
5. The 4D BIM model enhanced with safety information is sent and stored centrally in the DT platform;
6. 4D BIM model enhanced with safety information is visualised in the SafeConAI UI.

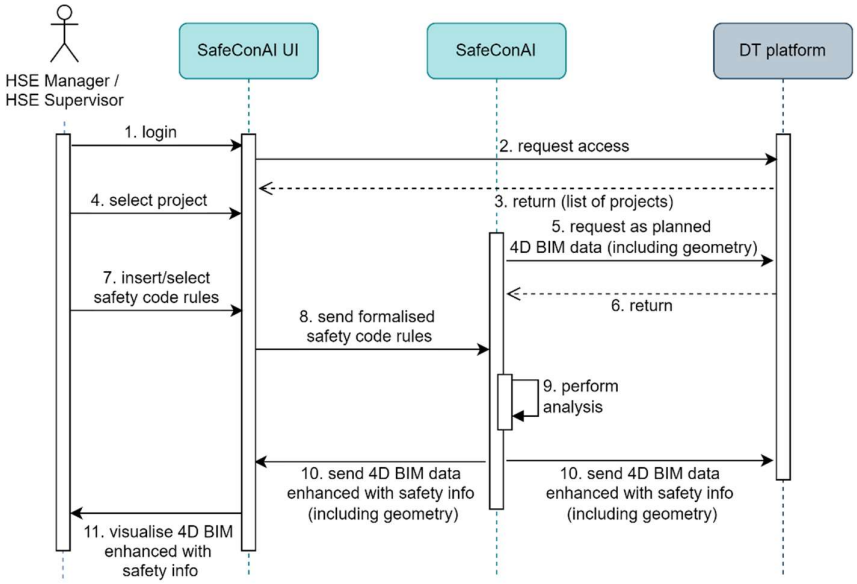


Figure 10 – Sequence diagram of UC-3.1

3.6 UC-3.2 – Monitoring, reporting, and proactive alarming of safety risks on outdoor construction sites

In this UC, location data of resources (equipment and personnel) on the construction site are monitored to avoid collision close-calls and accidents, and collateral damage. As shown in Figure 11, for this to be executed, the following steps/communications between components should take place:

1. ProactiveSafety request and receives from the DT platform the as-built 4D BIM model enhanced with safety information;
2. DT Platform is continuously feeding ProactiveSafety with the real-time IoT location tracking data of the involved resources;
3. ProactiveSafety analyses location tracking data against the 4D BIM model enhanced with safety information and estimates potential hazards;
4. ProactiveSafety sends hazards' alarms to WOEA;
5. WOEA notifies the concerned actor (worker) about these alarms;
6. ProactiveSafety runs statistical models and updates the safety parameters which are then sent to the SafeConAI.

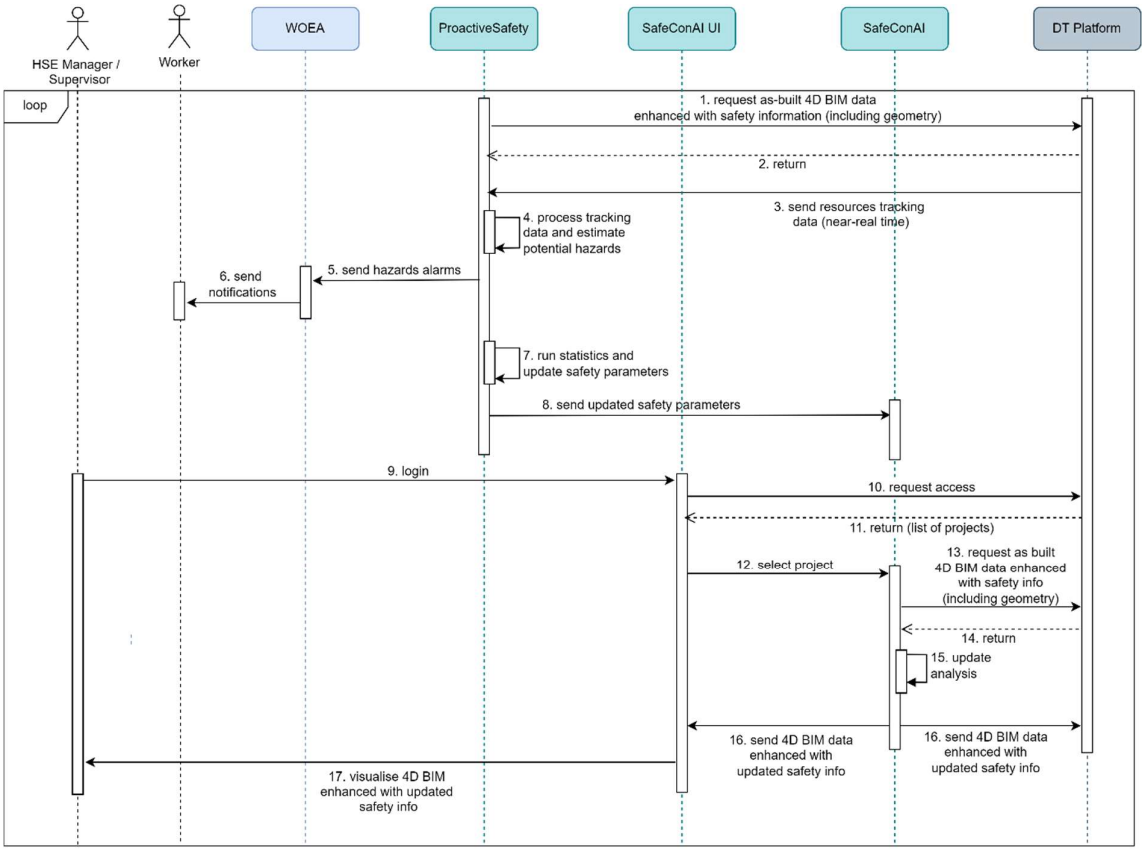


Figure 11 – Sequence diagram of UC-3.2

Once the safety parameters are updated, part of the sequence of UC3.1 is executed again towards updating the 4D BIM model enhanced with safety information. In particular:

1. The actor (HSE Manager/HSE Supervisor) logs in the SafeConAI UI (the credentials are provided centrally by the DT Platform) and selects the relevant project;
2. SafeConAI requests and receives the as-built 4D BIM data, enhanced with safety information, from the DT Platform;

3. SafeConAI analyses the as-built 4D BIM data and the updated safety parameters and produces an updated 4D BIM model enhanced with safety information;
4. The updated 4D BIM model enhanced with safety information is sent and stored centrally in the DT Platform;
5. The updated 4D BIM model enhanced with safety information is visualised in the SafeConAI UI.

3.7 UC-3.3 – Safety-augmented Digital Twin is used for construction safety training

In this UC, personalised construction safety education and training is provided. As shown in Figure 12, for this to be executed, the following steps / communications between components should take place:

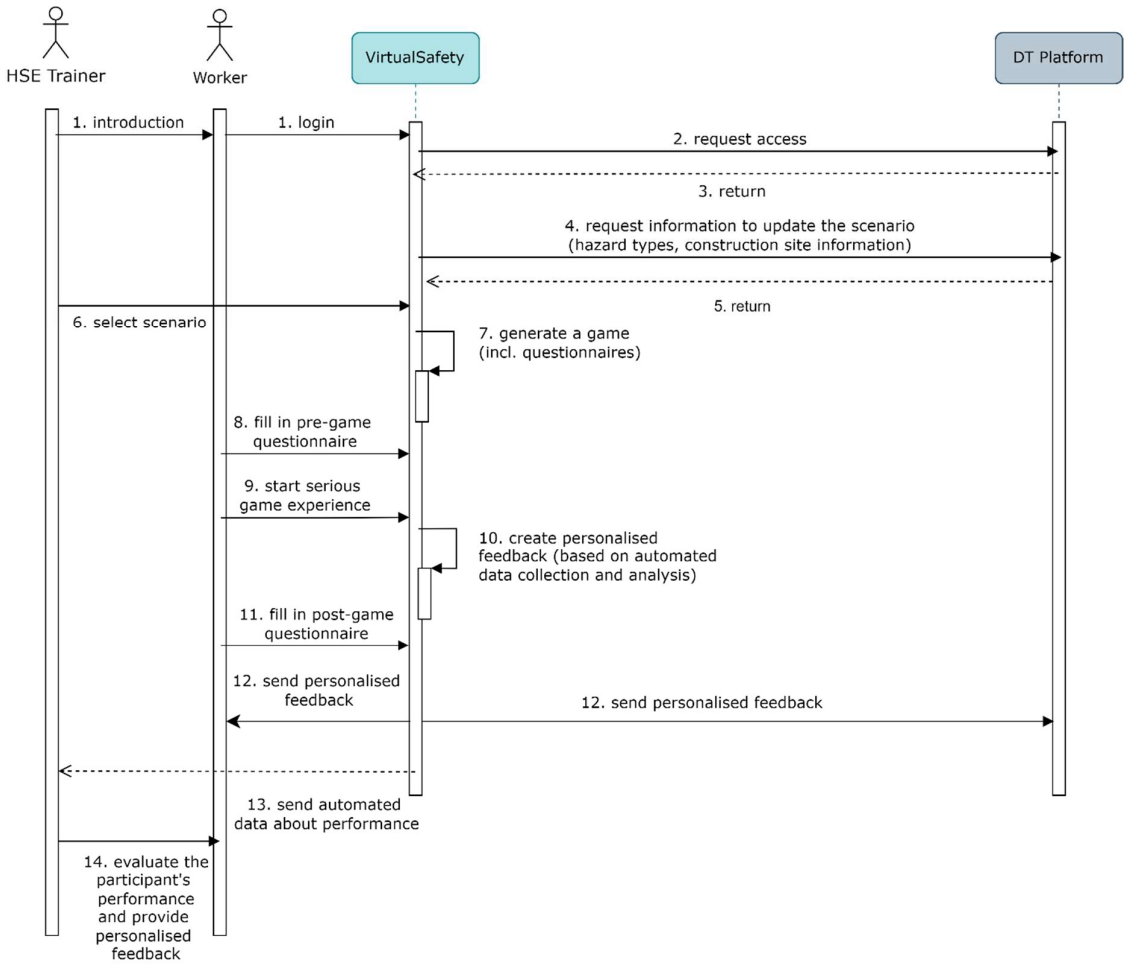


Figure 12 – Sequence diagram of UC-3.3

1. The actor (HSE Trainer) introduces the training platform to another actor (worker);
2. The actor (Worker) logs in the VirtualSafety (the credentials are provided centrally by the DT platform);
3. VirtualSafety requests and receives from the DT platform information about the construction site and the relevant hazard types;
4. The actor (HSE Trainer) selects a training scenario;
5. VirtualSafety generates a game including questionnaires;
6. The actor (Worker) fills in a pre-game questionnaire and starts the training experience;
7. VirtualSafety collects and analyses the data from the training experience and creates personalised feedback for this specific actor;
8. The actor (Worker) fills in a post-game questionnaire;

9. VirtualSafety collects and analyses the data from the post-game questionnaire and creates personalised feedback that is visualised to the actor (worker);
10. The overall training feedback is sent and centrally stored in the DT platform;
11. Performance data from the training are visualised to the actor (HSE trainer).

3.8 UC-4.1 – Remote visualisation of DT model information (Data Acquisition, Workflow, Safety, Quality) using the Digital Command Centre

In this UC, the 3D BIM model, IoT data and annotations generated by the QC, HSE and Workflow tools are rendered and visualised in different layers to the concerned actors.

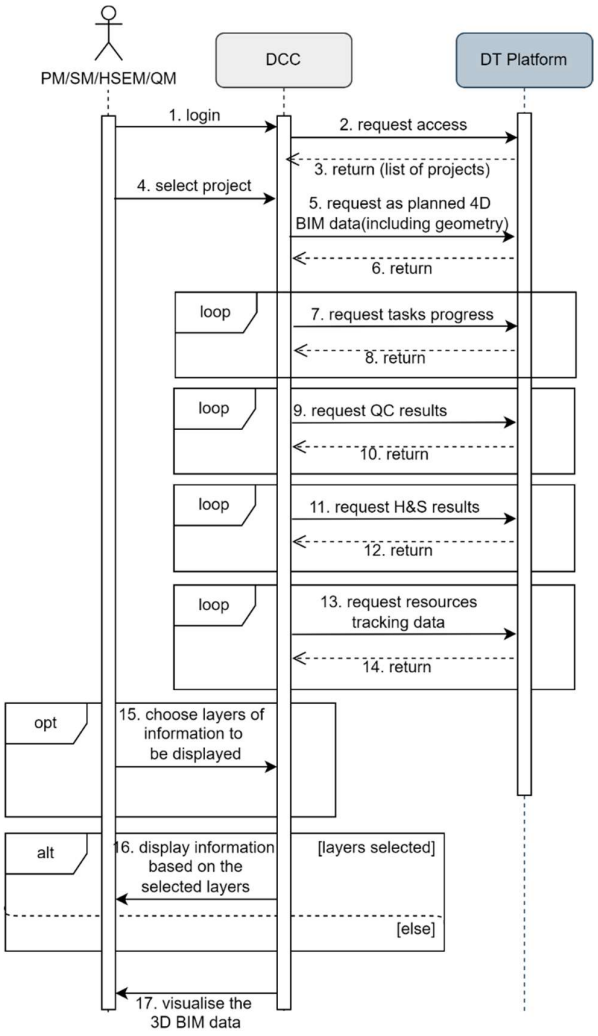


Figure 13 – Sequence diagram of UC-4.1

As shown in Figure 13, for this to be executed, the following steps/communications between components should take place:

1. The actor (PM/SM/HSEM/QM) logs in the DCC (the credentials are provided centrally by the DT platform);
2. DT Platform offers a list of available projects;
3. The actor (PM/SM/HSEM/QM) selects the project of interest;
4. DCC requests and receives from the DT Platform: the 3D BIM data; the workflow progress; the QC results; the H&S results; resources' tracking data;

5. DCC visualises the 3D BIM;
6. Once the actor (PM/SM/HSEM/QM) selects the layers of information to be displayed, the relevant information is also displayed based on this selection.

3.9 UC-4.2 – On-site visualisation of QC and Safety Planning information using AR/mobile device

In this UC, during the construction phase, the QC/safety information is displayed in an AR/mobile device to help effectively define the required remedy activities/mitigation measures.

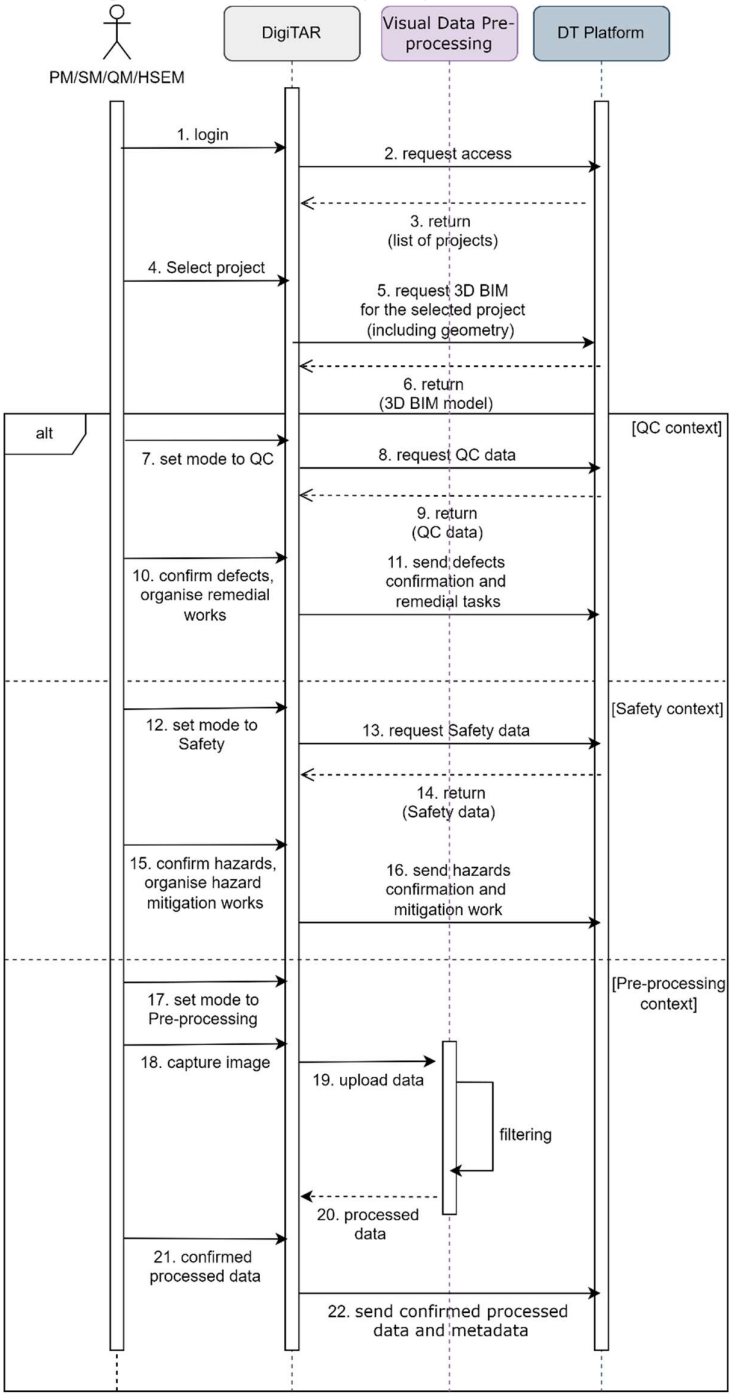


Figure 14 – Sequence diagram of UC-4.2

As shown in Figure 14, for this to be executed, the following steps / communications between components should take place:

1. The actor (PM/SM/HSEM/QM) logs in the DigiTAR (the credentials are provided centrally by the DT platform);
2. DT platform provides a list of available projects;
3. The actor (PM/SM/HSEM/QM) selects the project of interest;
4. Upon project selection, DigiTAR requests and receives the 3D BIM model of the project;
5. The actor (PM/SM/HSEM/QM) selects the operation mode of DigiTAR, where three options are offered: QC Context, Safety Context, and Pre-processing context.

If "QC context" is selected:

4a. DigiTAR requests and receives from the DT platform the 3D BIM model and the QC data for the selected project;
5a. The actor (PM/SM/QM) confirms the defects identified and organise remedial works in the DigiTAR;
6a. The DigiTAR sends the defects' confirmation and the remedial works to the DT platform to be centrally stored.

If "Safety context" is selected:

4b. DigiTAR requests and receives from the DT platform the 3D BIM model and the safety data for the selected project;
5b. The actor (PM/SM/HSEM) confirms the hazards identified and organise hazard mitigation works in the DigiTAR;
6b. The DigiTAR sends the hazards' confirmation, and the mitigation works to the DT platform to be centrally stored.

The "Pre-processing context" is a new feature of DigiTAR not initially planned. This new feature allows DigiTAR (running on HoloLens) to operate as an image data capturing device that feeds the Visual Data Pre-processing tool with pre-processed data. If "Pre-processing context" is selected:

4c. The actor (PM/SM/HSEM/QM) captures imaged using DigiTAR;
5c. The captured images are automatically uploaded to Visual Data Pre-Processing tool to be pre-processed;
6c. The pre-processed data are visualised in DigiTAR to let the actor confirm their validity;
7c. Upon confirmation, the confirmed pre-processed data are associated with relevant metadata in DigiTAR and automatically sent to the DT Platform to be centrally stored.

4 COGITO Components – Requirements and Specifications

A high-level representation of data exchange requirements among the COGITO components per use case has been introduced in Section 3. In this section, all these requirements are consolidated in input and output requirements per COGITO component, using a "Functional, Non-Functional Requirements and Interfaces" table template. Initially, the template aims to collect information about the hardware and software requirements of each component, the programming language(s), and the status of its development. Furthermore, it provides a draft version of each component's functional and non-functional requirements. Finally, in the table template, inputs and outputs of each component are listed, providing details about the format, method, endpoint and protocol for each data type and interface.

4.1 Visual Data Pre-processing

Table 1 presents the functional, non-functional requirements and the interfaces of the Visual Data Pre-Processing tool with other components of COGITO. It is a tool that is being developed from scratch and programmed utilising the Node.js for the backend development, Angular for the user interface development and MySQL as a relational database for storing the necessary data. Concerning the functional requirements of the tool, it must provide methods for storing raw data in a local database, both visual and geometric, as well as methods for pre-processing this data such as image smoothing (Gaussian blurring) and enhancing (contrast and brightness modification, cropping, rescaling). This data can be acquired using laser scanners (point clouds) and static cameras, mobile phones or Augmented Reality goggles (2D images). The Visual Data Pre-processing tool interacts only with the DT Platform and the Visual Data Acquisition tools. User credentials and authorisation, projects' information, building components data and work orders are received from the DT platform, while processed visual data and point cloud data constitute the outputs of the tool. DigiTAR can act as a Visual Data Acquisition tool and therefore the Visual Data Pre-Processing tool can send processed visual data directly to it.

Table 1 – Visual Data Pre-processing: Functional, Non-Functional Requirements and Interfaces

General Information				
Programming Language(s)		Node.js, Angular, MySQL		
Hardware Requirements		Physical or VM Server, Device with a web browser		
Software Requirements		Nodejs, Opencv, 4nodejsMySQL, Web Browser		
Development Status		Developed from scratch		
Functional and Non-Functional Requirements				
Functional	Req-1.1	Stores the raw data in a local database		
	Req-1.2	Processes the raw visual data input (Smoothing, enhancing)		
	Req-1.3	Sends Images or Point Clouds and associated data to DT Platform		
Non-Functional	Req-2.1	Web based App		
	Req-2.2	Offers efficiently structured database to store and retrieve data		
	Req-2.3	Reliability		
	Req-2.4	Scalability		
	Req-2.5	Performance		
	Req-2.6	Security		
	Req-2.7	Data Integrity		
Component Dependencies				
Internal Dependencies	Dep-1.1	Visual Data Acquisition Tools		
	Dep-1.2	DT Platform		
External Dependencies	Dep-2.1	Image Processing libraries (Opencv4nodejs)		
	Dep-2.2	MySQL		
Interfaces				
Input Data	Input-1	Received from: Data Acquisition Tools, incl. the "Pre-processing" operation mode of DigiTAR (raw visual data)	Format	Image file data, point cloud data & JSON
			Method	GET/POST
			Endpoint	REST API

			Protocol	HTTPS
	Input-2	Received from: DT Platform (user credentials and authorisation)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
	Input-3	Received from: DT Platform (projects' information)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
	Input-4	Received from: DT Platform (3D BIM model, building components)	Format	IFC
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
	Input-5	Received from: DT Platform (work orders/tasks reported by WODM)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
Output Data	Output-1	Sent to: DT Platform (processed visual data & metadata)	Format	Image file data & JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
	Output-2	Sent to: DT Platform (processed point cloud data & metadata)	Format	E57, PLY, MTL & JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
	Output-3	Sent to: DigiTAR (processed visual data & metadata)	Format	Image file data & JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS

4.2 IoT Data Pre-processing

The IoT Data Pre-Processing tool (see Table 2) ingests raw IoT data from sensorial devices installed or worn on the construction site and generates timeseries data to be conveyed and stored in the COGITO Digital Twin platform. Currently, solely resources tracking (location) data have been requested as IoT data input to other COGITO components.

Table 2 – IoT Data Pre-processing: Functional, Non-Functional Requirements and Interfaces

General Information		
Programming Language(s)		Python, Java
Hardware Requirements		IoT hardware; physical server(s) & cloud Infrastructure
Software Requirements		Docker/Kubernetes, Nginx/Traefik Proxy
Development Status		Developed from scratch
Functional and Non-Functional Requirements		
Functional	Req-1.1	Data management (Cleansing, Grouping, etc.)
	Req-1.2	Backup
	Req-1.3	Authorisation and encryption, pseudo-anonymisation
Non-Functional	Req-2.1	Interoperability
	Req-2.2	Security
	Req-2.3	Performance
	Req-2.4	Scalability
Component Dependencies		
Internal Dependencies	Dep-1.1	IoT Real-Time Location System (RTLS) Solution, e.g., Quuppa
	Dep-1.2	DT Platform

External Dependencies	Dep-2.1	Apache Tomcat, Apache Kafka (or Redpanda)		
	Dep-2.2	Python & Python third-party libraries		
	Dep-2.3	PostgreSQL with TimescaleDB and PostGIS extensions		
	Dep-2.4	Prometheus, Grafana		
Interfaces				
Input Data	Input-1	Received from: IoT Solution (raw IoT data)	Format	JSON
			Method	None/GET
			Endpoint	None/REST API
			Protocol	UDP/MQTT/HTTPS
Output Data	Output-1	Available to: DT Platform (Pre-processed resource tracking/location data)	Format	JSON
			Method	None/GET/POST
			Endpoint	Kafka Topics/REST APIs
			Protocol	Kafka/HTTPS

4.3 Digital Twin Platform – DT Platform

The DT Platform lies at the core of the COGITO solution and is responsible for implementing an information management solution that aims to enable interoperability with existing standards and ontologies covering various domains. It is a cloud-based and semantically enabled data integration middleware that includes a comprehensive suite of services responsible for loading, populating and managing data used by the various COGITO applications. The high level specifications and interfaces of the DT Platform with other components are presented in Table 3.

Table 3 – DT Platform: Functional, Non-Functional Requirements and Interfaces

General Information				
Programming Language(s)		Java EE, JavaScript, C++		
Hardware Requirements		Devices with internet connectivity (servers, computers, mobile phones, tablets, etc.)		
Software Requirements		Modern web browsers, software and applications implementing REST clients		
Development Status		Partially developed		
Function and Non-Functional Requirements				
Functional	Req-1.1	Authenticates users and applications		
	Req-1.2	Receives as-planned data (BIM models, construction schedule, available resources)		
	Req-1.3	Receives as-built data (video, images, and point-clouds)		
	Req-1.4	Handles real-time data of location tracking sensors		
	Req-1.5	Populates COGITO's ontology		
	Req-1.6	Orchestrates the execution of the included ETL services		
	Req-1.7	Manages the data requests of the COGITO applications by providing configurable APIs and the execution environment		
Non-Functional	Req-2.1	Scalability		
	Req-2.2	Responsiveness		
	Req-2.3	Security		
	Req-2.4	High availability		
Component Dependencies				
External Dependencies	Dep-2.1	Spring Framework Ecosystem		
	Dep-2.2	Akka Toolkit, Spring Integration		
	Dep-2.3	InfluxDB, Apache Fuseki, MySQL, Redis DB, Virtuoso		
	Dep-2.4	Apache ActiveMQ Artemis		
	Dep-2.5	Keycloak Identity and Access Management		
Interfaces				
Input Data	Input-1	Received from:	Format	JPG & JSON

		Visual Data Pre-processing (Processed visual data & metadata)	Method	POST
			Endpoint	REST API
			Protocol	HTTPS
	Input-2	Received from: Visual Data Pre-processing (Processed point cloud data & metadata)	Format	E57, PLY, MTL & JSON
			Method	POST
			Endpoint	REST API
			Protocol	HTTPS
	Input-3	Received from: IoT Data Pre-Processing (real-time location tracking data)	Format	None/GET/POST
			Method	Kafka Topics/REST APIs
			Endpoint	Kafka/HTTPS
			Protocol	None/GET/POST
	Input-4	Received from: IoT Data Pre-Processing (static configuration data)	Format	JSON
			Method	POST
			Endpoint	REST API
			Protocol	HTTPS
	Input-5	Received from: PMS (workflows, workflow's tasks, location, and resources)	Format	JSON
			Method	POST
			Endpoint	REST API
			Protocol	HTTPS
	Input-6	Received from: WODM (workorders, equipment and human instances)	Format	JSON
			Method	POST
			Endpoint	REST API
			Protocol	HTTPS
	Input-7	Received from: WODM (tasks' progress and metadata)	Format	JSON
			Method	POST
			Endpoint	REST API
			Protocol	HTTPS
	Input-8	Received from: GeometricQC (QC results and relevant metadata)	Format	JSON
			Method	POST
			Endpoint	REST API
			Protocol	HTTPS
	Input-9	Received from: VisualQC (Defects and metadata)	Format	JSON
			Method	POST
			Endpoint	REST API
			Protocol	HTTPS
	Input-10	Received from: SafeConAI (BIM data enhanced with health and safety information)	Format	IFC
			Method	POST
			Endpoint	REST API
			Protocol	HTTPS
	Input-11	Received from: VirtualSafety (Training feedback)	Format	JSON
			Method	POST
			Endpoint	REST API
			Protocol	HTTPS
	Input-12	Received from: DigiTAR (Defect confirmation & remedial works)	Format	JSON
			Method	POST
			Endpoint	REST API
			Protocol	HTTPS
	Input-13	Received from: DigiTAR (Hazard confirmation & mitigation works)	Format	JSON
			Method	POST
			Endpoint	REST API
			Protocol	HTTPS
Output Data	Output-1	Sent to: Visual Data Pre-processing, WODM, WOEA, PMS, SafeConAI, DCC, DigiTAR	Format	JSON
			Method	POST
			Endpoint	REST API
			Protocol	HTTPS

		(User credentials and authorisation)		
	Output -2	Sent to: Visual Data Pre-processing (Building components)	Format	IFC, OBJ
			Method	POST
			Endpoint	REST API
			Protocol	HTTPS
	Output -3	Sent to: Visual Data Pre-processing, DCC (work orders/tasks reported by WODM)	Format	JSON
			Method	POST
			Endpoint	REST API
			Protocol	HTTPS
	Output-4	Sent to: WODM, PMS (as-planned resources)	Format	JSON
			Method	POST
			Endpoint	REST API
			Protocol	HTTPS
	Output-5	Sent to: PMS, GeometricQC, SafeConAI, ProActiveSafety, VirtualSafety, DCC, DigiTAR (3D BIM Snapshots)	Format	IFC, OBJ
			Method	POST
			Endpoint	REST API
			Protocol	HTTPS
	Output-6	Sent to: ProActiveSafety, DCC (real time location tracking data)	Format	JSON
			Method	Publish/Subscribe
			Endpoint	Message Broker
			Protocol	MQTT
	Output-7	Sent to: PMS, WODM (location tracking configuration data)	Format	JSON
			Method	POST
			Endpoint	REST API
			Protocol	HTTPS
	Output-8	Sent to: PMS, DCC, DigiTAR (quality control and H&S issues relevant to the ongoing tasks)	Format	JSON
			Method	POST
			Endpoint	REST API
			Protocol	HTTPS
	Output-9	Sent to: Geometric QC (point clouds)	Format	E57, PLY
			Method	POST
			Endpoint	REST API
			Protocol	HTTPS
	Output-10	Sent to: VisualQC (Notification for visual data availability)	Format	JSON
			Method	Publish/Subscribe
			Endpoint	Message Broker
			Protocol	AMQP,MQTT
	Output-11	Sent to: VisualQC (Visual data and metadata)	Format	JPEG & JSON
			Method	POST
			Endpoint	REST API
			Protocol	HTTPS

4.4 Work Order Definition and Monitoring – WODM

As presented in Table 4, the WODM tool is a partly developed solution (i3D platform) for work orders definition and monitoring, programmed in Angular 6. This solution is extended and properly adapted to be compatible with the DT platform and interact with the DT Platform, PMS, SLAM, BCSC and WOEA components. These interactions have been realised based on the UC-1.1 and UC-1.2 sequence diagrams, illustrated in Sections 3.1 and 3.2 respectively. Primary inputs to the tool are the process model(s), the Service Level Agreements (SLAs) and their performance, the as-planned resources (number of workers, their roles, and appropriate equipment), the tasks' progress reported by the on-site crew, and user credentials and authorisation. The tasks assignment and progress, and the work orders with associated SLAs & KPIs, both enriched with relevant metadata, constitute the main outputs of the WODM tool.

Table 4 – WODM: Functional, Non-Functional Requirements and Interfaces

General Information				
Programming Language(s)		Angular 6		
Hardware Requirements		A device with a web browser		
Software Requirements		Web browser		
Development Status		Partly developed		
Function and Non-Functional Requirements				
Functional	Req-1.1	Connect and Authenticate to the DT platform (User login)		
	Req-1.2	Create work orders from workflows		
	Req-1.3	Assign multiple workers, allocate resources and associate SLAs to work orders and tasks		
	Req-1.4	Monitor and report the work order progress		
	Req-1.5	Update work orders and tasks details		
Non-Functional	Req-2.1	User-friendly UI		
	Req-2.2	Scalability		
	Req-2.3	Responsiveness		
	Req-2.4	Stability		
	Req-2.5	Security		
Component Dependencies				
Internal Dependencies	Dep-1.1	DT Platform		
	Dep-1.2	PMS		
	Dep-1.3	SLAM		
	Dep-1.4	BCSC		
	Dep-1.5	WOEA		
External Dependencies	Dep-2.1	Postgre database		
	Dep-2-2	I3D platform		
Interfaces				
Input Data	Input-1	Received from: DT Platform (process model)	Format	BPMN/JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
	Input-2	Received from: SLAM (SLAs and KPIs)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
	Input-3	Received from: BCSC (SLAs performance)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
	Input-4	Received from: DT Platform (IoT static configuration data, process model)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
	Input-5	Received from: WOEA (tasks' progress)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
	Input-6	Received from: DT Platform (user credentials and authorisation)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
		Sent to:	Format	JSON

Output Data	Output-1	SLAM (work orders with associated SLAs and KPIs)	Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
	Output-2	Sent to: PMS (tasks' progress)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
	Output-3	Sent to: BCSC (tasks' progress)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
	Output-4	Sent to: DT Platform (work orders and metadata)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
	Output-5	Sent to: DT Platform (tasks' progress and metadata)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
	Output-6	Sent to: WOEA (assigned tasks)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS

4.5 Work Order Execution Assistance – WOEA

Table 5 summarises the functional, non-functional requirements and interfaces of the WOEA application with other components of COGITO. It is a Unity3D application that is partially developed to work in collaboration with the WODM tool. The application is planned to be running on Android smart or Windows platform devices. It is designed to interact with the WODM tool, to exchange information about tasks' assignments and tasks' progress reports, and the DT Platform (user credentials and access control).

Table 5 – WOEA tool: Functional, Non-Functional Requirements and Interfaces

General Information		
Programming Language(s)		C# (Unity3D)
Hardware Requirements		Android smart device or Windows platform device (smart glasses, AR glasses, smart phone, tablet)
Software Requirements		Android 8.0, API 26 or higher
Development Status		Partially developed
Functional and Non-Functional Requirements		
Functional	Req-1.1	Connect and Authenticate to the DT platform (User login)
	Req-1.2	Display assigned tasks
	Req-1.2	Report progress of assigned tasks
Non-Functional	Req-2.1	User-friendly
	Req-2.2	Scalability
	Req-2.3	Stability
	Req-2.4	Multiplatform
	Req-2.5	Security
Component Dependencies		
Internal Dependencies	Dep-1.1	WODM
	Dep-1.2	DT Platform
External Dependencies	Dep-2.1	Unity 2018.4.29f1

Interfaces				
Input Data	Input-1	Received from: WODM (assigned tasks)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
	Input-2	Received from: DT Platform (user credentials and authorisation)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
Output Data	Output-1	Sent to: WODM (tasks' progress)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS

4.6 Process Modelling and Simulation – PMS

The PMS tool is used to define and simulate both the construction business process model and the operative workflow model. It allows the users to identify process steps critical for successfully implementing the project, exposing optimisation opportunities to minimise time and/or cost. The simulation models is combined with real-world data and are supported by data mining algorithms and statistical methods that allow the calibration of the simulation model to the actual process occurring on the construction site. For the modelling and simulation functionalities of the tool, the ADOxx meta-model platform is used.

Table 6 presents the functional and non-functional requirements, and the interactions of the PMS tool with other components of the COGITO solution. Apart from the ADOxx meta-model platform, Java, JavaScript, R, Python and Hugin programming languages are used to develop additional simulation and optimisation functionalities from scratch. Concerning the data exchange of the PMS tool with other components, in the planning phase, the PMS tool is used to populate the process/workflow model for the construction project in BPMN format, based on the 4D BIM data that is received from the DT platform. This process/workflow model is sent to the DT Platform. During the construction phase, resources' location (acquired by the IoT Data Pre-processing module), quality control and H&S issues relevant to the ongoing tasks constitute additional information that are queried from the DT platform. The tasks' progress is also required as input, received from the WODM tool.

Table 6 – PMS: Functional, Non-Functional Requirements and Interfaces

General Information		
Programming Language(s)		Java, JavaScript, R, Python, Hugin
Hardware Requirements		Physical or VM Server
Software Requirements		Web Server (Apache), Browser (Chrome, Firefox, Edge)
Development Status		Partially developed
Functional and Non-Functional Requirements		
Functional	Req-1.1	Construct workflow and simulation model and populate with historical data
	Req-1.2	Update simulation model using real-time data from WODM
	Req-1.3	Connect, Authenticate, and retrieve projects from the DT platform
	Req-1.4	Estimates project progress
	Req-1.5	Receive task progress from WODM
	Req-1.6	Output updated estimates of project progress to WODM according to real-time data and performed optimisation
Non-Functional	Req-1.7	Web-based app
	Req-2.1	User-friendly
	Req-2.2	Scalability
	Req-2.3	Security

Component Dependencies				
Internal Dependencies	Dep-1.1	DT platform		
	Dep-1.2	WODM		
External Dependencies	Dep-2.1	ADOxx meta-model platform		
	Dep-2.2	Data analytics and statistical methods libraries		
Interfaces				
Input Data	Input-1	Received from: DT Platform (As planned 4D BIM data)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
	Input-2	Received from: DT Platform (user credentials, authorisation, and projects list)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
	Input-3	Received from: WODM (tasks' progress)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
	Input-3	Received from: DT Platform (resources location data)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
	Input-4	Received from: DT Platform (quality control and H&S issues relevant to the ongoing tasks)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
Output Data	Output-1	Sent to: DT Platform (process models)	Format	BPMN/JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
	Output-2	Sent to: WODM (notification on updated process)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS

4.7 Service-Level Agreement Manager – SLAM

The SLAM component (see Table 7) provides a database with already designed SLAs that include predefined rules and KPIs. WODM fetches the SLAs through the SLAM to bind work orders with associated SLAs and KPIs. Then WODM informs the SLAM of the results and SLAM saves the SLAs with the respective Stakeholders on those mentioned above. The BCSC can fetch the saved work orders with associated SLAs and KPIs to initiate and instantiate the Smart Contract operation. These interactions have been obtained based on the UC-1.1 sequence diagram (see Section 3.1). The SLAM component is developed from scratch.

Table 7 – SLAM: Functional, Non-Functional Requirements and Interfaces

General Information		
Programming Language(s)		Go, Rust
Hardware Requirements		Physical or VM Server
Software Requirements		Linux Operating System
Development Status		Developed from scratch
Functional and Non-Functional Requirements		
Functional	Req-1.1	Provide Smart Contracts with predefined SLAs and KPI description
	Req-1.2	Provide SLAs to authorized users of WODM UI
	Req-1.3	Create SLA - Stakeholder bundles

	Req-1.4	Save complete SLAs		
Non-Functional	Req-2.1	Reusability		
	Req-2.2	Scalability		
	Req-2.3	Security		
Component Dependencies				
Internal Dependencies	Dep-1.1	WODM		
	Dep-1.2	BCSC		
External Dependencies	Dep-2.1	Candidates Database (under investigation the usage of Postgres)		
	Dep-2.2	CosmWasm, Rust, Cargo, Docker, Starport		
Interfaces				
Input Data	Input-1	Received from: WODM (work orders with associated SLAs and KPIs)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
Output Data	Output-1	Sent to: WODM (SLAs and KPIs)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
	Output-2	Sent to: BCSC (work orders with associated SLAs and KPIs)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS

4.8 BlockChain Platform – BCSC

The Blockchain Platform is an immutable database system that can execute smart contracts. It allows the deployment of smart contracts, through the SLAM component (see Table 8). It interacts with the WODM tool and based on the operative workflow model. It provides the blockchain based smart contracts to enhance transparency and provide trusted means to verify the completion of construction tasks. To deliver its scope, it receives the work orders with associated SLAs and KPIs from SLAM, the calculated KPIs from WODM, and it sends the SLAs performance to WODM. These input and output requirements have been extracted from the sequence diagrams of UC-2.1 and UC-2.2, presented in Sections 3.3 and 3.4 respectively.

Table 8 – BlockChain Platform: Functional, Non-Functional Requirements and Interfaces

General Information		
Programming Language(s)		Go (with Java Plugins)
Hardware Requirements		Physical or VM Server
Software Requirements		Linux Operating System
Development Status		Developed from scratch
Functional and Non-Functional Requirements		
Functional	Req-1.1	Blockchain-Smart Contract Network will instantiate depending on SLA – Stakeholder bundle
	Req-1.2	Store transaction data
	Req-1.3	Receive Smart Contract inputs from SLA Manager
	Req-1.4	Update Smart Contracts
	Req-1.5	Receive Task Related Inputs from WODM
	Req-1.6	Send Smart Contract Result to WODM
Non-Functional	Req-2.1	Scalability
	Req-2.2	Stability
	Req-2.3	Security
Component Dependencies		

Internal Dependencies	Dep-1.1	SLA Manager		
	Dep-1.2	WODM		
External Dependencies	Dep-2.1	Cosmos SDK		
	Dep-2.2	Terndermind Starport, CosmWasm, IBC, Go, Docker		
Interfaces				
Input Data	Input-1	Received from: SLAM (work orders with associated SLAs and KPIs)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTP
	Input-2	Received from: WODM (updated KPIs)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTP
Output Data	Output-1	Sent to: WODM (SLA performance)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTP

4.9 Geometric Quality Control – GeometricQC

As Table 9 indicates, the GeomtricQC tool is developed from scratch, planned to be deployed on a Physical or Virtual Machine Linux server, and programmed utilising the C++, Python and C# programming languages. Widely used BIM, Point Cloud, and 3D Data Processing libraries constitute the technology stack of the tool. Regarding its functional requirements, it must provide methods for loading 4D BIM and point cloud data, detecting objects in the point cloud, detecting defects on digitalised dimensional quality control specifications, and communicating with the DT Platform. As depicted in the sequence diagram of UC-2.1 (see Figure 8), the GeometricQC tool interacts only with the DT Platform. It receives the as-planned 4D BIM data and the processed point cloud data from the DT Platform, and it sends geometric quality control results, and relevant metadata, back to the DT Platform.

Table 9 – GeometricQC tool: Functional, Non-Functional Requirements and Interfaces

General Information		
Programming Language(s)		C++ Python C#
Hardware Requirements		Physical or VM Server (Windows/Linux)
Software Requirements		Multiple C++ libraries (Boost, Eigen, Open3D, IfcOpenShell)
Development Status		Developed from scratch
Functional and Non-Functional Requirements		
Functional	Req-1.1	Loading as planned 4D BIM and point cloud data
	Req-1.2	Object detection in point cloud
	Req-1.3	Generate QC task list from input 4D BIM based on digitalised dimensional QC specifications
	Req-1.4	Conduct QC task given BIM and input point cloud
	Req-1.5	DT platform/NS notification to execute QC
Non-Functional	Req-2.1	Scalability
	Req-2.2	Reusability
	Req-2.3	Interoperability
Component Dependencies		
Internal Dependencies	Dep-1.1	DT Platform
	Dep-1.2	Geometric Data Acquisition Tools
	Dep-2.1	Visual Studio 2019

External Dependencies	Dep-2.2	C++ 14		
	Dep-2.3	Boost		
	Dep-2.4	Open3D		
	Dep-2.5	Eigen		
	Dep-2.6	IfcOpenShell		
Interfaces				
Input Data	Input-1	Received from: DT Platform (as planned 4D BIM data)	Format	IFC, OBJ, JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
	Input-2	Received from: DT Platform (point cloud)	Format	E57, PLY
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
	Output-1	Sent to: DT Platform (QC results and relevant metadata)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS

4.10 Visual Quality Control – VisualQC

Following the sequence diagram of UC-2.2 (see Figure 9), the DT Platform constitutes the unique COGITO component that the VisualQC tool interacts with. The VisualQC tool receives notifications for visual data availability, visual data and metadata from the DT Platform and sends detected defects and relevant metadata back to the DT Platform.

As Table 10 presents, the VisualQC tool is being developed from scratch, planned to be deployed on a Physical or Virtual Machine server, and programmed utilising the Python programming language. Deep learning and image processing libraries constitute the main external dependencies of the tool.

Table 10 – VisualQC tool: Functional, Non-Functional Requirements and Interfaces

General Information		
Programming Language(s)		Python
Hardware Requirements		Physical or virtual machine (16 GB RAM, a modern CPU with 4 Cores and SSD preferably)
Software Requirements		Web Server (Flask), Anaconda Python 3.8
Development Status		Developed from scratch
Functional and Non-Functional Requirements		
Functional	Req-1.1	Connects and Authenticate to the DT Platform
	Req-1.2	Detects defects based on trained models
	Req-1.3	Generates reports for defect detection results
	Req-1.4	Sends defect detection results to the DT Platform
Non-Functional	Req-2.1	Security
	Req-2.2	The component must use APIs for all expected operations
	Req-2.3	The component must have low latency
	Req-2.4	Scalability
	Req-2.5	Strong CPU/GPU is needed for image processing
Component Dependencies		
Internal Dependencies	Dep-1.1	DT Platform
External Dependencies	Dep-2.1	Deep learning libraries (Tensorflow, Keras, Numpy etc.)
	Dep-2.2	Image Processing libraries (PIL/Pillow etc.)

Interfaces				
Input Data	Input-1	Received from: DT Platform (Notification for visual data availability)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
	Input-2	Received from: DT Platform (Visual data and metadata)	Format	Image file data & JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
Output Data	Output-1	Sent to: DT Platform (Defect detection results and metadata)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS

4.11 SafeConAI

The SafeConAI tool (see Table 11) is used to identify regions where (specific types of) hazards are, suggests and "injects" mitigation measures into the model. It uses as input the as planned or as-built 4D BIM data received from the DT platform, enhance them with safety information and communicate the enhanced 4D BIM data back to the DT platform. It also receives updated safety parameters as input from the ProActiveSafety tool. The aforementioned data exchange requirements have been extracted from the sequence diagrams of UC-3.1 and UC-3.2 (see Figure 10 and Figure 11, respectively).

Table 11 – SafeConAI: Functional, Non-Functional Requirements and Interfaces

General Information				
Programming Language(s)		C# C++ Python		
Hardware Requirements		TBD		
Software Requirements		GAMA PyTorch / Tensorflow		
Development Status		Partially developed		
Functional and Non-Functional Requirements				
Functional	Req-1.1	Loading as planned 4D BIM data		
	Req-1.2	Loading as built 4D BIM data		
	Req-1.3	Loading of known hazard zones		
	Req-1.4	Injection of potential mitigation measures		
	Req-1.5	Exploration of potential changes of hazard zones based on the expected changes on the construction site (i.e., construction progress or mitigation measures)		
	Req-1.6	Provide information about accidents and hazards		
Non-Functional	Req-2.1	Scalability		
	Req-2.2	Usability		
Component Dependencies				
Internal Dependencies	Dep-1.1	DT Platform		
	Dep-1.2	ProActiveSafety		
External Dependencies	Dep-2.1	GAMA		
Interfaces				
Input Data	Input-1	Received from: DT Platform (as planned 4D BIM data)	Format	IFC
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS

	Input-2	Received from: DT Platform (user credentials and authorisation)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
	Input-3	Received from: ProactiveSafety (updated safety parameters)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
Output Data	Output-1	Sent to: DT Platform (4D BIM data enhanced with safety information)	Format	IFC/XML/JSON/PLY
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS

4.12 ProActiveSafety

The ProActiveSafety tool (see Table 12) utilises behavioural data of resources (equipment and personnel) on the construction site to avoid close-calls, accidents, and collateral damage. It also runs statistics and update the safety parameters. Based on the sequence diagram of UC-3.2 (see Figure 11), to fulfil its requirements, the ProActiveSafety tool receives near real-time resources tracking data and the 4D BIM data enhanced with safety information from the DT platform, and it sends hazards alarms to the WOEA application and updated safety parameters to the SafeConAI tool.

Table 12 – ProActiveSafety: Functional, Non-Functional Requirements and Interfaces

General Information				
Programming Language(s)		C# C++ Python Proprietary (GAMA)		
Hardware Requirements		Potentially high-end graphics card for prediction algorithms relying on deep learning.		
Software Requirements		(GAMA) PyTorch / Tensorflow		
Development Status		Developed from scratch		
Functional and Non-Functional Requirements				
Functional	Req-1.1	Loading as planned 4D BIM data		
	Req-1.2	Loading as performed 4D BIM data		
	Req-1.3	Interfacing with location data from DT platform		
	Req-1.4	Extrapolation and estimation of future trajectories		
	Req-1.5	Warning issued in expected close-calls		
Non-Functional	Req-2.1	Scalability		
	Req-2.2	Usability		
Component Dependencies				
Internal Dependencies	Dep-1.1	DT Platform		
External Dependencies	Dep-2.1	PyTorch / Tensorflow (Python)		
Interfaces				
Input Data	Input-1	Received from: DT Platform (resources tracking data)	Format	PLY/JSON/XML
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTP
	Input-2	Received from:	Format	IFC
			Method	GET/POST

		DT Platform (4D BIM data enhanced with safety information)	Endpoint	REST API
			Protocol	HTTP
Output Data	Output-1	Sent to: SafeConAI (Updated safety parameters/hazard zones)	Format	PLY / JSON / XML
			Method	POST/GET
			Endpoint	REST API
			Protocol	HTTP
	Output-2	Sent to: WOEA (hazards alarms)	Format	TBD
			Method	TBD
			Endpoint	TBD
			Protocol	TBD

4.13 VirtualSafety

The VirtualSafety application provides personalised construction safety education and training, focusing on the 'Top 4' hazards: Slips/trips/falls from height, caught-in between, struck-by, and electrocution. The VR provides an easy-to-use, reliable safe learning environment and technology that assists advanced HSE decision making and provide personalized feedback in a safe learning environment. It is a Unity3D application running on Windows PCs or VR headsets. It is a standalone application for training purposes. It interacts with the DT platform only to receive the 4D BIM data enhanced with safety information of a construction project and send the training feedback.

Table 13 – VirtualSafety: Functional, Non-Functional Requirements and Interfaces

General Information				
Programming Language(s)		Java / C# / C++		
Hardware Requirements		Standard PC, VR headset (e.g., Oculus Rift or Quest, HTC Vive)		
Software Requirements		Unity		
Development Status		Proof of concept implemented		
Functional and Non-Functional Requirements				
Functional	Req-1.1	Loading as-planned / as-performed 4D BIM model		
	Req-1.2	Loading of safety issues		
	Req-1.3	Definition and permanent storage of safety scenarios		
	Req-1.4	Loading of predefined safety scenarios		
Non-Functional	Req-2.1	Performance		
	Req-2.2	Usability		
Component Dependencies				
Internal Dependencies	Dep-1.1	DT platform		
External Dependencies	Dep-2.1	Unity3D		
Interfaces				
Input Data	Input-1	Received from: DT Platform (4D BIM data enhanced with safety information)	Format	IFC/XML/JSON
			Method	POST/GET
			Endpoint	REST API
			Protocol	HTTPS
Output Data	Output-1	Sent to: DT Platform (Training feedback)	Format	PLY/JSON/XML
			Method	POST/GET
			Endpoint	REST API
			Protocol	HTTPS

4.14 Digital Command Centre – DCC

The DCC (see Table 14) is a Unity WebGL application for the PMs to visualise and navigate the digital twin data. It renders the 4D BIM model, resources tracking data and other annotations generated by the QC, H&S and Workflow tools. Prerequisites for visualising such data is the as-planned (4D BIM) and as-built (resources tracking, images and QC, H&S and Workflow tools' annotation) data to be compliant to the COGITO data models and available through the DT Platform's endpoints. There are no outputs planned to be exported from this tool.

Table 14 – DCC: Functional, Non-Functional Requirements and Interfaces

General Information				
Programming Language(s)		C#		
Hardware Requirements		Physical or VM Server (Linux)		
Software Requirements		Web Server (Apache), Browser (Chrome, Firefox, Edge), Unity		
Development Status		Developed from scratch		
Functional and Non-Functional Requirements				
Functional	Req-1.1	Connect and Authenticate to the DT platform (User login)		
	Req-1.2	Browse and select project from the list of available projects of the DT platform		
	Req-1.3	3D visualisation of the infrastructure's geometry (panning, rotation, camera movement and placement, walkthrough) (layer-0)		
	Req-1.4	BIM-elements tree-view and element's selection (layer-0)		
	Req-1.5	Resources tracking data display (layer-1)		
	Req-1.6	QC defects display (layer-2)		
	Req-1.7	H&S issues display (layer-3)		
	Req-1.8	tasks progress display (layer-4)		
Non-Functional	Req-2.1	Web-based App		
	Req-2.2	Scalability		
	Req-2.3	Reusability		
	Req-2.4	Interoperability		
	Req-2.5	Security		
	Req-2.6	User-friendly		
	Req-2.7	Performance		
Component Dependencies				
Internal Dependencies	Dep-1.1	DT Platform		
External Dependencies	Dep-2.1	IfcOpenShell IfcConvert 0.6.0b0		
	Dep-2.2	Unity (v2020.2)		
	Dep-2.3	New UI Widgets (unity AssetStore)		
Interfaces				
Input Data	Input-1	Received from: DT Platform (4D BIM data)	Format	IFC
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
	Input-2	Received from: DT Platform (resources tracking data)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
	Input-3	Received from: DT Platform (QC results)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS

	Input-4	Received from: DT Platform (H&S results)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
		Received from: DT Platform (workflow progress)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
Output Data	Output-1	Sent to: Not Applicable (N/A)	Format	N/A
			Method	N/A
			Endpoint	N/A
			Protocol	N/A

4.15 Digital Twin visualisation with Augmented Reality – DigiTAR

DigiTAR (see Table 15 is a Unity application for commercial AR head mounted displays (HMDs) to help to visualise and interact in situ with the output of the QC tools (location, type and severity of geometric and visual defects) and Safety tools (location and type of safety hazards and expected mitigation measures).

Table 15 – DigiTAR: Functional, Non-Functional Requirements and Interfaces

General Information		
Programming Language(s)		C#
Hardware Requirements		Augmented-Reality Smart Glasses with RGB-D camera and all the necessary sensors
Software Requirements		Unity
Development Status		Developed from scratch
Functional and Non-Functional Requirements		
Functional	Req-1.1	Maps the BIM 3D model enriched with defects and safety information, to the site.
	Req-1.2	Tracks every building component, defect and safety hazard that has been registered.
	Req-1.3	Determines user's position and orientation.
	Req-1.4	Confirms annotations about defects and safety hazards.
	Req-1.5	Creates annotations about safety hazards.
	Req-1.6	Creates task annotations for remedial work and safety hazard mitigation work.
	Req-1.7	Sends relevant information about annotations to DT platform.
	Req-1.8	Captures and sends visual data to Visual Data Pre-processing module.
	Req-1.9	Receives and displays processed visual data from Visual Data Pre-processing module.
Non-Functional	Req-2.1	User friendly interface
	Req-2.2	WiFi connection on site
	Req-2.3	Menu to discern the different functionalities
	Req-2.4	AR Application
	Req-2.5	Scalability
	Req-2.6	Reusability
	Req-2.7	Security
Component Dependencies		
Internal Dependencies	Dep-1.1	DT Platform
	Dep-1.2	Visual Data Pre-processing tool
External Dependencies	Dep-2.1	Unity (v2020.2)
	Dep-2.2	Xbim
	Dep-2.3	IfcOpenShell IfcConvert 0.6.0b0

	Dep-2.4	Mixed Reality Toolkit		
Interfaces				
Input Data	Input-1	Received from: DT Platform (User Credentials and Authorisation)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
	Input-2	Received from: DT Platform (3D BIM data)	Format	IFC
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
	Input-3	Received from: DT Platform (QC data)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
	Input-4	Received from: DT Platform (Safety data)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
	Input-5	Received from: Visual Data Pre-processing module (Processed visual data & metadata)	Format	Image file data and JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
Output Data	Output-1	Sent to: DT Platform (Defect confirmation & remedial works)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
	Output-2	Sent to: DT Platform (Hazard confirmation & mitigation works)	Format	JSON
			Method	GET/POST
			Endpoint	REST API
			Protocol	HTTPS
	Output-3	Sent to: Visual Data Pre-processing module (raw visual data)	Format	Format
			Method	Method
			Endpoint	Endpoint
			Protocol	Protocol

5 Deployment and Data Protection

5.1 Components diagrams

In UML, a component diagram depicts how components are linked together to form larger entities. In general, they are used to illustrate the structure of arbitrarily complex systems. In alignment with Section 4 and to verify that COGITO components' required functionalities are acceptable, the relevant diagrams are provided in the sub-sections below. These diagrams are also aimed to be used as a communication tool between the developers of the components and the relevant stakeholders involved in each UC that incorporates the components under investigation.

5.1.1 Visual Data Pre-processing

Figure 15 illustrates the UML component diagram of the visual data pre-processing tool. As shown below, the tool is composed of:

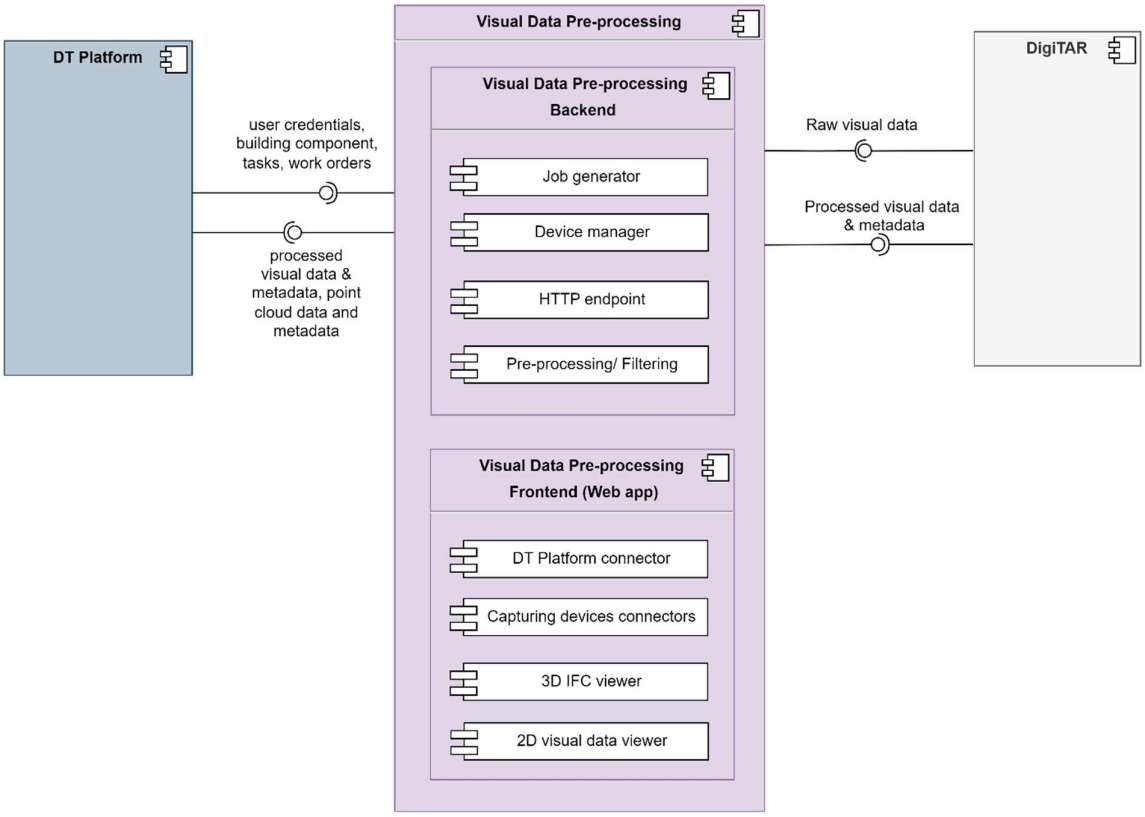


Figure 15 – Component diagram of the Visual Data Pre-processing tool

- the **Visual Data Pre-processing backend** that includes:
  - the **Job generator** responsible for generating new jobs including new data that will be sent for pre-processing and defect detection through the DT Platform;
  - the **Device manager** that organises and handles details about existing and new data capturing devices (i.e. position, orientation etc.);
  - the **HTTP endpoint** that enables the inter-communication of the Visual Data Pre-processing tool backend with the Visual Data Pre-processing web and mobile application;
  - the **Pre-processing/Filtering module** that implements filters at raw visual data to prepare it for the Visual QC tool; and
- the **Visual Data Pre-processing frontend** (web application) that includes:
  - the **DT Platform connector** which enables the communication with the DT Platform;

- the **Capturing devices connectors** that enable the communication with the DigiTAR tool and other capturing devices;
- the **3D IFC viewer** which renders the as-planned BIM data;
- the **2D visual data viewer** that visualises the raw and processed visual data.

5.1.2 IoT Data Pre-Processing

Figure 16 illustrates the UML component diagram of the IoT data pre-processing tool. As shown below, the tool is composed of:

- the **communication layer** responsible for managing the input/output communication and more particularly:
  - maintaining the connection with the proprietary IoT sensors/RTLS (e.g., *Quuppa*) and requesting (or receiving) the location tracking data (e.g., using a RESTful API/UDP packets/subscribing to a MQTT topic);
  - parsing the input data stream (commonly JSON) to a format appropriate for further processing; also generating the pre-processed data in JSON format to be communicated to the DT platform;
  - managing the available output REST API endpoints and publishing the near real-time pre-processed data to the Kafka broker;
- the **database layer:** pre-processed data is archived and accessed through this subcomponent. It consists of
  - a database suitable for timeseries data (e.g., Timeseries DB);
  - a database wrapper providing an abstract way to communicate with the application layer;
- the **application layer** constituting the main subcomponent of IoT pre-processing module; it is responsible for:
  - aggregating the available data;
  - pre-processing them based on a set of rules and algorithms;
  - managing the storage and retrieval of the pre-processed data;

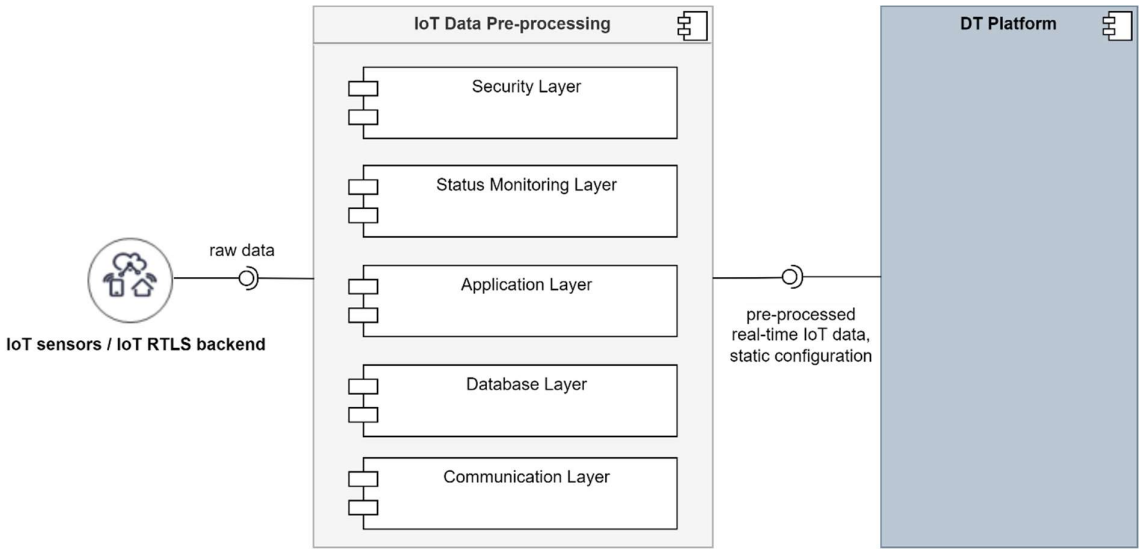


Figure 16 – Component diagram of the IoT Data Pre-processing tool

- the **status monitoring & alerting layer** responsible for:
  - monitoring the status, performance and resource usage of the several module's subcomponents;
  - providing visual feedback through dashboards, including the option to generate visual alerts; and
- the **security layer** that provides the security features required to ensure:

- authentication and access control: access to the pre-processed data is granted only to trusted parties (i.e., the DT platform) using well-established authentication mechanisms (e.g., token, credentials authentication, client certificates);
- data integrity and confidentiality: the data stream flows through an encrypted channel (e.g. TLS/VPN tunnel).

5.1.3 Digital Twin Platform

The design of the DT Platform delivered in "T7.1 - Digital Twin Platform Design & Interface Specification" is performed adopting the SOA design pattern, following a layer architecture.

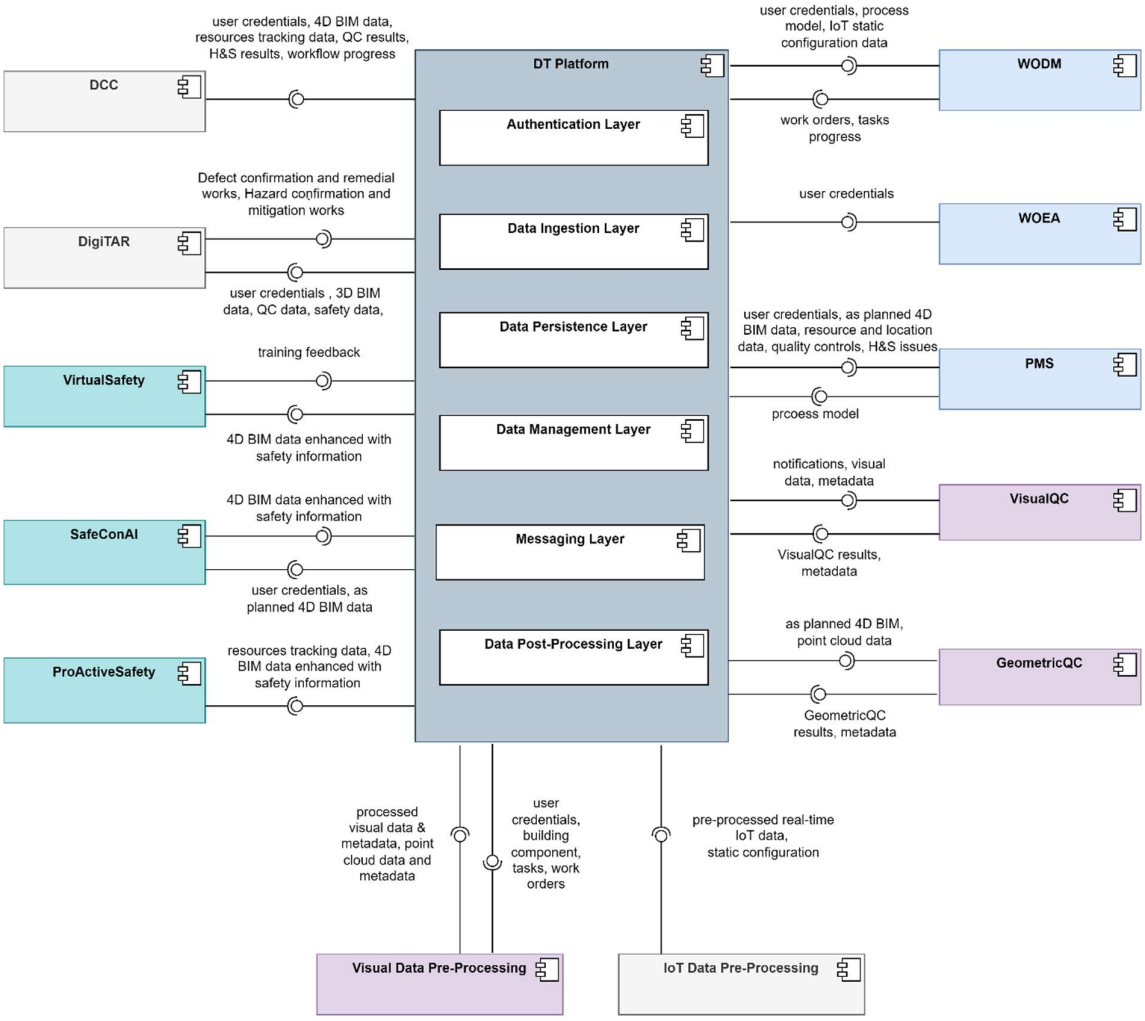


Figure 17 – Component diagram of the Digital Twin platform

The DTP follows a multi-layered architecture comprising six core layers (see Figure 17). Each layer contains a set of software components implementing various business logic operations to support the main objectives of the DTP. The layers of the DTP architecture and their functional roles has as follows:

- The **Authentication Layer** performs user management and authentication tasks using the open-source project Keycloak.
- The **Data Ingestion Layer** provides services to load and semantically link new datasets into the knowledge graph. This component is deployed on the cloud and implemented in Java EE utilizing Apache Jena and the Spring Framework Ecosystem.

- The **Data Persistence Layer** provides a shared storage data repository including graph and time-series databases deployed on a cloud infrastructure and implemented in Java EE unitizing Apache Fuseki, InfluxDB and the Spring Framework Ecosystem.
- The **Data Management Layer** responds to COGITO applications data requests, synchronizing the multiple parallel responses to data queries from these applications quickly and efficiently. It uses the W3C Web of Things standard to discover and retrieve information and resources from the DTP.
- The **Messaging Layer** coordinates data transformation operations and the responses to data queries from other COGITO applications. It contains the Apache ActiveMQ Artemis message broker for enabling asynchronous communications between the DTP services and the other COGITO applications using various messaging protocols such as Advanced Message Queueing Protocol (AMQP), Streaming Text-Oriented Messaging Protocol (STOMP) and Message Queue Telemetry Transport (MQTT).
- The **Data Post-Processing Layer** to perform ETL and model checking operations on COGITO's BIM data. The components of this layer are partly developed but will be extended to support the project's needs and integrated with the other DTP components. This component is deployed on the cloud and implemented in Java EE and C++.

5.1.4 Work Order Definition and Monitoring – WODM

Figure 18 illustrates the UML component diagram of the WODM tool. As shown below, the tool is composed of:

- the **Security Layer** responsible for user authentication and authorisation;
- the **Frontend Layer** that provides an interactive and responsive GUI allowing to create work orders, to assign workers, resources, and SLAs, to monitor progress, and to present reports;
- the **Backend Layer** that processes all information needed to create work orders, to assign workers, resources and SLAs, to monitor progress and to present reports; and
- the **Communication Layer** that orchestrates the communication with DT Platform, PMS, SLA Manager, BCSC and WOEA.

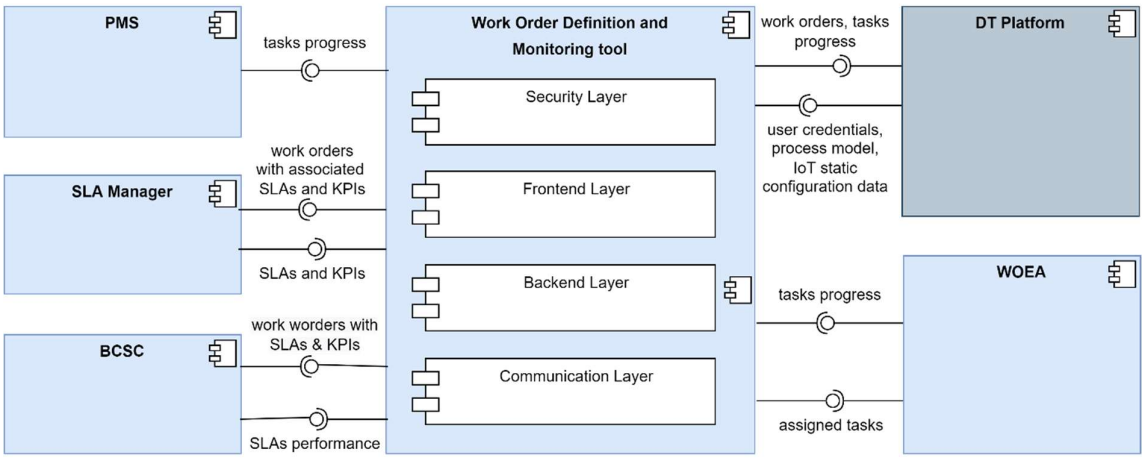


Figure 18 – Component diagram of the WODM tool

5.1.5 Work Order Execution Assistance – WOEA

Figure 19 illustrates the UML component diagram of the WOEA tool. As shown below, the tool is composed of:

- the **Security Layer** responsible for user authentication and authorisation;
- the **Application Layer** that displays and processes all information needed to perform work and to report work progress; and

- the **Communication Layer** that orchestrates communication with DT Platform and WODM.

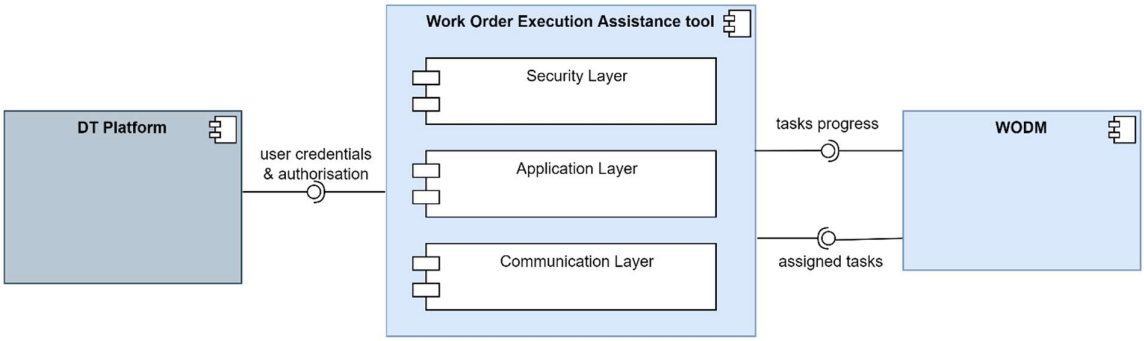


Figure 19 – Component diagram of the WOEA tool

5.1.6 Process Modelling and Simulation tool – PMS

Figure 20 illustrates the UML component diagram of the PMS tool. As shown below, the tool is composed of:

- the **Modelling component** that provides a modelling environment for the construction processes, workflows and KPIs;
- the **Simulation component** that simulates a construction process estimating costs and times;
- the **Optimisation component** that provides a mechanism to optimise the construction process during its execution; and
- the **Integration component** that allows the integration of the modelling, simulation and optimisation with the DT platform and the WODM.

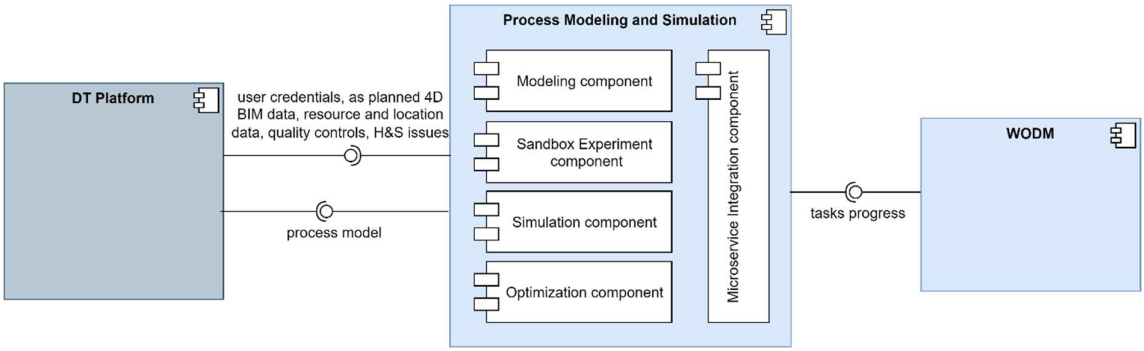


Figure 20 – Component diagram of the PMS tool

5.1.7 Service-Level Agreement Manager – SLAM

Figure 21 illustrates the UML component diagram of the SLA Manager tool. As shown below, the tool is composed of:

- the **Smart Contract Manager** which is the component responsible for establishing communication with WODM in order to retrieve information regarding tasks and stakeholders; it is also responsible for providing to WODM predefined KPIs; this information is then combined to create smart contract enabled SLAs;
- the **Smart Contract Orchestrator** that selects the appropriate nodes on the BS-SC to initiate the deployment of Smart Contracts;
- the **Security** subcomponent that provides the means for user authentication and authorisation;

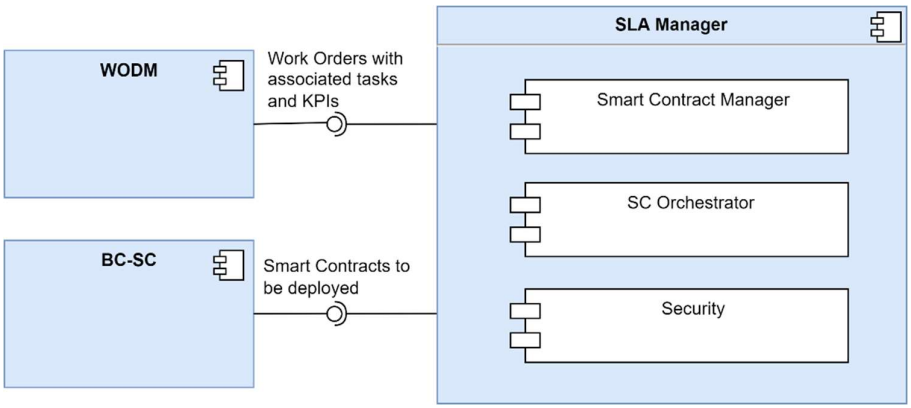


Figure 21 - Component diagram of the SLA Manager

5.1.8 BlockChain Platform – BCSC

Figure 22 illustrates the UML component diagram of the BlockChain – Smart Contract (BC-SC) Platform tool. As shown below, the tool is composed of:

- the **KPI Checker** which is the component that communicates with WODM in order to receive updated KPIs that are used for the execution of the respective smart contracts;
- the **Smart Contract Initiator** that receives smart contracts from the SLA Manager and initiates them on the appropriate nodes; and
- the **Smart Contracts** is the Blockchain plugin that is used to deploy and execute smart contracts. Smart Contract execution results, such as KPIs and predefined actions, is communicated to authorized parties (i.e., WODM) upon request.

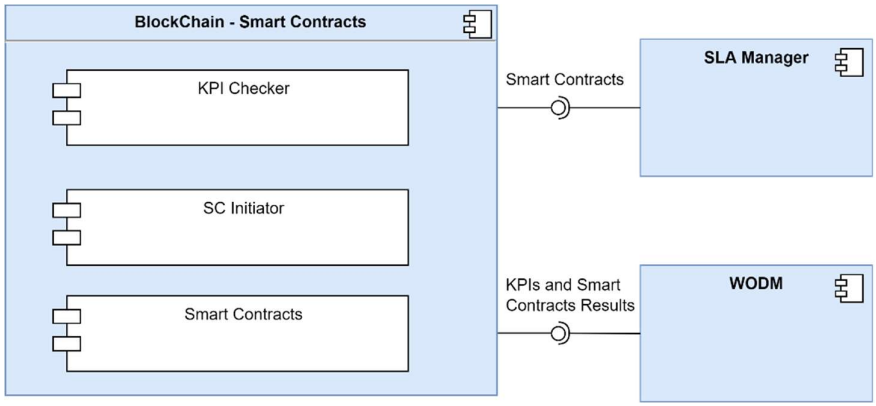


Figure 22 – Component diagram of the BCSC tool

5.1.9 Geometric Quality Control – GeometricQC

Figure 23 illustrates the UML component diagram of the GeometricQC tool. As shown below, the tool is composed of two main sub-components: (i) the **BIM file pre-processing** and (ii) the **GeomQC,** each one further de-composed to internal modules as follows:

- The **BIM file pre-processing** sub-component analyses the relationships between the building elements (adjacency, location, connections, etc.) to obtain and provide to the GeomQC – the main sub-component of the tool – the list of QC checks that is needed to be performed for each element. More specifically, the BIM file pre-processing includes:
  - the **building elements relationships' analyser** that extracts the elements from the as planned BIM file and generates a relationships network between them;

- the **QC regulation dictionary** providing a dictionary with the quality control checks that are needed according to the standards, regulations and client controls;
- the **Elements QC listing** that generates the list of QC needed for each element utilising the elements' relationships and the QC regulation dictionary provided by the building elements relationships' analyser and the QC regulation dictionary respectively.

- The **GeomQC** provides the main functionality of the tool through:
  - the **point cloud matching and segmentation** sub-component that matches the as-built data with the as-designed elements and stores them into objects to be analysed by the *Quality control check* (described below);
  - the **QC task list generator** that creates a list of QC checks that are needed for each element using the as-planned elements that are already built and the information extracted by the BIM file pre-processing;
  - the **Quality control check** that carries out the geometric QC using the segmented as-built data, the as-planned data, and the QC work order generator info;
  - the **QC report generator** that generates the QC reports that are going to be delivered to the DTP, DCC, and the client in an appropriate format.

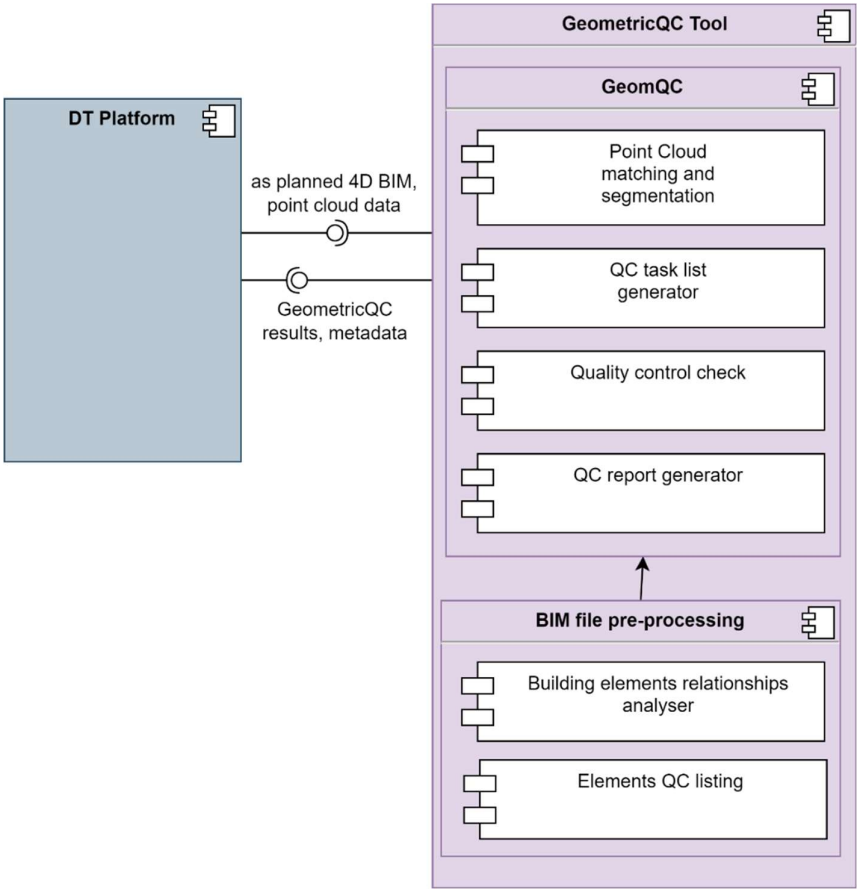


Figure 23 – Component diagram of the GeometricQC tool

5.1.10 Visual Quality Control – VisualQC

Figure 24 illustrates the UML component diagram of the VisualQC tool. As shown below, the tool is composed of:

- the **communication layer** responsible enabling the communication with the DT Platform to allow VisualQC to receive notification when new visual data are available an query the required data from the DT Platform;

- the **Structural type-based selector** that identifies the type of the structure (concrete or steel) and selects the appropriate trained model for defect detection;
- the **Defect detector** responsible for detecting defects with the use of deep learning algorithms applied on processed visual data; and
- the **QC report generator** that generates the QC reports that are going to be delivered to the DT Platform.

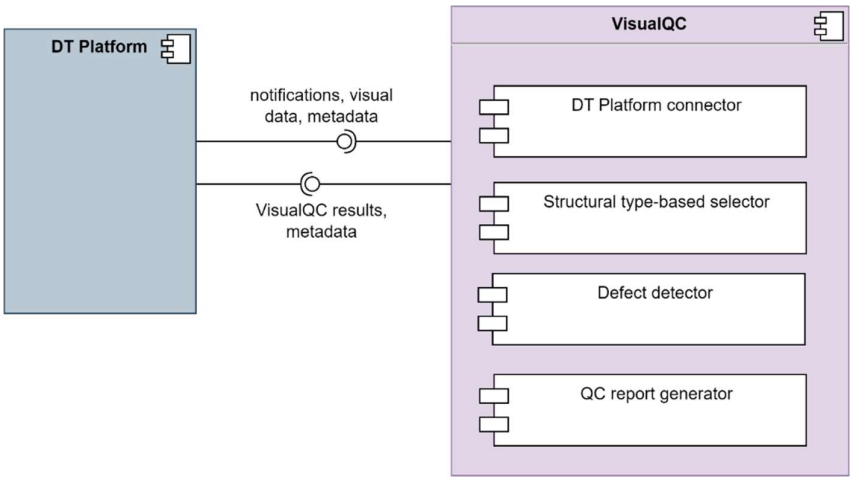


Figure 24 – Component diagram of the VisualQC tool

5.1.11 SafeConAI

Figure 25 illustrates the UML component diagram of the SafeConAI tool. As shown below, the tool is composed of:

- the **Rule-based safety analyser** responsible for analysing the imported 4D as-planned model against a set of predefined rules inserted / selected by the user i.e. HSE Manager / HSE Supervisor;
- the **Hazard zone identification** sub-component that identifies regions where specific types of hazards are (so-called hazard zones); and
- the **Mitigation measures generator** that generates and "injects" mitigation measures into the model.

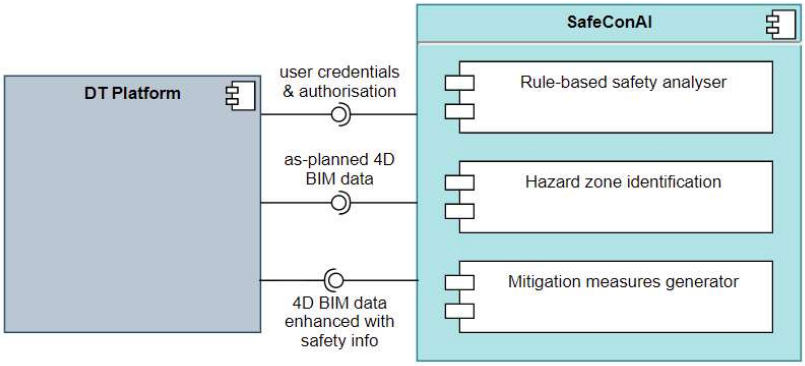


Figure 25 – Component diagram of the SafeConAI tool

5.1.12 ProActiveSafety

Figure 26 illustrates the UML component diagram of the ProActiveSafety tool. As shown below, the tool is composed of:

- the **Data analyser** responsible for analysing the location tracking data of the resources coming from the IoT solution deployed on site;
- the **Trajectory predictor** that performs short-term location prediction based on location tracking data;
- the **Hazard zones checker** that cross-checks: (i) the paths estimated by the trajectory predictor with (ii) the identified hazards extracted by the SafeConAI and stored in the DT platform as a safety-enhanced 4D BIM model; and
- the **Risk analyser** that assesses the probability of hazards to result on an accident and issues relevant warnings (alarms) that are sent as notifications to the concerned workers through WOEA.

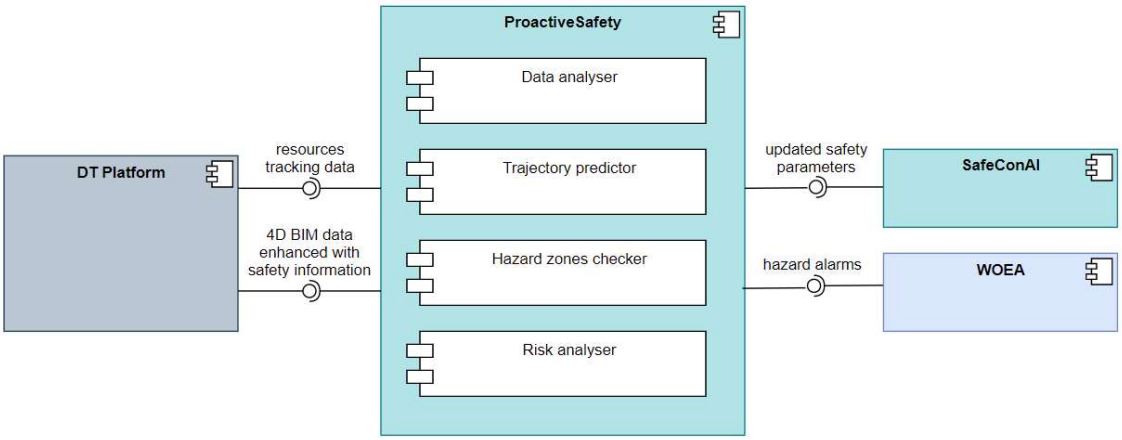


Figure 26 – Component diagram of the ProactiveSafety tool

5.1.13 VirtualSafety

Figure 27 illustrates the UML component diagram of the VirtualSafety tool. As shown below, the tool is composed of:

- the **Game generator** responsible for creating a game based on the selected scenario by the user (i.e., HSE trainer) and the updated input from the DT platform as regards hazards' type, construction site information, etc.;
- the **Data collector and analyser** that collects, processes and analyses the in-game performance data (data gathered throughout the game experience provided to the worker); and
- the **Personalized feedback generator** that constructs a personalised feedback and shares it with the game participant (i.e., worker) and the HSE trainer for further evaluation and feedback.

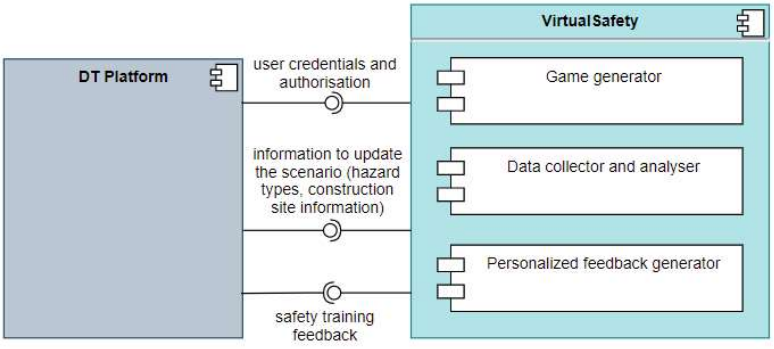


Figure 27 – Component diagram of the VirtualSafety tool

5.1.14 Digital Command Centre – DCC

Figure 28 illustrates the UML component diagram of the DCC tool. As shown below, the tool is composed of:

- the **DCC backend** that includes:
  - the **DT platform connector** enabling the communication with the DT platform;
  - the **IFC converter** responsible for obtaining the IFC file and converting it to a mesh-type file (OBJ, DAE) with the accompanying metadata (XML);
  - the **HTTP endpoint** enabling the inter-communication of the DCC backend with the DCC unity application; and
- the **DCC Unity application** that includes**:**
  - the **Software Development Kit (SDK)** that is packaged as a unity package, reusable for any Unity-based COGITO project, (i) connects to the DT Platform for receiving near-real-time data, (ii) obtains the mesh and metadata files from the DCC backend and generates a Unity scene, (iii) communicates with the DCC backend and allows the selection of the desired project and the retrieval of the mesh and metadata files;
  - the **real-time data overlay** that uses the SDK and converts and renders the data in the 3D view; it also allows the user to select which data he wants to visualise;
  - the **element browser** that displays a tree-like view of the building and allows the user to select, filter and manipulate the elements. It also shows element details and properties; and
  - the **3D viewer**, a Unity3D scene, that visualises the as-planned and as-built elements as well as the sensor data.
  -

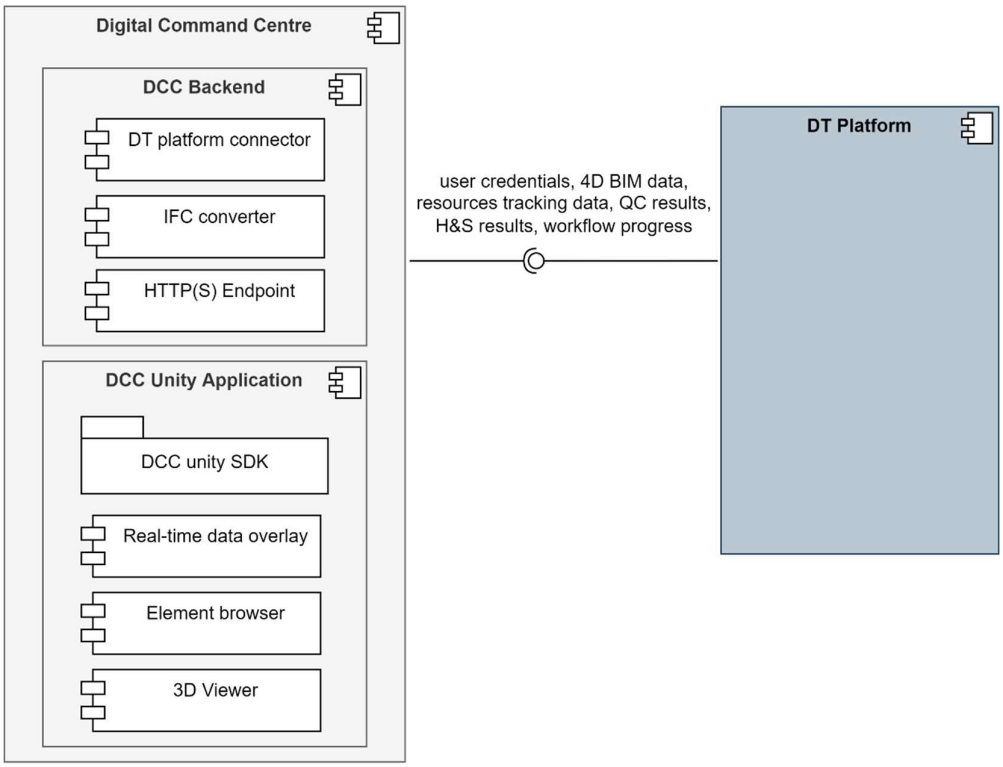


Figure 28 – Component diagram of the DCC

5.1.15 Digital Twin visualisation with Augmented Reality – DigiTAR

Figure 29 illustrates the UML component diagram of the DigiTAR tool. As shown below, the tool is composed of:

- the **DT Platform connector** that enables the communication with the DT platform;
- the **Visual Data Pre-processing connector** that enables the communication with the Visual Data Pre-processing tool;

- the **Mode selector** that upon selection activates one of the three different operation modes of DigiTAR (QC, Safety or Pre-processing);
- the **Data collector** that collects on site raw data that is going to be sent to the Visual Data Pre-processing tool for pre-processing;
- the **Job generator** that generates new jobs including new data that will be sent for pre-processing and defect detection through the DT Platform;
- the **3D IFC viewer**, a Unity3D scene, that visualises the as-planned elements as well as the QC and H/S results;
- the **2D visual data viewer** that visualises the raw and processed visual data; and
- the **Report generator** that generates confirmation reports or additional remedial works that are going to be delivered to the DT Platform.

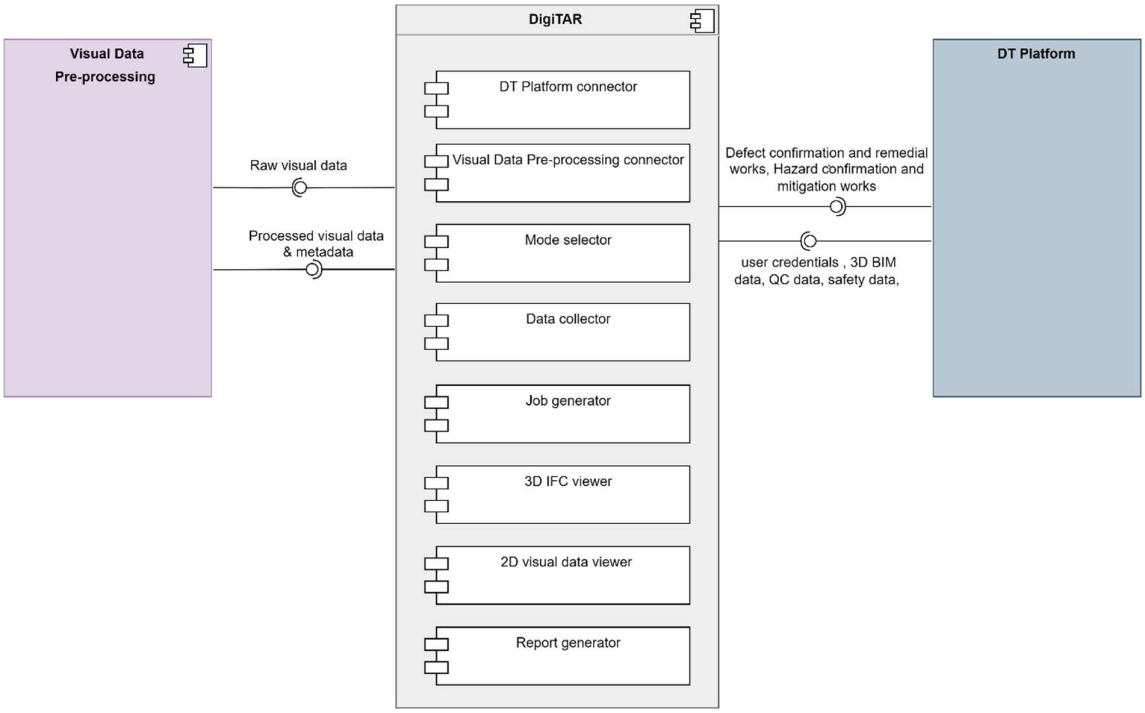


Figure 29 – Component diagram of the DigiTAR tool

5.2 Data Protection

To innovate ethically and responsibly, following the guidelines and best practices of the European Commission [6], COGITO applies the concept of 'privacy by design', utilising a framework where systems, databases and processes are designed and developed in way that safeguards the fundamental rights of data subjects. The broader concept of 'data protection by design', now included in the GDPR, requires data controllers to implement appropriate technical and organisational measures to affect the GDPR's core data-protection principles (articles 5 and 25 GDPR). In COGITO, being in the research and development context, steps to achieve data protection by design will include:

- data minimisation;
- technical and organisation measures description including the usage of data-protection focused service providers and storage platforms;
- arrangements that enable data subjects to exercise their fundamental rights (e.g., as regards direct access to their personal data, consent to its use or transfer, make people aware of any tracking/profiling, etc.).
- pseudonymisation or anonymisation of personal data;

- detailed description of COGITO data handling including, for instance, description of the applied cryptography (e.g., encryption).

In this deliverable, the scope of the section has been to provide the technical measures already defined as part of the COGITO architecture design that can support data protection and privacy throughout the COGITO system. The initial list of technical security measures, extracted from the second version of the "*D1.2 – Data Management Plan*" [3], is shown in Table 16. In the second version of D1.2, the primary and secondary data types that are collected and (or) produced by each component of the COGITO architecture were introduced. Technical measures to ensure data protection and privacy throughout the WPs dealing with the development of the various COGITO's components are currently being collected and processed. The results will be documented in the third version of D1.2, expected to be submitted by the end of M24 (October 2022).

Table 16 – Technical measures to ensure data protection and privacy throughout the COGITO system as extracted from D1.2

Technical Measure Description	COGITO approach
Access control and authentication incl. role-based authorisation	Keycloak is used to allow single sign-in with identity and access management compliant with OAuth 2.0 protocol for authorisation
Logging and monitoring enabling the identification and tracking of user actions (with regard to the processing of personal data);	Supported
Server and database security configured to run using a separate account for COGITO related activities;	Supported
Network/communication security	Whenever access request is performed through the Internet, communication should be encrypted through cryptographic protocols (e.g., TLS/SSL v3, SHA 256 RSA)
Backup and data restore procedures	Supported
Irreversibly deletion of personal data so that it cannot be recovered	Supported
Anonymisation/Pseudonymisation of personal data	Any personal data (requested in the informed consent) are pseudo-anonymised. More details to be reported in the third version of D1.2

6 Conclusions

The aim of this deliverable has been the documentation of the work carried out within "*T2.4 – COGITO System Architecture Design*" that sets the foundation for the design and development of all COGITO components, and mainly the integration among them, taking into account the outcomes of all WP2 tasks activities and predominately the results of "*T2.1 – Elicitation of Stakeholder Requirements*".

In alignment with T2.4 work-planning, it has reported on the final version of the overall architecture of the COGITO solution, including specifications of its key components and their functionalities, an overview of the system architecture describing the components and introducing the various sub-components, and their functional and technical specifications. A revision of the conceptual architecture diagram has initially been performed concluding to an overview of the system architecture, describing the different components, and drafting the information flow among them. Additionally, the data exchange requirements among the COGITO components, per use case, has been represented in sequence diagrams, followed by the detailed design of each COGITO component. To gather information about the functional and non-functional, software and hardware requirements, dependencies to external systems, programming languages, and interactions among the COGITO components, a table template has been used. Finally component diagrams have been designed, illustrating a high-level decomposition of each component to its sub-components. For the development of different types of UML diagrams, the diagrams.net software has been used.

References

[1] COGITO, “Deliverable D2.1 - Stakeholder requirements for the COGITO system,” 2021.

[2] COGITO, “Deliverable D10.2 - POPD Requirements No. 2,” 2021.

[3] COGITO, “Deliverable D1.2 - Data Management Plan,” 2021.

[4] COGITO, “Deliverable D3.1 - Survey of Existing Models & Ontologies & Associated Standardization Efforts,” 2021.

[5] COGITO, “Description of Action (DoA),” 2021.

[6] European Commision, “Ethics and data protection,” 2018.

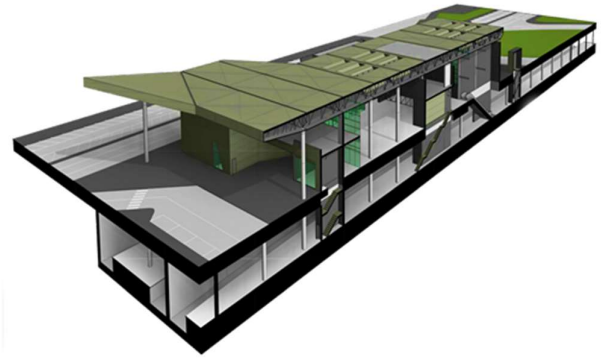
[7] B. Unhelkar, Software Engineering with UML, Auerbach Publications; CRC, , 2018.

[8] COGITO, “Deliverable D10.1 - H-Requirements No.1,” 2021.

Annex A – Component Functional, Non-Functional Requirements and Interfaces Template

Table 17 – <Component Name>: Functional, Non-Functional Requirements and Interfaces

General Information				
Short Description				
Programming Language(s)				
Hardware Requirements				
Software Requirements				
Development Status				
Function and Non-Functional Requirements				
Functional	Req-1.1			
	Req-1.2			
	Req-1.3			
	Req-1.4			
Non-Functional	Req-2.1			
	Req-2.2			
	Req-2.3			
Component Dependencies				
Internal Dependencies	Dep-1.1			
	Dep-1.2			
	Dep-1.3			
External Dependencies	Dep-2.1			
	Dep-2.2			
	Dep-2.3			
Interfaces				
Input Data	Input-1	Received from:	Format	
			Method	
			Endpoint	
			Protocol	
	Input-2	Received from:	Format	
			Method	
			Endpoint	
			Protocol	
	Input-3	Received from:	Format	
			Method	
			Endpoint	
			Protocol	
Output Data	Output-1	Sent to:	Format	
			Method	
			Endpoint	
			Protocol	
	Output-2	Sent to:	Format	
			Method	
			Endpoint	
			Protocol	
	Output-3	Sent to:	Format	
			Method	
			Endpoint	
			Protocol	



COGITO

CONSTRUCTION PHASE DIGITAL TWIN MODEL

cogito-project.eu

HYPERTECH energy labs

UCL

AARHUS UNIVERSITET

THE UNIVERSITY of EDINBURGH

BoC www.boc-group.com

iti

POLITÉCNICA

QUE TECHNOLOGIES

Novitech NEW INFORMATION TECHNOLOGIES

ASM

ferrovial construction

ΟΛΥΜΠΙΑ ΟΔΟΣ

RHOMBERG SERSA RAIL GROUP

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958310