RECLAIM
Refurbishment and re-manufacturing of large industrial equipment

The RECLAIM architecture specification - Period 1

July 2020 – M10

AUTHORS: THANASIS VAFEIADIS, NIKOLAOS KOLOKAS, ANGELIKI ZACHARAKI, KONSTANTINOS GEORGIADIS

DATE: 27.07.2020

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 869884

Technical References

Project Acronym	RECLAIM
Project Title	RE-manufaCturing and Refurbishment LArge Industrial equipMent
Project Coordinator	HARMS & WENDE GMBH & CO KG
Project Duration	01/10/2019 – 31/03/2023

Deliverable No.	2.3
Dissemination level [1]	PU
Work Package	2
Task	2.3
Lead beneficiary	CERTH
Contributing beneficiary(ies)	HWH, LINKS, SUPSI, FEUP, TECNALIA, ADV, FINT, FCY, SCM, ICE
Due date of deliverable	31/07/2020
Actual submission date	-

[1] PU = Public

PP = Restricted to other programme participants (including the Commission Services)

RE = Restricted to a group specified by the consortium (including the Commission Services)

CO = Confidential, only for members of the consortium (including the Commission Services)

Document history

V	Date	Beneficiary	Author
1	21/01/20	CERTH	Thanasis Vafeiadis, Angeliki Zacharaki
2	04/06/20	CERTH	Nikolaos Kolokas
3	15/07/20	CERTH	Nikolaos Kolokas, Konstantinos Georgiadis
4	27/07/20	CERTH	Nikolaos Kolokas

Acronyms

Acronym	Explanation
2D	two-dimensional
3D	three-dimensional
6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks
AC	Alternative Current
AD	Architectural Description
AES	Advanced Encryption Standard
AI	Artificial Intelligence
AMQP	Advanced Message Queuing Protocol
API	Application Programmer Interface
AR	Augmented Reality
ARM	Advanced RISC Machine
ASN	Adaptive Sensorial Network
CBS	Cost Breakdown Structure
CD	Continuous Deployment
CI	Continuous Integration
CIM	Common Information Model
CMOS	Complementary metal-oxide-semiconductor

CSV	Comma-separated values
CUDA	Compute Unified Device Architecture
D	Deliverable
DB [db]	Database
DDD	libraries for the digital twin development and process plant simulation of complex environments (provided by TTS)
DRy	Distributed data storage and analytics
DSF	Decision Support Framework
ETL	extract/transform/load
EU	European Union
FPGA	Field Programmable Gate Array
FWM	Friction Welding Machine
GPU	Graphics processing unit
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
GW	Gateway
HMD	Head-mounted display
HMI	Human-machine interface
HW	Hardware
I(/)O	Input/Output

ID	Identity
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IMU	Inertial Measurement Unit
IoT	Internet of Things
IP	Internet Protocol
ISO	International Organization for Standardization
IT	Information Technology
JPA	Java Persistence API
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
LCA	Life Cycle Assessment
LCC	Life Cycle Cost
LoRa	Long Range
LoRaWAN	Long Range Wide Area Network
M	Month
ML	Machine learning
MQTT	Message Queuing Telemetry Transport
MRL	Mean Residual Life
MTBF	Mean Time Between Failures

(M)TTF	(Mean) Time To Failure
MTTR	Mean Time To Repair
NoSQL	non-SQL
OGC	Open Geospatial Consortium
OPC-UA	Open Platform Communications Unified Architecture
OS	Operating system
PC	Personal Computer
PHM	Prognostic & Health Management
QC	quality check
R&D	Research & Development
RAM	Random Access Memory
REST	Representational state transfer
RGB-D	red-green-blue-depth
RISC	Reduced instruction set computer
RUL	Remaining Useful Life
S(/)W	Software
SCM[1]	Source Code Management
SOAP	Simple Object Access Protocol
SQL	Structured Query Language

[1] This is also the abbreviation of the RECLAIM beneficiary “SCM Group”.

T	Task
TBA	To Be Announced
TCP	Transmission Control Protocol
TTL	Transistor-transistor logic
TXT	filename extension for text files
UC	Use Case
UCD	user-centred design
UDP	User Datagram Protocol
UI	User Interface
UML	Unified Modelling Language
URI	Uniform Resource Identifier
USB	Universal Serial Bus
WIP	Work In Process
WP	Work Package
WSN	Wireless sensor network
XLSX	filename extension for Excel files
XML	eXtensible Markup Language

Summary

This deliverable mainly describes the architecture of the RECLAIM platform. It also presents the approach and methodology that have been followed by T2.3 to define this architecture. There will be two revisions of this deliverable in M20 (5/2021) and M30 (3/2022), including also stakeholders that may be relevant to this project and the platform.

Disclaimer

Table of Contents

1. Introduction

This section describes the purpose, background and structure of this deliverable, the related terminology, as well as its relation with other tasks and deliverables.

1.1 Purpose, context and scope

In this deliverable the architecture for the RECLAIM platform is defined, based on the technologies brought to the project and the user needs. Detailed descriptions of the architectural elements, e.g. the cybersecurity framework, the Decision Support Framework or RECLAIM Repository, will be available as separate deliverables as outlined in the project specification. For implementation details and specifications, the reader should refer to these. This deliverable will focus on the fundamental concepts and properties of the RECLAIM system. The architectural description includes aspects pertinent to the identification of the major system components, their appropriate interactions and definitions of their external interfaces.

Various key functional requirements and architectural constraints had been defined in the project specification before the beginning of the project. The architecture was defined while gathering and validating requirements and definition of pilot scenarios and use cases. Also, within the architecture task (T2.3), the detailed system requirements were defined.

WP2 concerns Requirements Engineering and the Reference Architecture and deals with the multiple phases of an evolutionary requirements engineering process, that will include engineering and refinement of user, operational and system requirements, specification and refinement of architecture design, as well as model development and descriptions. An iterative approach is followed, i.e. the contents of this deliverable are continuously updated during the project.

1.2 Background

The vision of RECLAIM is to demonstrate technologies and strategies in support of a new paradigm for the management of large industrial equipment that approaches the end of its designed life. This paradigm will substantially reduce the opportunity cost of retain strategies (both money-wise and resource-wise) by allowing relatively old equipment that faces the prospect of decommissioning to reclaim its functionalities and role in the overall production system. Such new strategies will contribute to a more sustainable and resource-friendly asset management and, at the same time, offer economic and competitive advantages to the manufacturing sector. To achieve the above, a Decision Support Framework (DSF) will be developed to accumulate knowledge of the health status of machinery and propose innovative methods, tools or services for the appropriate lifetime extension strategy:

- *Refurbishment and Upgrade of industrial equipment* to improve machinery operation and avoid unplanned downtime due to machine failure.
- *Re-manufacturing and Re-use of industrial equipment* to estimate Lifecycle cost and contribute to the re-use of old machinery assets in renewed and new factories.
- *In-situ Repair* to minimise the extra cost and downtime associated with the disassembly and transportation of the machinery.
- *Predictive Maintenance and Fault Diagnosis* to maximize the performance of machinery during its lifetime and provide pragmatic maintenance able to identify equipment failures before they occur.

The RECLAIM Solution with its planned activities addresses currently neglected industrial needs and contributes to unleashing the full potential of sustainable, green, and smart factories, by empowering the industry to produce components and assembly systems that meet fast changing requirements. RECLAIM focuses on 100% re-use of equipment through flexible and low-cost systems that support the fast and easy process of refurbishment and re-manufacturing. This perspective will develop self-aware and knowledge-based equipment for the collection and management of operation-related information. All the above-mentioned solutions will be demonstrated in real industrial environments to evaluate the lifecycle of the industrial equipment (machines, production lines, robotic system, etc.) and to implement the appropriate recovery strategies (refurbishment, re-manufacturing, upgrade, re-use, repair, etc.).

The platform will be demonstrated in 5 different pilots belonging to the five RECLAIM's end users, where alternative industrial machines will be refurbished and/or re-manufactured:

- The scenario of GORENJE (white goods manufacturer) includes robots' cells and white enamelling line.
- The scenario of FLUCHOS (footwear manufacturer) comprises cutting machines.
- The scenario of PODIUM (wood manufacturer) contains machines for cutting, drilling, and finishing.
- The scenario of HWH (producer of control systems in the welding sector) comprises friction welding machines.
- The scenario of ZORLUTEKS (textile manufacturer) includes bleaching machines.

1.3 Relation to other tasks and deliverables

The are also other deliverables which are more dedicated to requirements and use cases: (D2.1, 7/2020), (D2.2, 7/2020), (D2.6, 3/2021), (D2.4, 1/2022), (D2.7, 3/2022). The architecture definition process has been progressively taken direct or indirect inputs from them, following their submission deadlines.

1.4 Content and structure of this deliverable

The remainder of this deliverable is organized as follows: Section 2 describes the methodology for creating an architecture description, whereas section 3, which is the core of the deliverable, describes the architecture of the RECLAIM software platform, applying the methods of section 2. There are also important annexes in the end of the deliverable. Annex 1 lists the system requirements which are satisfied by the software components of the architecture, Annex 2 contains the available specifications of hardware components, whereas Annex 3 includes the individual data models defined by software component owners so far.

1.5 Terminology

The currently adopted domain-specific terminology used in the remainder of the document is presented in Table 1 below.

Table 1 - RECLAIM-specific terminology

Term	Definition
Adaptive Sensorial Network & Digital Retrofitting Infrastructure	the part of the architecture which is responsible for near-real-time data delivery from the machinery data collectors (e.g. sensors, actuators, controllers) to RECLAIM repository for the health status monitoring of the machines
AR Mechanisms	multimodal interaction system which provides technicians with an augmented reality view of several streams of data, animated 3D stepwise instructions on disassembly and reassembly required, as well as support in the form of on-the-job remote assistance with real-time audio-visual communication and 3D annotation to technicians during the procedure
Cost Modelling and Financial Analysis Toolkit	cost estimation tool for cost and financial impact analysis, which contributes to the Optimization Toolkit
Decision Support Framwork	the technological core of RECLAIM, that guides the optimal refurbishment and re-manufacturing of electromechanical machines and robotics systems
Fault Diagnosis and Predictive Maintenance Simulation Engine using Digital Twin	component developing models to be used for monitoring and predicting the performance and status of factory assets, using the data collected by the Adaptive Sensorial Network & Digital Retrofitting Infrastructure, machinery profiles and expert feedback
In-situ Repair Data Analytics	the toolkit that the project deploys to raise awareness of the health status of the machine and situational of the shop

	floor during maintenance activities
Optimization Toolkit for Refurbishment & Re-manufacturing Planning	component which aims to support the optimization planning through multi-variable monitoring of the machine's operational parameters where the effects of variable changes will be possible to determine and combine known best practices methodologies for model-based plant-site / shop-floor control
Prognostic and Health Management (PHM) Toolkit	comprehensive framework for predictive and preventive control and management
RECLAIM Repository	the component of RECLAIM architecture which is responsible for storing raw data, pre-processed user input, trained models, predictions and evaluation metrics of algorithms, and delivering them to other components whenever needed
Refurbishment and Re-manufacturing Framework	the framework which processes the refurbishment / remanufacturing steps and delivers the optimal steps to the AR mechanisms

2. Architectural Design Methodology

This section presents the key concepts related to the methodology used to develop the architectural design of the software system developed in RECLAIM.

Standards and best practices have been followed, as described in the next subsections. In addition, there have been several remote technical teleconferences to discuss, produce and refine the architecture design.

The duration of T2.3 is from M4 (1/2020) to M30 (3/2022), but it is split in three iterations. These are M4-M9 (1-6/2020), M14-M20 (11/2020-5/2021) and M24-M30 (9/2021-3/2022). The first iteration defines the first version of the architecture, whereas the other two iterations represent further improvements as feedback from the technical work packages is being collected and processed. The first version comprises the first consolidation of dependencies, inputs/outputs, and specifications of architectural components. The purpose of the second version is to present detailed information about the interfaces between the components, while the last version is intended to describe in detail the whole platform in terms of architecture, modules, dataflow, processes, APIs specifications, etc.

The process followed is based on (1471-2000 - IEEE Recommended Practice for Architectural Description for Software-Intensive Systems, 2000) and (ISO/IEC/IEEE 42010: Systems and Software Engineering - Architecture Description, 2011), by which the former was superseded.

An architectural viewpoint is a collection of patterns, templates, and conventions for constructing one type of architectural view. An example is the functional viewpoint (and therefore a functional view), which contains all functions that should be performed by the system, the responsibilities and interfaces of the functional elements and the relation between them. These functions can be described in a standardized way using UML diagrams. Moreover, it also describes which stakeholders need to be involved and how their needs should be applied in the architecture as stated in the "architectural perspectives" chapter by Rozanski and Woods (N. Rozanski, 2012).

Based on the specification of the ISO/IEC/IEEE 42010:2011 standard, the main concepts of architecture view and architecture viewpoint are formally defined as follows:

- *Architecture viewpoint:* Work product establishing the conventions for the construction, interpretation and use of architecture views to frame specific system concerns
- *Architecture view:* A representation of a whole system from the perspective of a related set of concerns

The initial user, system and operational requirements from T2.1 and T2.2, which were obtained from the T2.1 survey or the communications with the pilots within the context of T2.2, were the input for the first architecture design phase where a first draft of the architectural description was created. Based on this architectural description, the first prototype is going to be created, which can be seen as a skeleton system with minimal functionality developed above that. These development efforts will reveal some experiences and lessons learnt which in turn will constitute a valuable source for the derivation of additional requirements and the revision of already existing ones, within the context of T2.4.

The RECLAIM project decided on the following viewpoints from which the architectural document views are derived:

- *Context viewpoint:* This describes interactions, relationships, and dependencies between the system of interest and its environment. The environment includes the external entities with which the system interacts, e.g. other systems, users, or developers.

- *Functional viewpoint:* This describes the functional elements needed to meet the key architectural requirements. It will present proposals in a descriptive way and UML diagrams will facilitate the understanding of the proposal. It depicts responsibilities, interfaces, and interactions between the functional elements.

- *Information viewpoint:* This describes the data models and the data flow, as well as the distribution. The viewpoint also specifies which data will be stored and where. Describing where data will be manipulated is also a responsibility of this viewpoint.

- *Deployment viewpoint:* This describes how and where the deployment of the system will take place and what dependencies exist, considering for example hardware requirements and physical limitations. If there are technology compatibility issues, they can be addressed in this viewpoint too.

- *Development viewpoint:* This viewpoint addresses concerns from the developers' point of view. It describes the way in which the software development process is supported, for example what conventions should be followed and what the artefact management will look like.

In order to address quality properties and cross-cutting concerns, architectural perspectives will be used. A typical example, which is a particular concern for RECLAIM, is security: the way the data are secured, the communication channels in which data will be transacted, and the functional elements needing to be protected should be considered. Another perspective which is interesting is availability of the hardware, the functional elements, the data, etc. There are also relevant standards about safety and satisfaction, being these described in D2.2. Furthermore, the performance perspective has been taken into account, in terms of latency mitigation in data delivery between the Adaptive Sensorial Network or the Users Interfaces on the one side and the RECLAIM platform on the other, therefore in some cases dedicated components have been envisioned for more real-time data communication than in case that these data are transferred through the RECLAIM Repository. Finally, since numerous partners will be using/developing different technologies, also the compatibility, which may be the most challeging perspective, should be taken into account.

The detailed procedure followed during the course of T2.3 is described below.

2.1 First iteration

The first iteration consisted of a bottom-up phase, within which the architecture was defined based on the technologies (mainly software, but also hardware components) that relevant partners stated that they can bring to the project based on their previous experience and their skills. The top-down phase, which is based on the pilot needs, will be elaborated to a high extent in the next iteration, when there will be more time for T2.3 participants to consider the pilot-oriented requirements and use cases coming from T2.1 and T2.2. Particularly, regarding T2.2, during the 2nd iteration the UML Use Case diagrams obtained from that task will be instantiated as UML sequence diagrams.

More especially, in the first iteration, the discussions on the definition of the architecture started based on the conceptual architecture of the Description of Action, which is shown in Figure 1, along with the descriptions of the related tasks. During the phase of this process, the following steps were followed:

1. Owners of architectural components filled in templates to describe them, along with system requirements satisfied by them. Presentations were also created for the different components as a supplementary material.
2. A series of teleconferences took place so that the interdependencies of components and Tasks are clearly specified, and based on the conclusions from these discussions the consolidated view (Figure 2) was designed. This regards a high-level informal figure depicting the communications among services, frameworks, repositories, and hardware. In parallel, the high-level functional view and the information view UML diagrams were created under several refinements. The consolidated view started to be instantiated for particular pilots in the end of the first iteration, but this instantiation is going to be shown in the 2nd version of the architecture, because it has not been completed sufficiently yet.
3. In order to define in detail the functional view, multiple UML component diagrams were created, focusing on specific components and communications with others.
4. An important component that was defined is the Data Handler of the RECLAIM Repository, which is responsible to exchange data between the Repository database and other components. Because the data need transformations so as to be communicated, the data format that each involved component accepts was defined, and afterwards a holistic data model (common information model) was created so that these transformations are made based on it.
5. The initial ideas for the development view were documented, based on previous experience of the technical partners.
6. A high-level UML deployment view diagram was created after collaboration among hardware and software providers.

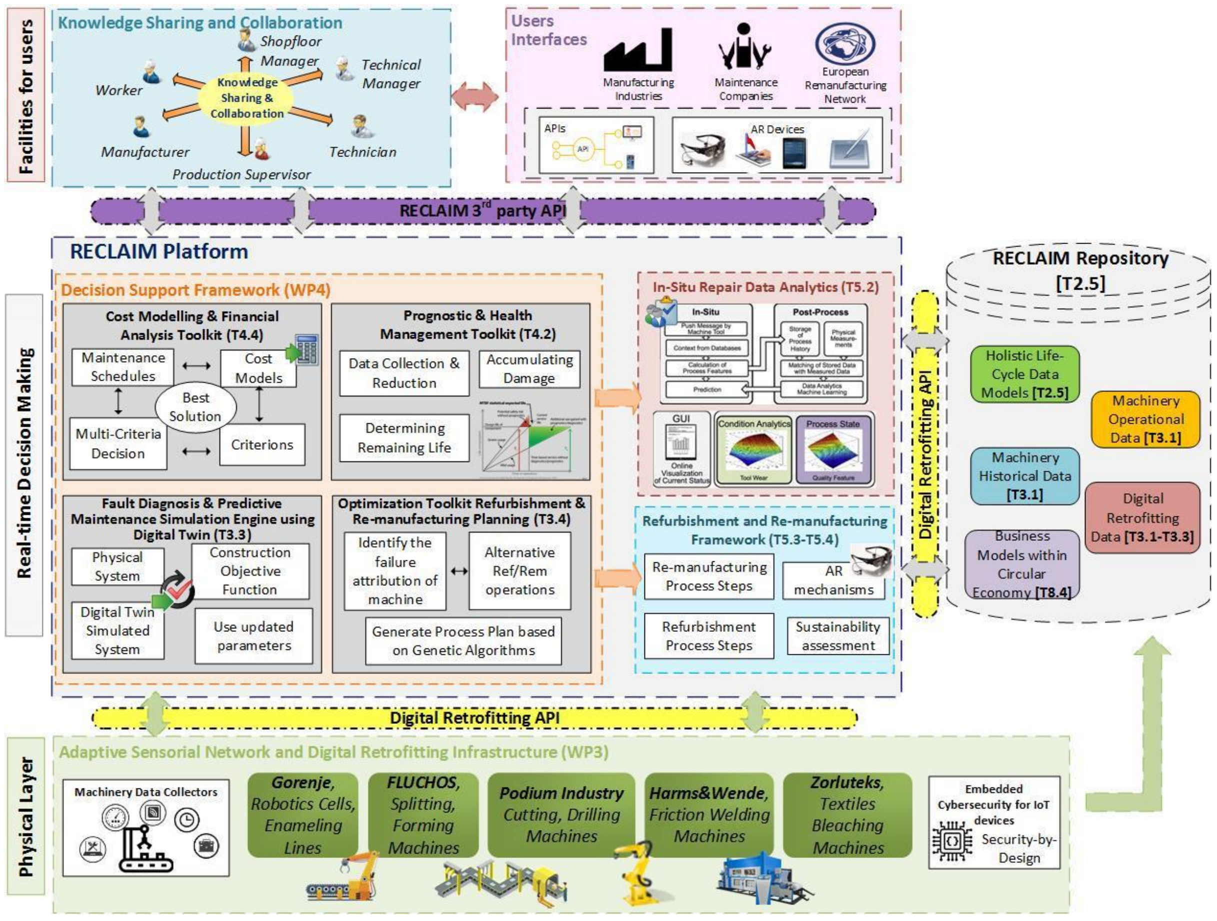

Facilities for users
Knowledge Sharing and Collaboration
Shopfloor Manager
Technical Manager
Worker
Knowledge Sharing & Collaboration
Manufacturer
Technician
Production Supervisor
Users Interfaces
Manufacturing Industries
Maintenance Companies
European Remanufacturing Network
APIs
AR Devices
RECLAIM 3rd party API
RECLAIM Platform
Real-time Decision Making
Decision Support Framework (WP4)
Cost Modelling & Financial Analysis Toolkit (T4.4)
Maintenance Schedules
Cost Models
Best Solution
Multi-Criteria Decision
Criterions
Prognostic & Health Management Toolkit (T4.2)
Data Collection & Reduction
Accumulating Damage
Determining Remaining Life
Fault Diagnosis & Predictive Maintenance Simulation Engine using Digital Twin (T3.3)
Physical System
Construction Objective Function
Digital Twin Simulated System
Use updated parameters
Optimization Toolkit Refurbishment & Re-manufacturing Planning (T3.4)
Identify the failure attribution of machine
Alternative Ref/Rem operations
Generate Process Plan based on Genetic Algorithms
In-Situ Repair Data Analytics (T5.2)
In-Situ
Post-Process
GUI
Online Visualization of Current Status
Condition Analytics
Process State
Tool Wear
Quality Feature
Refurbishment and Re-manufacturing Framework (T5.3-T5.4)
Re-manufacturing Process Steps
AR mechanisms
Refurbishment Process Steps
Sustainability assessment
Digital Retrofitting API
RECLAIM Repository [T2.5]
Holistic Life-Cycle Data Models [T2.5]
Machinery Operational Data [T3.1]
Machinery Historical Data [T3.1]
Digital Retrofitting Data [T3.1-T3.3]
Business Models within Circular Economy [T8.4]
Digital Retrofitting API
Physical Layer
Adaptive Sensorial Network and Digital Retrofitting Infrastructure (WP3)
Machinery Data Collectors
Gorenje, Robotics Cells, Enameling Lines
FLUCHOS, Splitting, Forming Machines
Podium Industry Cutting, Drilling Machines
Harms&Wende, Friction Welding Machines
Zorluteks, Textiles Bleaching Machines
Embedded Cybersecurity for IoT devices
Security-by-Design

Figure 1: Conceptual technical architecture of RECLAIM as presented in the Grant Agreement

The consolidated view below shows that every pilot may use the RECLAIM Repository and/or their own Repository for data storage, the RECLAIM dashboards and services/frameworks or also their own services, including dashboards, if they already exist. In order that the project's achievements are demonstrated, every pilot has to benefit at a high percentage from the project's achievements. There will be two main ways for data communication between components: the RECLAIM Repository bus for communication through the RECLAIM Repository under cybersecurity mechanisms, and the RECLAIM Service bus for more direct communication (of relatively low amount of data) without intervention of the repository. The consolidated view is described more extensively in D2.2.

RECLAIM
Refurbishment and re-manufacturing
of large industrial equipment

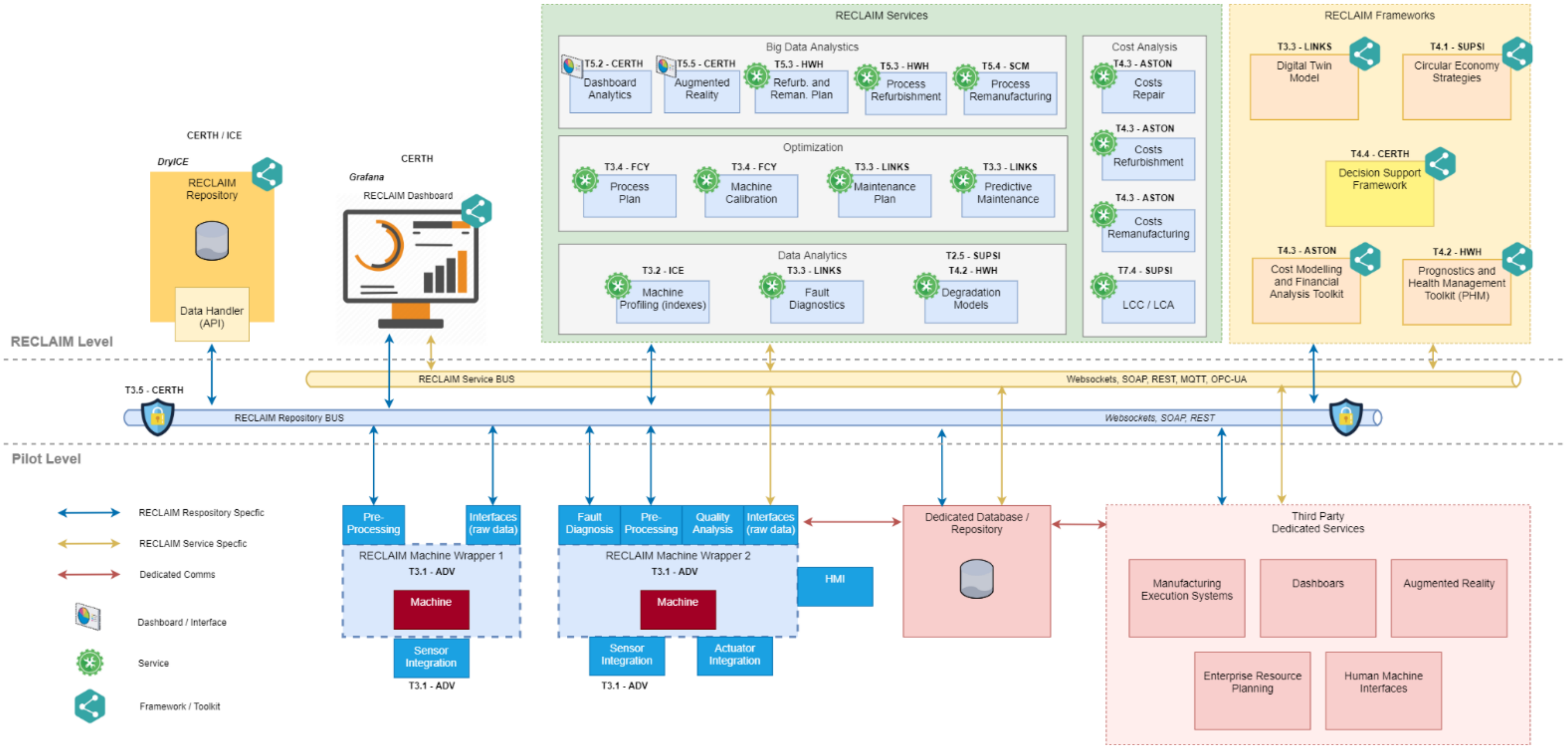


Figure 2: Consolidated view of RECLAIM

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°869884

The idea was to instantiate this view for every pilot, and this is envisioned for the 2nd iteration.

3. Architectural views

This is the core section of the deliverable and describes the architectural views corresponding to the viewpoints mentioned in Section 2.

3.1 Functional and information view

As mentioned above, the first step of the bottom-up process during the first iteration of T2.3 was the definition of architectural components and system requirements satisfied by them. These were defined by the technical partners responsible for bringing these components to RECLAIM. On the other hand, requirements based on end user needs and questionnaires and specifications for the RECLAIM framework are included in D2.1. These requirements are based on the user scenarios and are used to evaluate the platform. Also, the purpose of D2.2, D2.6 and D2.7 is to present in detail the use cases and operational (field-related) requirements in three phases, as an outcome of T2.2 activities. Based on the components and requirements, the functional and information views were defined.

Functional requirements specify the product's functionality, derived from its fundamental purpose. They frame the solution space for the problem that is being addressed by describing the way in which the system should behave in particular situations: its inputs, its outputs and actions it needs to perform so as to accomplish its fundamental purpose. Sometimes, functional requirements also define explicitly what the system should not do or detail the needed exceptions [see: (Sommerville, 2011)]. Although no established subcategories exist in order to further sort functional requirements, the next lines show high level groups aiming at illustrating some examples:

- Descriptions of the product’s desired functionality
- Explanation of the services that it should provide
- Details of the processing actions or operations which it must take
- Description of the way in which the system should react to particular inputs

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 869884

In contrast, non-functional requirements are the product's properties, the qualities and characteristics which make the product attractive, usable, fast, or reliable. They are not concerned directly with the specific operations and services that the product should perform or deliver, but they are related to emergent properties like modifiability, scalability, and interoperability. At the same time, non-functional requirements can reflect constrains or restrictions of the system. They are just as relevant and critical as functional requirements for ensuring the solution success.

Non-functional requirements can be grouped based on following subcategories:

- Look and feel requirements (desired appearance for end users)
- Usability requirements (according to the intended end users and the context of use)
- Performance requirements (speed, size, accuracy, safety, reliability, etc.)
- Operational requirements (desired operating environment)
- Maintainability and portability requirements (how adaptable it must be)
- Security requirements (security, integrity, and confidentiality)
- Cultural and political requirements (human factors)
- Legal requirements (compliance with applicable laws)

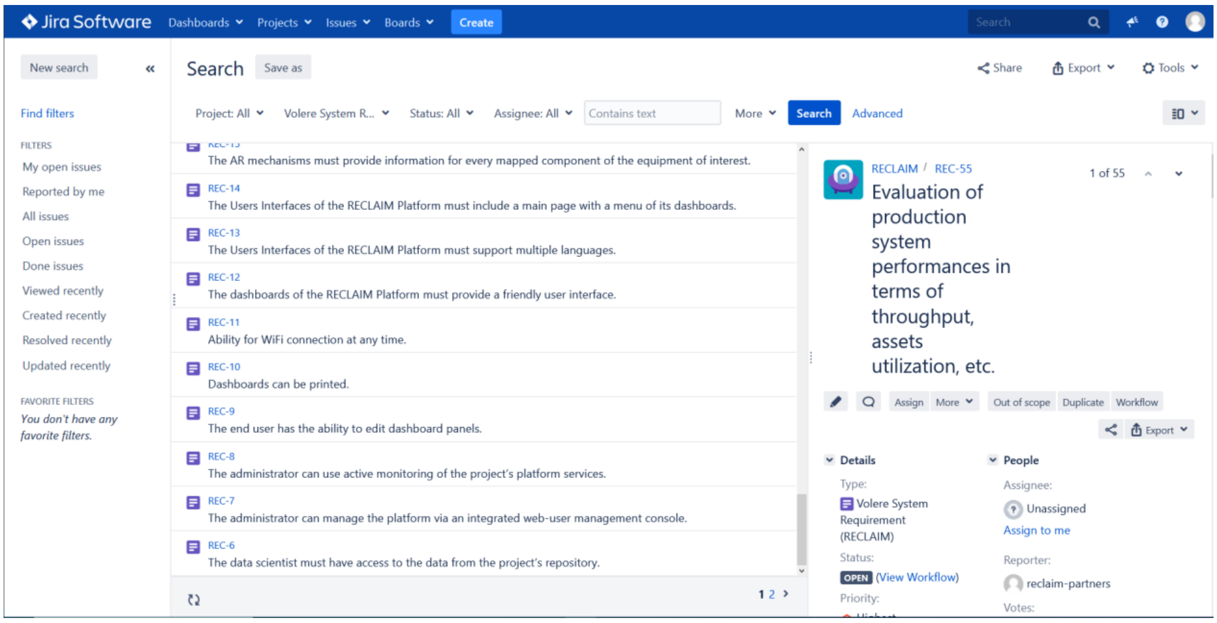
Look and feel, usability and cultural requirements are not as relevant as the others for the assessment of requirements for a software platform, but they are highly important for the assessment of qualities and aspects of the user interfaces to be developed. The current set of requirements can be found in the online workspaces of the project and thus has become accessible for all users and also traceable for evaluation of design solutions, while any updates are made known in real time to the consortium.

3.1.1 System requirement description (Volere-based)

The Volere process recommended by Robertson and Robertson (S. Robertson, 1999) verifies that all important aspects of requirements are addressed carefully and that the methods applied have practically proven their value. It has been proven to be very valuable to put in the effort to define the global constraints that affect the project and the fine-grained distinction of different types of (non-)functional requirements. The philosophy of Robertson and Robertson allows the requirements to be processed in a structured way, assuring that they can always be applied and tested.

The workflow ensuring that all necessary details and procedures in the Volere schema are followed is rather complex, and it was decided to support this process with a tool for all partners within RECLAIM.

It was decided to use JIRA, a web-based bug tracker that allows implementing and tracking the workflow of the Volere schema. Figure 3 contains a screenshot of JIRA showing a part of the list of system requirements.



REC-15
The AR mechanisms must provide information for every mapped component of the equipment of interest.

REC-14
The Users Interfaces of the RECLAIM Platform must include a main page with a menu of its dashboards.

REC-13
The Users Interfaces of the RECLAIM Platform must support multiple languages.

REC-12
The dashboards of the RECLAIM Platform must provide a friendly user interface.

REC-11
Ability for WiFi connection at any time.

REC-10
Dashboards can be printed.

REC-9
The end user has the ability to edit dashboard panels.

REC-8
The administrator can use active monitoring of the project's platform services.

REC-7
The administrator can manage the platform via an integrated web-user management console.

REC-6
The data scientist must have access to the data from the project's repository.

RECLAIM / REC-55
Evaluation of production system performances in terms of throughput, assets utilization, etc.

Details
Type: Volere System Requirement (RECLAIM)
Status: OPEN (View Workflow)
Priority:

People
Assignee: Unassigned
Assign to me
Reporter: reclaim-partners
Votes:

Figure 3: Screenshot of JIRA with a part of the list of requirements

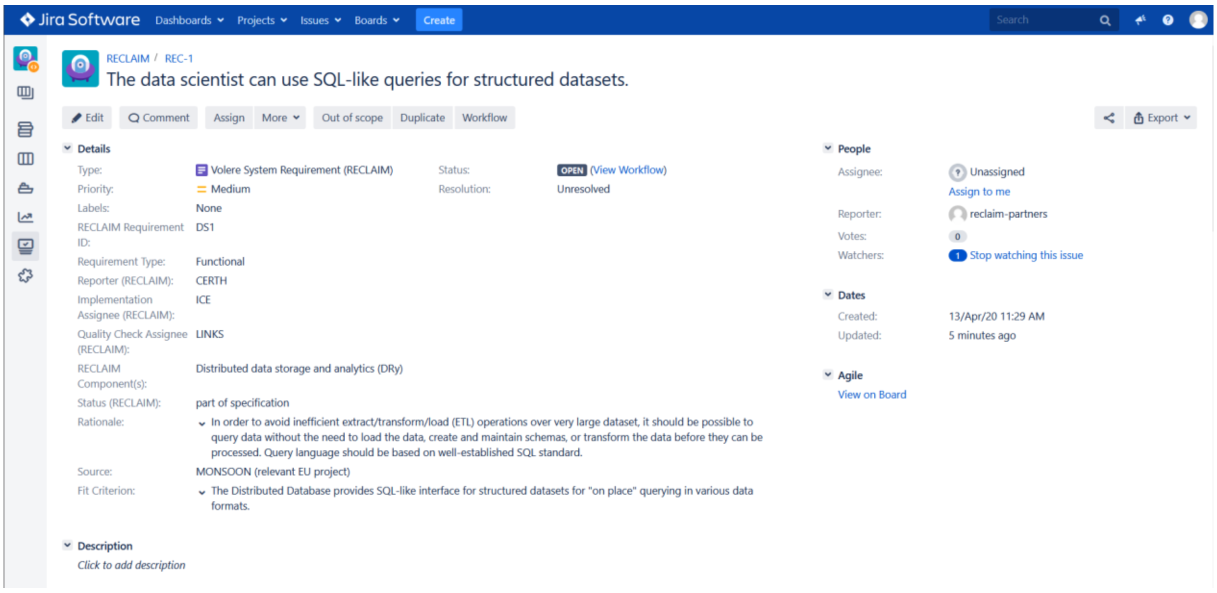
The main fields of a requirement template based on the Volere schema follow:

- The *name* of a requirement describes it in a single sentence. This description tells about the purpose of the requirement and should be clear and brief.
- The *rationale* of a requirement expresses the reason behind its existence, explaining its importance and its contribution to the product's purpose. The rationale facilitates the understanding of the requirement.
- The *Fit Criterion* is the quantified goal that the realization of the requirement has to meet. This field determines when the requirement is met. It should be written in a precise quantifiable way. The Fit Criterion sets the standard to which the product is constructed by the developer.
- *Priority* is an essential field which defines the relevance of this requirement in relation to the other requirements. It allows the specified requirement to be classified in one of 5 categories: "Highest", "High", "Medium", "Low" and "Lowest". The rating was taken into account during the quality check.

The priority of a requirement is based on the next fields in the Volere schema.

- The *source* defines if a requirement was raised by the Description of Action itself, primary or secondary stakeholders, or through discussions among consortium members, by vision and technical scenarios.
- The *component(s)* that the requirement is associated to.
- The *status*, which is an estimation whether the requirement is well-defined and within the scope of the project.

Figure 4 shows a screenshot of JIRA with the details of a particular requirement.



RECLAIM / REC-1

The data scientist can use SQL-like queries for structured datasets.

Details

Type:	Volere System Requirement (RECLAIM)	Status:	OPEN (View Workflow)
Priority:	Medium	Resolution:	Unresolved
Labels:	None		
RECLAIM Requirement ID:	DS1		
Requirement Type:	Functional		
Reporter (RECLAIM):	CERTH		
Implementation Assignee (RECLAIM):	ICE		
Quality Check Assignee (RECLAIM):	LINKS		
RECLAIM Component(s):	Distributed data storage and analytics (DRy)		
Status (RECLAIM):	part of specification		
Rationale:	In order to avoid inefficient extract/transform/load (ETL) operations over very large dataset, it should be possible to query data without the need to load the data, create and maintain schemas, or transform the data before they can be processed. Query language should be based on well-established SQL standard.		
Source:	MONSOON (relevant EU project)		
Fit Criterion:	The Distributed Database provides SQL-like interface for structured datasets for "on place" querying in various data formats.		

Description

Click to add description

People

Assignee:	Unassigned Assign to me
Reporter:	reclaim-partners
Votes:	0
Watchers:	1 Stop watching this issue

Dates

Created:	13/Apr/20 11:29 AM
Updated:	5 minutes ago

Agile

View on Board

Figure 4: Screenshot of JIRA with the details of a particular requirement under the Volere schema

3.1.2 System requirements workflow

Three different user groups are involved in the requirements process:

- Reporter: The organization of the person who creates the requirement.
- Quality check assignee: Each newly reported requirement is assigned to a single organization – the quality check assignee. This organization is responsible for passing the requirement through the quality check.
- Implementation assignee(s): organization(s) responsible to implement a requirement after it has passed a quality check successfully

Figure 5 displays a requirement's possible states and the possible transitions among them.

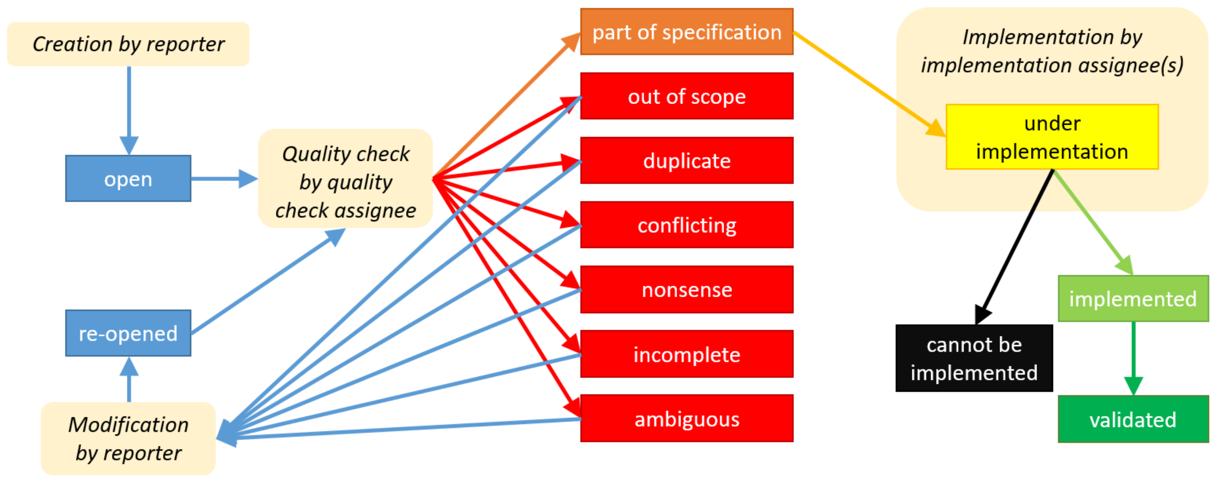


Figure 5: Structure of the Requirements Workflow

When a reporter creates a requirement, it gets assigned the status *open*. This reporter assigns the requirement to a quality check assignee, that changes the requirement's status into *part of specification* if it is complete and unambiguous. The quality check assignee and the reporter belong to different organizations. A requirement that has passed the quality check has its text fields filled in sensibly, with appropriate values chosen from the drop-down lists.

A requirement can fail to pass the quality gateway for the next reasons:

1. It can be incomplete. Some fields may have meaningless entries, e.g. "?".
2. It can be ambiguous; certain terms are not clearly specified.
3. It is too general or does not make sense at all; this can happen for example when the reporter of the requirement does not include sufficiently detailed information in order for others to understand the reasoning behind it.
4. It is a duplicate of another requirement.
5. It is out of the project's scope.

If a requirement fails the quality check, it gets rejected for one of the following possible reasons: *out of scope* / *duplicate* / *conflicting* / *nonsense* / *ambiguous* / *incomplete*; the status with the respective name is assigned to it. When the requirement is updated properly, its status is changed to *reopened*. This status almost equals the initial status *open* and the quality check process restarts. The only difference is that the status *open* corresponds to requirements that have never been checked for quality, whereas the status *reopened* indicates that the respective requirement went through the quality control at least once.

When a requirement passes the quality gateway, it means that its implementation and validation will be attempted.

3.1.3 Overview of system requirements

This section describes the status of the initial system requirements. Their full list can be found in Annex 1: System requirements list. The purpose of this approach is to provide a simple and structured representation of requirements, that will be used as a reference for the current steps to develop the platform and the applications. The requirements will be updated during the project lifetime, according to the emerging needs for new or modified features. Various methods will be applied to improve the understanding of the user needs and the user-perceived qualities of the prototypes. Especially, the requirements will be reviewed during the evaluation of the first application prototypes so that an improved set of requirements is created [see for example (B. Schmidt-Belz, 1999)].

Every requirement listed in the requirements table obtains a unique ID to refer to. The description of a requirement is synthetic but clear.

Current number of requirements: 49 (55 together with the rejected ones)

According to priority:

- Highest: 1
- High: 15
- Medium: 30
- Low: 3
- Lowest: 0

According to requirement type:

- Functional: 43
- Non-Functional: 6

According to status (among the non-rejected ones):

- Validated: 0

- Implemented: 0
- Under implementation: 1
- Part of specification: 47
- Open: 1
- Re-opened: 0
- Consensus needed: 0

3.1.4 Software components gathering - overview

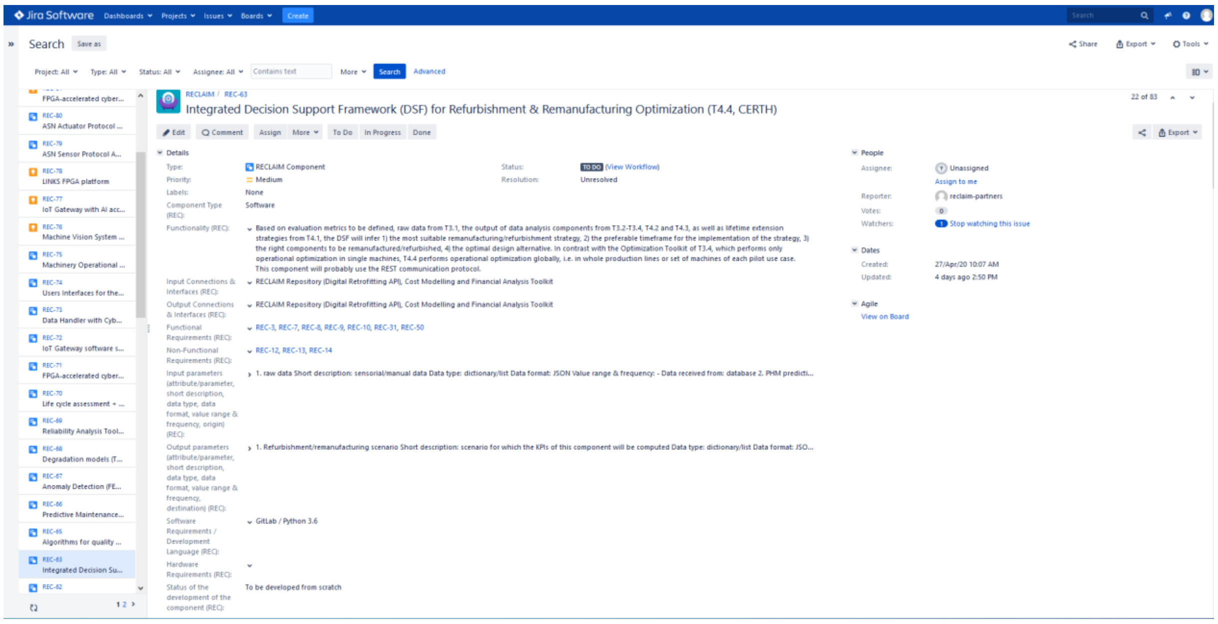
JIRA was used also for filling in different templates for software components. This was done primarily because every component satisfies at least one system requirement, and thus it makes sense to view components and requirements in parallel.

The following fields are included in the component template:

- *Name of New Component/Service:* Name of the architectural element, e.g. Baseline Flexibility Estimation.
- *Functionality:* Short description of the operation of this module/component. A list of functions and operations will be an added value.
- *Input Connections & Interfaces:* Components from which it receives input (input dependencies) and available connection interfaces, e.g. API.
- *Output Connections & Interfaces:* Components to which it sends the results (output dependencies) and available interfaces, e.g. API.
- *Functional Requirements:* Functional requirements that the module satisfies, respective IDs.
- *Non-functional Requirements:* Non-functional requirements that the module satisfies, respective IDs.
- *Input parameters* [*attribute/parameter*, *short description* (if necessary), *data type* (int/string/list/object etc.), *data format* (XML/JSON etc.), *value range & frequency* (measurement unit, range, sampling rate), *origin* (source component/module sending input)]
- *Output parameters* [*attribute/parameter*, *short description* (if necessary), *data type* (int/string/list/object etc.), *data format* (XML/JSON etc.), *value range & frequency* (measurement unit, range, sampling rate), *origin* (source component/module getting output)]
- *Software Requirements / Development Language:* Software requirements related to the architectural element, programming language used during the development of the component.
- *Hardware Requirements:* Hardware requirements of the module, specifications of hardware requirements necessary for the best functionality of the component. Any special sensor included in the sensor specification can be included here as reference.

- *Communications:* Specific communication requirements either for data input or for data output.
- *Status of the development of the component:* Status among "already developed", "partially developed" and "to be developed from scratch".
- *Comments (optional):* Comments by any organization about the correctness of the component. Each person writing some comment should specify their organization, as well as the organization(s) addressed using the symbol "@".

In Figure 6, a filled in component template is shown.



FPGA-accelerated cyber...
REC-80 ASN Actuator Protocol ...
REC-79 ASN Sensor Protocol A...
REC-78 LINKS FPGA platform
REC-77 IoT Gateway with AI acc...
REC-76 Machine Vision System ...
REC-75 Machinery Operational ...
REC-74 Users Interfaces for the...
REC-73 Data Handler with Cyb...
REC-72 IoT Gateway software s...
REC-71 FPGA-accelerated cyber...
REC-70 Life cycle assessment + ...
REC-69 Reliability Analysis Tool...
REC-68 Degradation models (T...
REC-67 Anomaly Detection (FE...
REC-66 Predictive Maintenance...
REC-65 Algorithms for quality ...
REC-63 Integrated Decision Su...
REC-62

RECLAIM / REC-63

Integrated Decision Support Framework (DSF) for Refurbishment & Remanufacturing Optimization (T4.4, CERTH)

22 of 83

Details

Type:	RECLAIM Component
Priority:	Medium
Labels:	None
Component Type (REC):	Software
Functionality (REC):	Based on evaluation metrics to be defined, raw data from T3.1, the output of data analysis components from T3.2-T3.4, T4.2 and T4.3, as well as lifetime extension strategies from T4.1, the DSF will infer 1) the most suitable remanufacturing/refurbishment strategy, 2) the preferable timeframe for the implementation of the strategy, 3) the right components to be remanufactured/refurbished, 4) the optimal design alternative. In contrast with the Optimization Toolkit of T3.4, which performs only operational optimization in single machines, T4.4 performs operational optimization globally, i.e. in whole production lines or set of machines of each pilot use case. This component will probably use the REST communication protocol.
Input Connections & Interfaces (REC):	RECLAIM Repository (Digital Retrofitting API), Cost Modelling and Financial Analysis Toolkit
Output Connections & Interfaces (REC):	RECLAIM Repository (Digital Retrofitting API), Cost Modelling and Financial Analysis Toolkit
Functional Requirements (REC):	REC-3, REC-7, REC-8, REC-9, REC-10, REC-31, REC-50
Non-Functional Requirements (REC):	REC-12, REC-13, REC-14
Input parameters (attribute/parameter, short description, data type, data format, value range & frequency, origin) (REC):	1. raw data Short description: sensorial/manual data Data type: dictionary/list Data format: JSON Value range & frequency: - Data received from: database 2. PHM predicti...
Output parameters (attribute/parameter, short description, data type, data format, value range & frequency, destination) (REC):	1. Refurbishment/remanufacturing scenario Short description: scenario for which the KPIs of this component will be computed Data type: dictionary/list Data format: JSO...
Software Requirements / Development Language (REC):	GitLab / Python 3.6
Hardware Requirements (REC):	
Status of the development of the component (REC):	To be developed from scratch

Status:	TO DO (View Workflow)
Resolution:	Unresolved

People

Assignee:	Unassigned Assign to me
Reporter:	reclaim-partners
Votes:	0
Watchers:	1 Stop watching this issue

Dates

Created:	27/Apr/20 10:07 AM
Updated:	4 days ago 2:50 PM

Agile

View on Board

Figure 6: Screenshot of JIRA with the details of a particular component template. On the left, a part of the list of RECLAIM JIRA templates (requirements, components etc.) is shown.

The component template has been completed for 23 (sub)components.

3.1.5 Hardware specifications gathering - overview

The following template for hardware specifications was filled in also in JIRA by hardware providers:

- Device/Gateway/Infrastructure Description and Functionality:
  - *Name:* Provide the name of the sensor.
  - *Short Description:* Provide a brief statement of the sensor, mentioning its WP/Task Number within the overall architecture.
  - *Measurement:* Provide description of the sensor measurement (directly, how, any restrictions).

  - *Digital/Analog Signals:* Describe the signalling mode (analog, TTL, CMOS, etc.), if applicable for the sensor.
  - *Functionality:* Describe how the sensor functions within the project's system architecture.
- Physical Characteristics:
  - *Dimensions:* length x width x height in mm
  - *Weight:* total weight of sensor in kg
  - *Material:* materials used for its construction
  - *Mounting:* how the sensor is attached
- Operational Characteristics:
  - *Measurement Range:* minimum to maximum values that can be measured by the sensor (e.g. -40 to +80ºC)
  - *Measurement Resolution:* level of measurement (e.g. to 0.01ºC)
  - *Accuracy:* accuracy of the measurement (e.g. ±x% of actual reading)
  - *Zero Error:* amount required to pre-calibrate sensor and/or adjust readings by (e.g. ±0.05ºC)
  - *Temperature:* minimum to maximum temperature levels in ºC: range in which the sensor can operate
  - *Humidity:* minimum to maximum humidity levels in %: range in which the sensor can operate
  - *Pressure:* minimum to maximum pressure levels in Pa/kgm-3/N etc.: range in which the sensor can operate
  - *Lifetime:* specify approximate lifetime under standard operating conditions
- Hardware Requirements:
  - *Power Requirements:* specify electrical power supply required for sensor to operate without disruption
  - *Data Connections:* specify the communication networks and protocols involved, e.g. USB, GSM, WiFi, Bluetooth
  - *Data Format:* specify the output format of the sensor
  - *Data Rate:* specify at what rate data are read/extracted/logged
  - *Data Availability:* specify whether the data stream is continuous, periodic, on demand etc.
  - *Transmission Frequency:* specify the power of the data stream, e.g. X mW, if applicable
- Software Requirements (e.g. API creation):
  - *Software Required:* yes/no
  - *Software Details:* provide details of software required for proper sensor function
- Note: write any important note related to the sensor

During the first iteration of T2.3, the templates of 5 hardware components were completed with all information available at that time. The full list of hardware templates may be found in Annex 2: Hardware components list.

3.1.6 Overall functional architecture

The figure below shows the functional view of the architecture, which is based on the conceptual architecture, the consolidated view and the feedback received by component owners. The Adaptive Sensorial Network with IoT Cybersecurity (Physical Layer) is mainly responsible for sending raw sensorial data to the Real-time Decision Making Layer, which receives also user feedback from the Users Interfaces in order to perform all computations related to data analysis and afterwards send the visualization data to the end user through the Users Interfaces (User Facilities Layer). In particular, the Real-time Decision Making Layer includes the repositories for data storage, and the services and frameworks of the Decision Support Framework, the In-Situ Repair Data Analytics, the AR mechanisms, and the Life Cycle Assessment / Life Cycle Cost component. Finally, it is possible that some configuration data are transferred back from the Real-time Decision Making Layer to the Physical Layer.

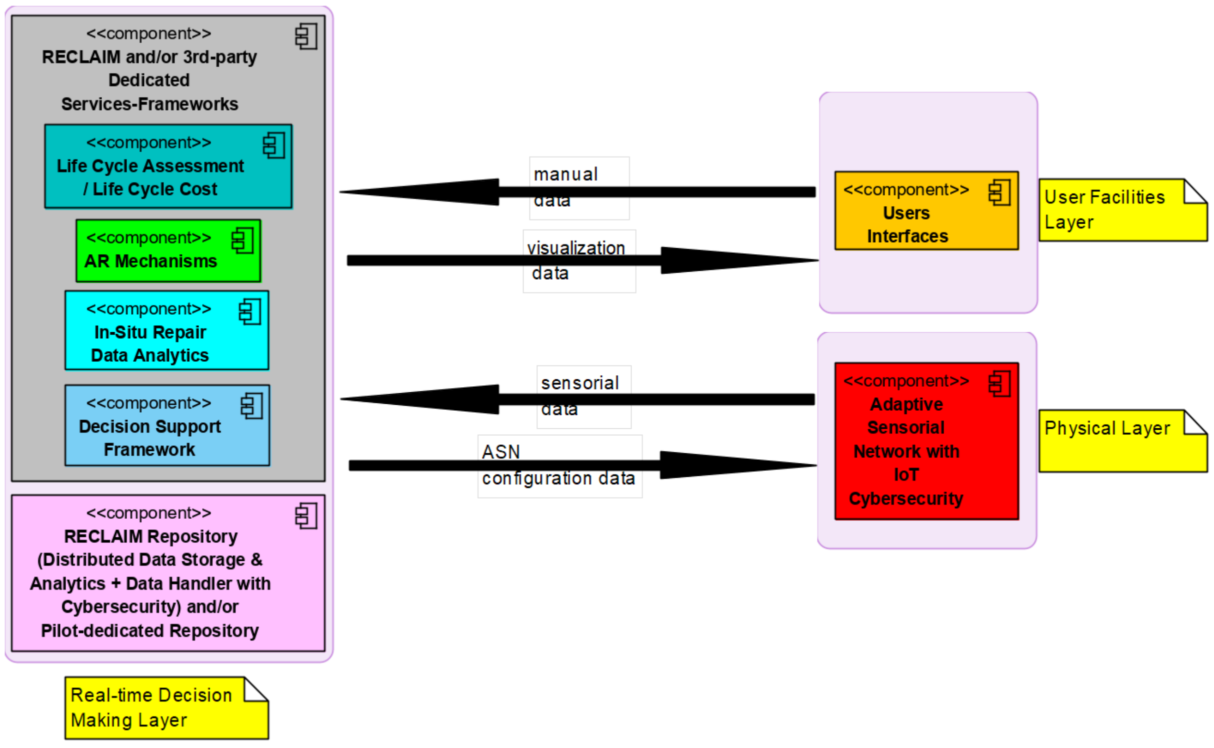
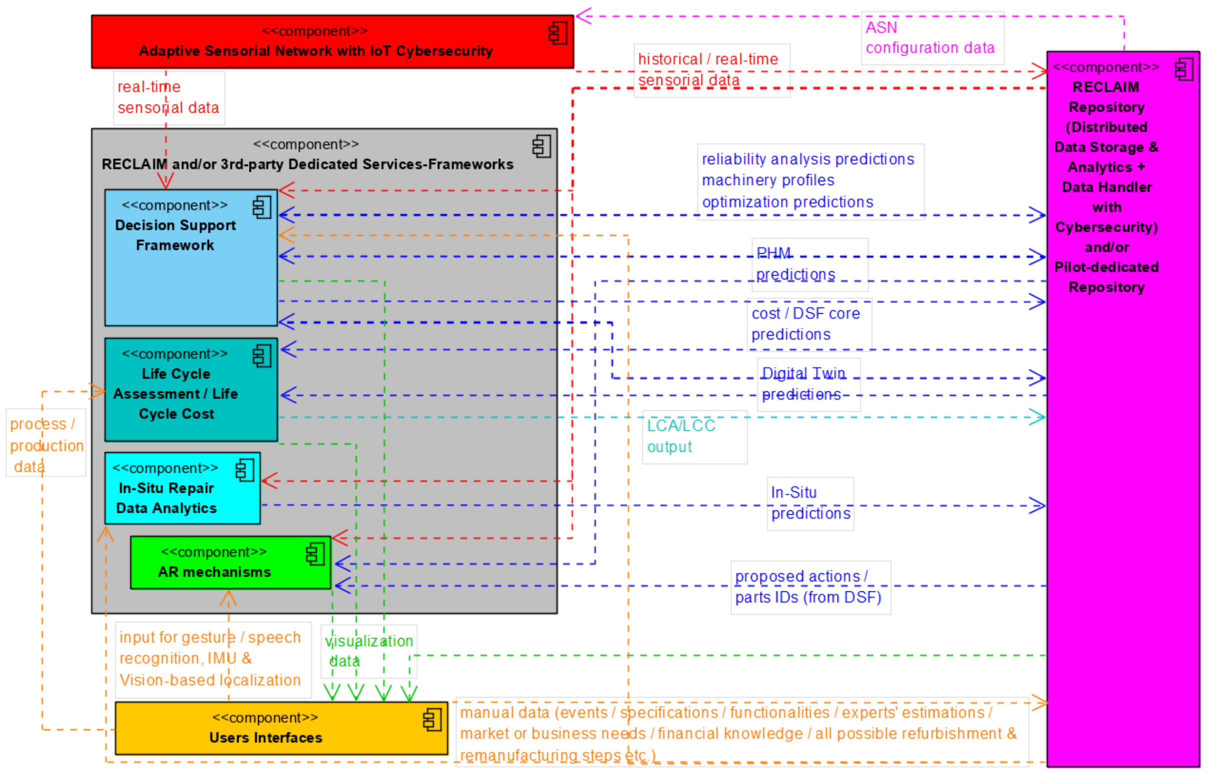

<<component>>
RECLAIM and/or 3rd-party Dedicated Services-Frameworks
<<component>>
Life Cycle Assessment / Life Cycle Cost
<<component>>
AR Mechanisms
<<component>>
In-Situ Repair Data Analytics
<<component>>
Decision Support Framework
<<component>>
RECLAIM Repository (Distributed Data Storage & Analytics + Data Handler with Cybersecurity) and/or Pilot-dedicated Repository
Real-time Decision Making Layer
manual data
visualization data
sensorial data
ASN configuration data
<<component>>
Users Interfaces
User Facilities Layer
<<component>>
Adaptive Sensorial Network with IoT Cybersecurity
Physical Layer

Figure 7 - Functional view diagram

3.1.7 Information view

The information flow diagram, which shows what kinds of data are exchanged between all main software components, follows below. More detailed and component-specific diagrams are shown in the next subsection, which is related to the functional view.



<<component>> Adaptive Sensorial Network with IoT Cybersecurity
real-time sensorial data
historical / real-time sensorial data
ASN configuration data
<<component>> RECLAIM Repository (Distributed Data Storage & Analytics + Data Handler with Cybersecurity) and/or Pilot-dedicated Repository
<<component>> RECLAIM and/or 3rd-party Dedicated Services-Frameworks
<<component>> Decision Support Framework
reliability analysis predictions
machinery profiles
optimization predictions
PHM predictions
cost / DSF core predictions
<<component>> Life Cycle Assessment / Life Cycle Cost
Digital Twin predictions
LCA/LCC output
process / production data
<<component>> In-Situ Repair Data Analytics
In-Situ predictions
<<component>> AR mechanisms
proposed actions / parts IDs (from DSF)
input for gesture / speech recognition, IMU & Vision-based localization
visualization data
<<component>> Users Interfaces
manual data (events / specifications / functionalities / experts' estimations / market or business needs / financial knowledge / all possible refurbishment & remanufacturing steps etc.)

Figure 8 - RECLAIM information flow

3.1.8 Software component details and diagrams

In this subsection, the UML diagrams for all main software components, showing their internal architecture, as well as their internal and external dependencies, are included. Also, the content of the component templates is included.

A hierarchical overview of the components follows:

- Adaptive Sensorial Network with IoT Cybersecurity
  - IoT Gateway software stack
  - FPGA-accelerated cyber security module
  - ASN Sensor Protocol Adaptation
  - ASN Actuator Protocol Adaptation
  - Machine Vision Toolkit
- RECLAIM Repository
  - Distributed data storage and analytics (DRy)
  - Data Handler with Cybersecurity
- Decision Support Framework
  - RECLAIM Reliability Analysis Tool
  - Machinery Operational Profiling

    - Fault Diagnosis and Predictive Maintenance Simulation Engine using Digital Twin
        - Digital Twin for Simulation
        - Anomaly Detection
        - Predictive maintenance
    - Optimization Toolkit for Refurbishment and Remanufacturing Planning
        - Algorithms for quality prediction and process parameter optimization
    - Prognostic and Health Management Toolkit
        - Degradation models
    - Cost Modelling and Financial Analysis Toolkit
    - Integrated Decision Support Framework (DSF) for Refurbishment & Remanufacturing Optimization - core component
- In-Situ Repair Data Analytics
- AR mechanisms
- Life Cycle Assessment
- Users Interfaces for the Integrated Decision Support Framework, the In-Situ Repair Data Analytics and the AR mechanisms

3.1.8.1 Adaptive Sensorial Network with IoT Cybersecurity

The component diagram of the ASN with IoT Cybersecurity follows below, and its subcomponents are described in more detail afterwards.

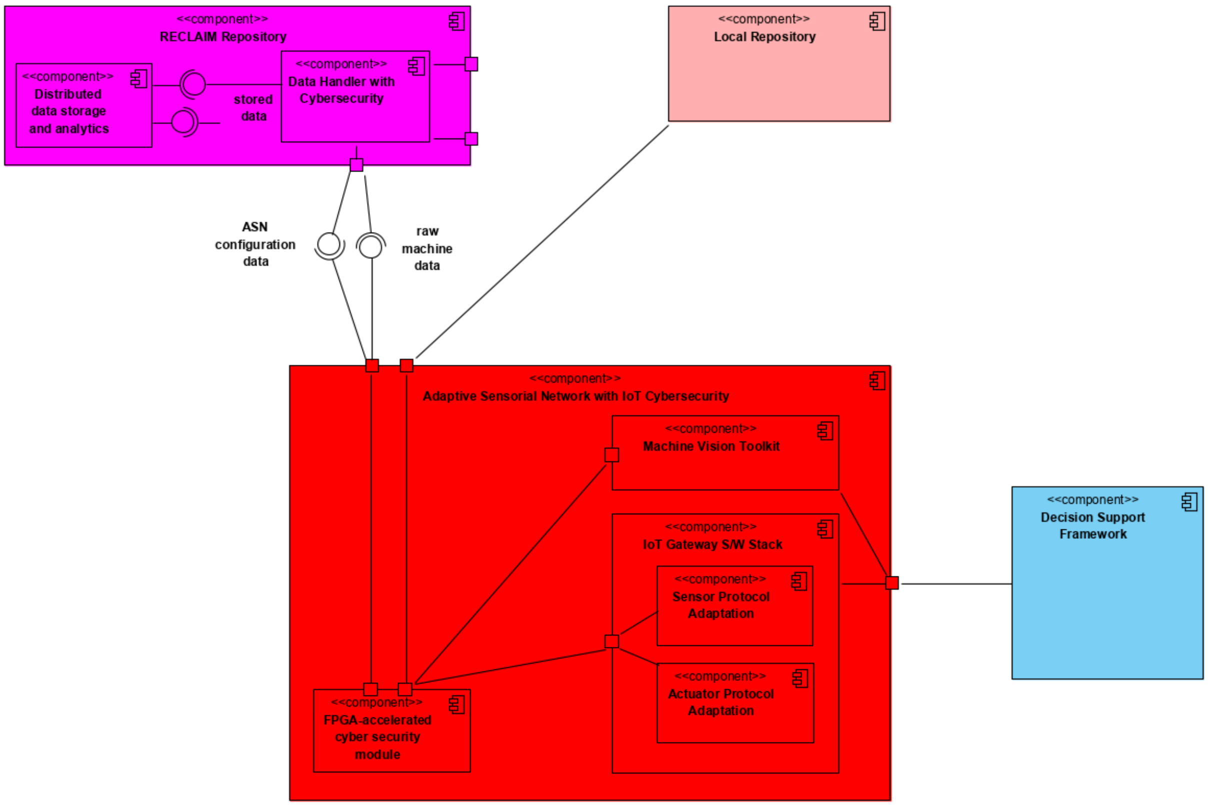


Figure 9 - Adaptive Sensorial Network with IoT Cybersecurity component diagram

3.1.8.1.1 IoT Gateway software stack

Table 2 - Filled-in software component template for the IoT Gateway software stack

Issue Links:	**Relates** relates to REC-77 IoT Gateway with AI acceleration (T3.... relates to REC-79 ASN Sensor Protocol Adaptation (T3.1,... relates to REC-80 ASN Actuator Protocol Adaptation (T3....
Component Type:	Software
Functionality:	The software stack will be comprised of the communication and protocol adaptation services for the IoT device communication and will provide the necessary tools, libraries and middleware for using the tensorflow framework on IoT gateways based on the NVIDIA Jetson platform.

Input Connections & Interfaces:	IEEE1451 packets over UDP over 6LoWPAN, MQTT, NGSI[2]
Output Connections & Interfaces:	IEEE1451 packets over UDP over 6LoWPAN, MQTT, NGSI
Functional Requirements:	REC-23, REC-24
Input parameters (attribute/parameter, short description, data type, data format, value range & frequency, origin):	1. Sensor data Short description: Data acquired from various deployed sensors Data type: data frame with headers Data format: IEEE1451 Value range & frequency: - Data received from: IoT Nodes 2. Actuation data Short description: Data to be transmitted to various deployed actuators Data type: data frames Data format: JSON Value range & frequency: - Data received from: MQTT publishers
Output parameters (attribute/parameter, short description, data type, data format, value range & frequency, destination):	1. Sensor data Short description: Data acquired from various deployed sensors Data type: data frames Data format: JSON Value range & frequency: - Data sent to: MQTT publishers 2. Actuation data Short description: Data to be transmitted to various deployed actuators Data type: data frame with headers Data format: IEEE1451 Value range & frequency: - Data sent to: IoT Nodes
Hardware Requirements:	REC-77
Status of the development of the component:	Partially Developed

[2] data model of FIWARE

3.1.8.1.2 FPGA-accelerated cyber security module

Table 3 - Filled-in software component template for the FPGA-accelerated cyber security module

Issue Links:	Relates relates to REC-47 Cybersecurity monitoring and protecti...
Component Type:	Software
Functionality:	The modules comprising this software will a) facilitate the use of acceleration (FPGA) for detecting and analysing cybersecurity threats; b) provide the monitor and management modules of the Cyber security GW; c) provide the cyber threat detection and analysis modules; d) provide the threat mitigation/control modules, that will block the threats; and e) the communication/interfacing modules, that will enable for the interaction with other external components (central logging server, software agents, etc.)
Functional Requirements:	REC-47
Input parameters (attribute/parameter, short description, data type, data format, value range & frequency, origin):	1. Cyber threat detection and analysis module Short description: Anasyses the network traffic towards detecting cyber security threats. Data type: Network traffic Data format: Raw bytes, JSON, XML Value range & frequency: - Data received from / sent to: GW's Network interfaces 2. FPGA-ARM middleware Short description: facilitates the use of FPGA from the SW (ARM) modules. Data type: - Data format: binary Value range & frequency: - Data received from / sent to: SW modules deployed in the ARM, Accelerated functionalities deployed in the FPGA. 3. Firewall Short description: embedded firewall functionallity adapted per equipment or set of machineries/equipment at the production line. Data type: Control commands, Network traffic Data format: JSON for delivering control commands and describing the firewall rules. 4. Accelerated encryption Short description: based on AES algorithm specifically designed for FPGA. Use for encrypt sensitive information

	where needed Data type: - Data format: Value range & frequency: - Data received from / sent to: Machinery Data Collectors, RECLAIM repository, In-Situ Repair Data Analytics 5. Local Dashboard for Visualisation and Management Short description: Dashboard for Visualisation and Management Data type: - Data format: - Value range & frequency: - Data received from / sent to: -
Output parameters (attribute/parameter, short description, data type, data format, value range & frequency, destination):	Similar to the input parameters.
Hardware Requirements:	FPGA-accelerated network appliance supplied by FINT
Communications:	Ethernet interface x 2
Status of the development of the component:	Partially Developed

3.1.8.1.3 ASN Sensor Protocol Adaptation

Table 4 - Filled-in software component template for the ASN Sensor Protocol Adaptation

Issue Links:	**Relates** relates to REC-72 IoT Gateway software stack (T3.1, FIN... relates to REC-77 IoT Gateway with AI acceleration (T3....
Functionality:	This component is responsible for the protocol adaptation of the deployed sensor devices in the shop floors which communicate through the IoT Gateways. It registers these devices through the RECLAIM Repository and exposes a unified communication interface, irrespective of the underlying communication protocol of each device. It

	employs a time ordered volatile queue mechanism ensuring that no data are lost in case of brief communication failures with the RECLAIM Repository and timestamps sensor data coming from sensors that do not inherently provide time information.
Input Connections & Interfaces:	Sensor devices connected to the IoT Gateway through either Ethernet, WiFi, 6LoWPAN over IEEE802.15.4 or LoRa.
Output Connections & Interfaces:	RECLAIM Repository, Decision Support Framework.
Input parameters (attribute/parameter, short description, data type, data format, value range & frequency, origin):	Adapted to the respective data and communication format supported by the respective sensor device. Supported protocols: - IEEE1451 over IEEE1451.5 - MODBUS-TCP - MQTT - LoRaWAN
Output parameters (attribute/parameter, short description, data type, data format, value range & frequency, destination):	- Analog input value - Digital input value NGSI JSON format
Status of the development of the component:	Partially Developed

3.1.8.1.4 ASN Actuator Protocol Adaptation

Table 5 - Filled-in software component template for the ASN Actuator Protocol Adaptation

Issue Links:	**Relates** relates to REC-72 IoT Gateway software stack (T3.1, FIN... relates to REC-77 IoT Gateway with AI acceleration (T3....
Functionality:	This component is responsible for the protocol adaptation of the deployed actuator devices which communicate through the IoT Gateways. It registers these devices through the RECLAIM Repository and exposes a unified communication interface, irrespective of the underlying

	communication protocol of each device.
Input Connections & Interfaces:	RECLAIM Repository, Decision Support Framework.
Output Connections & Interfaces:	Actuator devices connected to the IoT Gateway through either Ethernet, WiFi, 6LoWPAN over IEEE802.15.4 or LoRa.
Input parameters (attribute/parameter, short description, data type, data format, value range & frequency, origin):	- Analog output value - Digital output value NGSI JSON format
Output parameters (attribute/parameter, short description, data type, data format, value range & frequency, destination):	Adapted to the respective data and communication format supported by the respective actuator device. Supported protocols: - IEEE1451 over IEEE1451.5 - MODBUS-TCP - MQTT - LoRAWAN
Status of the development of the component:	Partially Developed

3.1.8.1.5 Machine Vision Toolkit

Table 6 - Filled-in software component template for the Machine Vision Toolkit

Issue Links:	**Relates** relates to REC-76 Machine Vision System (T3.1, FINT)
Functionality:	The toolkit is comprised of: computer vision library which provides the modules used for object identification and attributes recognition, CUDA toolkit for enabling the use of CUDA cores of the hardware platform, machine learning platform with CUDA support which provides the means for hardware acceleration of the computer vision library, Python services which utilize the above, perform the object analysis and generate output based on the MachineVision data model.
Input Connections &	Computer vision camera with Ethernet and/or USB

Interfaces:	interface.
Output Connections & Interfaces:	Output is sent to the RECLAIM Repository and the Decision Support Framework.
Input parameters (attribute/parameter, short description, data type, data format, value range & frequency, origin):	- Video feed from computer vision cameras. - Requested part IDs in NGSI (JSON) format.
Output parameters (attribute/parameter, short description, data type, data format, value range & frequency, destination):	Output parameters: - Position of computer vision camera in the plant - Identified Part ID - Part match with requested part ID boolean value - Part correctly aligned boolean value Output format: NGSI JSON
Hardware Requirements:	REC-76
Status of the development of the component:	To be developed from scratch

3.1.8.2 RECLAIM Repository

The RECLAIM Repository consists of two main sub-components: the Distributed data storage and analytics (DRy) and the Data Handler with cybersecurity.

3.1.8.2.1 Distributed data storage and analytics (DRy)

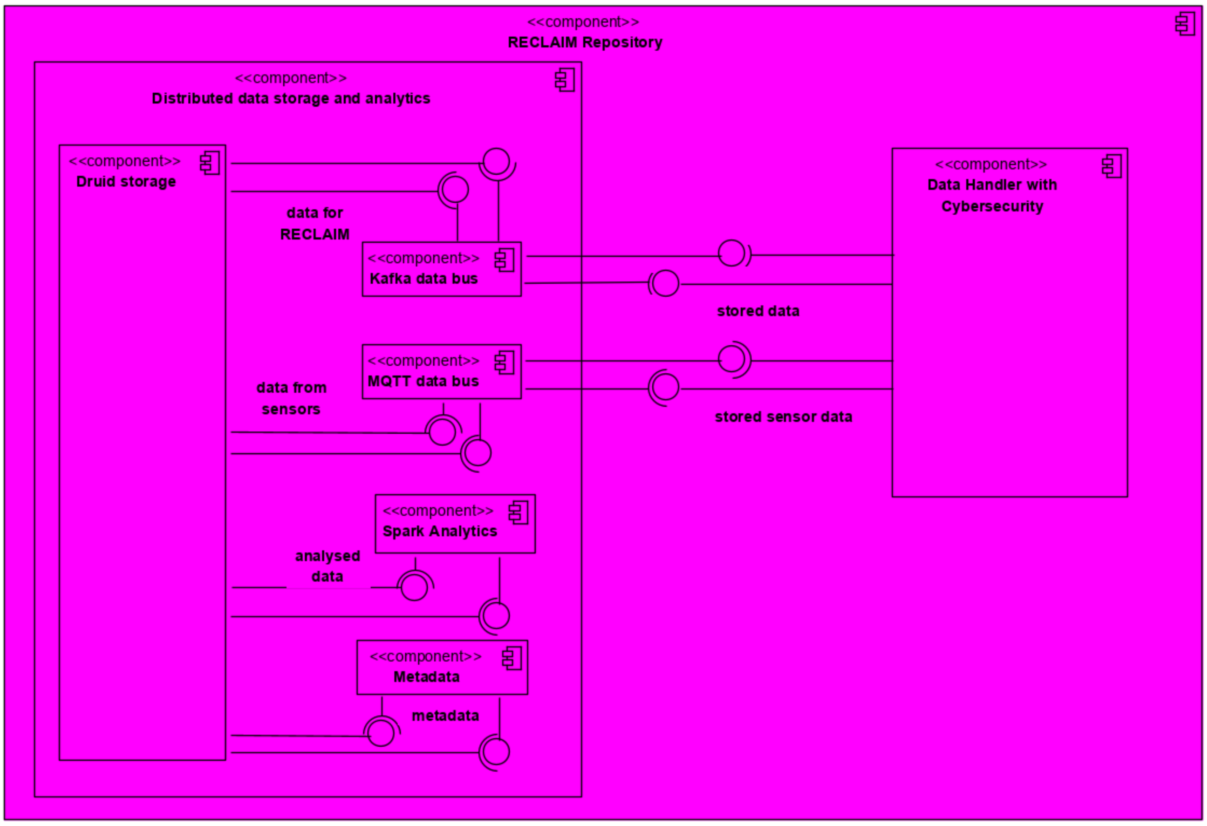


Figure 10: Distributed data storage and analytics component diagram

Table 7 - Filled-in software component template for the Distributed data storage and analytics

Component Type:	Software
Functionality:	The Distributed Data Storage and Analytics component is a storage and a set of data aggregators tailored according to the requirements from RECLAIM. The storage will store data from the machines connected to the RECLAIM environment as well as the data calculated by the different algorithms. Internally, the component accounts with two data buses: Kafka data bus to serve the data stored in the storage to the rest of the RECLAIM system and an MQTT data bus to store the data coming from the sensors. In addition, two extra modules are present: one to generate the necessary metadata for the calculation of the three indexes and another one (based on Spark) to analyse the data and extract patterns from the underlying data. DRy ICE is based on Apache Druid, a column-store distributed database. Since Joins are not supported and the storage model is based on columnar decomposition, it's

	based on NoSQL technology. This software is part of the RECLAIM repository together with the Data Handler with Cybersecurity.
Input Connections & Interfaces:	REC-73 (Data Handler with Cybersecurity)
Output Connections & Interfaces:	REC-73 (Data Handler with Cybersecurity)
Functional Requirements:	REC-1, REC-3, REC-5, REC-6, REC-7, REC-8, REC-11, REC-23, REC-24, REC-26, REC-27, REC-35
Input parameters (attribute/parameter, short description, data type, data format, value range & frequency, origin):	No need to pre-define data models before data ingestion
Software Requirements / Development Language:	Series of docker images and access to physical space for data storage. Based on Druid and Spark
Hardware Requirements:	Typically runs in a server box
Status of the development of the component:	Partially Developed

3.1.8.2.2 Data Handler with Cybersecurity

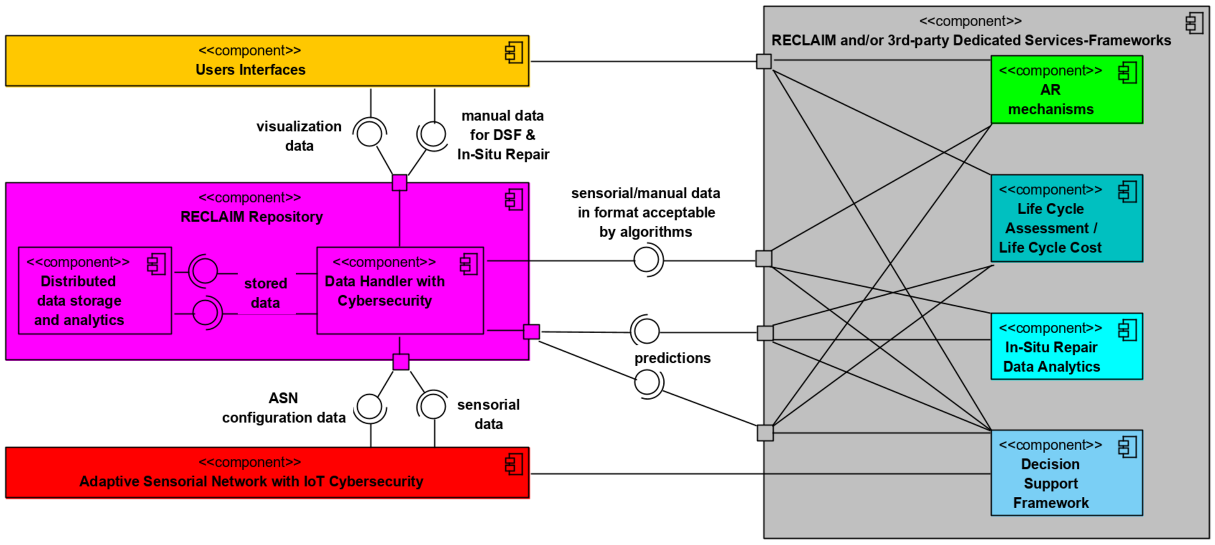

<<component>> Users Interfaces
visualization data
manual data for DSF & In-Situ Repair
<<component>> RECLAIM Repository
<<component>> Distributed data storage and analytics
stored data
<<component>> Data Handler with Cybersecurity
ASN configuration data
sensorial data
<<component>> Adaptive Sensorial Network with IoT Cybersecurity
sensorial/manual data in format acceptable by algorithms
predictions
<<component>> RECLAIM and/or 3rd-party Dedicated Services-Frameworks
<<component>> AR mechanisms
<<component>> Life Cycle Assessment / Life Cycle Cost
<<component>> In-Situ Repair Data Analytics
<<component>> Decision Support Framework

Figure 11 - Data Handler with Cybersecurity component diagram

Table 8 - Filled-in software component template for the Data Handler with Cybersecurity

Component Type:	Software
Functionality:	This component transforms the format of data exchanged between the RECLAIM Repository and other components. A holistic data model (information model) will be used for storing data in the database of the RECLAIM Repository. The data handler will include also cybersecurity mechanisms that will ensure the secure data communication with the Distributed data storage and analytics component. It needs to be clarified among partners during which of the following transfers the data handler will need to be transforming the format: i] Input from machinery data collectors and end users to RECLAIM Repository. ii] Input from RECLAIM Repository database to the algorithmic components (DSF, In-Situ Repair Data Analytics, AR mechanisms). iii] Output from algorithmic components to RECLAIM Repository database. iv] Output from RECLAIM Repository database to users interfaces.
Input Connections & Interfaces:	Adaptive Sensorial Network with IoT Cybersecurity (Digital Retrofitting API), Distributed Storage and Analytics, algorithmic components (Digital Retrofitting API), Users Interfaces
Output Connections & Interfaces:	Users Interfaces, Distributed Storage and Analytics, algorithmic components (Digital Retrofitting API, RECLAIM

	3rd party API)
Functional Requirements:	REC-3, REC-5, REC-7, REC-8, REC-9, REC-10, REC-11, REC-23, REC-24, REC-35, REC-37
Non-Functional Requirements:	REC-12, REC-13, REC-14, REC-36
Input parameters (attribute/parameter, short description, data type, data format, value range & frequency, origin):	1. Input from machinery data collectors and end users to RECLAIM Repository Short description: - Data type: - Data format: format acceptable by pilots Value range & frequency: - Data received from: Adaptive Sensorial Network + Cybersecurity 2. Input from RECLAIM Repository database to the algorithmic components Short description: - Data type: - Data format: holistic data model format Value range & frequency: - Data received from: Distributed data storage and analytics (DRy) 3. Output from algorithmic components to RECLAIM Repository database Short description: - Data type: - Data format: format acceptable by algorithmic components Value range & frequency: - Data received from: Decision Support Framework, In-situ Repair Data Analytics, AR mechanisms, Life Cycle Assessment / Life Cycle Cost 4. Output from RECLAIM Repository database to Users Interfaces Short description: - Data type: - Data format: holistic data model format Value range & frequency: - Data received from: Distributed data storage and analytics (DRy)
Output parameters (attribute/parameter, short description, data type, data format, value range & frequency,	1. Input from machinery data collectors and end users to RECLAIM Repository Short description: - Data type: - Data format: holistic data model format Value range & frequency: - Data sent to: Distributed data storage and analytics (DRy)

destination):	2. Input from RECLAIM Repository database to the algorithmic components Short description: - Data type: - Data format: format acceptable by algorithmic components Value range & frequency: - Data sent to: Decision Support Framework, In-situ Repair Data Analytics, AR mechanisms, Life Cycle Assessment / Life Cycle Cost 3. Output from algorithmic components to RECLAIM Repository database Short description: - Data type: - Data format: holistic data model format Value range & frequency: - Data sent to: Distributed data storage and analytics (DRy) 4. Output from RECLAIM Repository database to Users Interfaces Short description: - Data type: - Data format: format acceptable by pilots Value range & frequency: - Data sent to: Users Interfaces
Software Requirements / Development Language:	Java / Python / C++
Hardware Requirements:	no requirements
Communications:	no communication requirements
Status of the development of the component:	To be developed from scratch

3.1.8.3 Decision Support Framework

The Decision Support Framework (DSF) includes all subcomponents corresponding to WP4, WP3 and the Reliability Analysis Tool from T2.5. Its main subcomponent is the DSF core, corresponding to T4.4. The high-level diagram of the DSF is shown below, whereas the more detailed diagrams of its subcomponents follow right after.

In principle, the DSF subcomponents receive historical and/or real-time input data from the RECLAIM or a Pilot Repository, but in particular cases (at least for the

PHM Toolkit) dedicated interfaces are going to be implemented for faster real-time data acquisition directly from the ASN. As described later in more detail, some subcomponents of the DSF will have their own APIs, mainly for data visualization but also for user feedback with manual data in some cases, whereas other subcomponents will post their predictions to the repository so that they can be visualized through its API or reused by the same and/or other (sub)components. Finally, it will be possible for a subcomponent to execute another subcomponent with some input and obtain respective output. This will be needed at least in case that the "other subcomponent" is the Cost Modelling & Financial Analysis Toolkit, which will not post directly its output for particular input, but the Optimization Toolkit and the DSF Core will execute it for multiple input considerations in order to optimize cost w.r.t. the input, and afterwards the last two subcomponents may post the cost for particular scenarios, e.g. the optimized one and the naive approach.

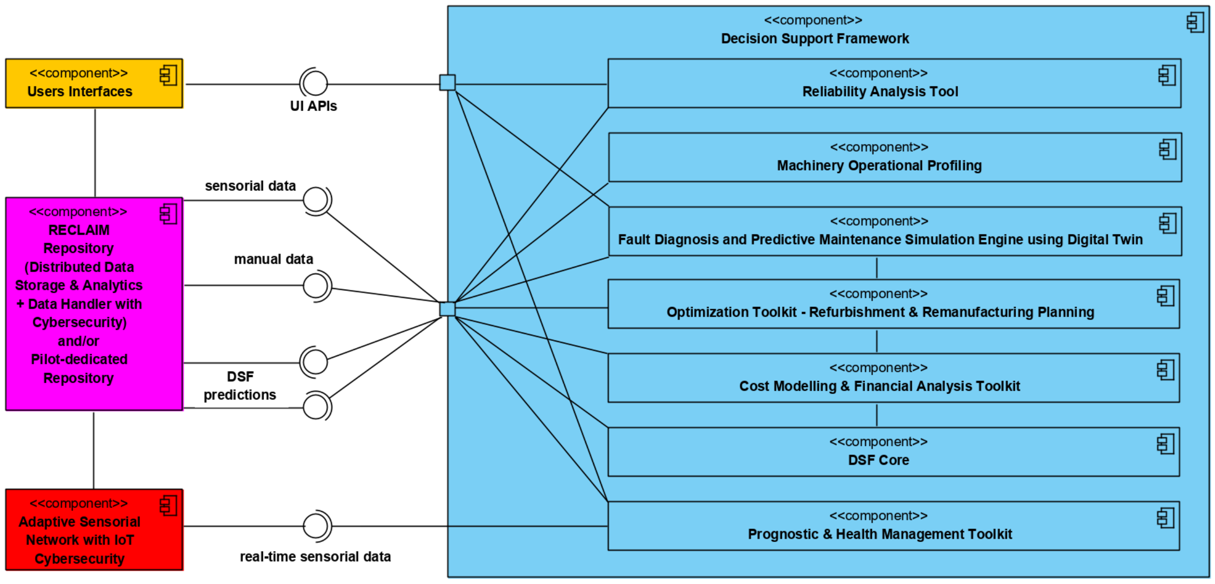
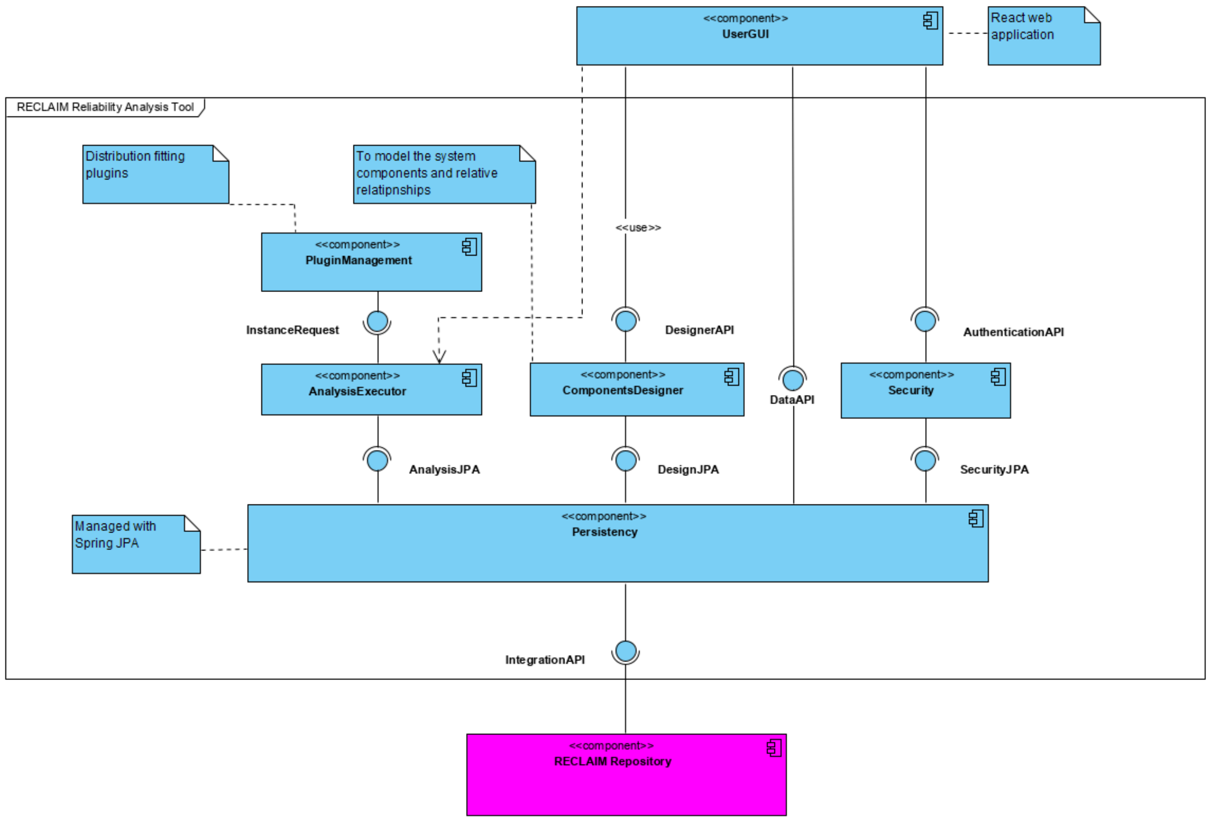

<<component>> Decision Support Framework
<<component>> Users Interfaces
UI APIs
<<component>> Reliability Analysis Tool
<<component>> Machinery Operational Profiling
sensorial data
<<component>> RECLAIM Repository (Distributed Data Storage & Analytics + Data Handler with Cybersecurity) and/or Pilot-dedicated Repository
<<component>> Fault Diagnosis and Predictive Maintenance Simulation Engine using Digital Twin
manual data
<<component>> Optimization Toolkit - Refurbishment & Remanufacturing Planning
DSF predictions
<<component>> Cost Modelling & Financial Analysis Toolkit
<<component>> DSF Core
<<component>> Adaptive Sensorial Network with IoT Cybersecurity
<<component>> Prognostic & Health Management Toolkit
real-time sensorial data

Figure 12 - DSF high-level component diagram

3.1.8.3.1 RECLAIM Reliability Analysis Tool



<<component>> UserGUI
React web application
RECLAIM Reliability Analysis Tool
Distribution fitting plugins
To model the system components and relative relatipnships
<<use>>
<<component>> PluginManagement
InstanceRequest
DesignerAPI
AuthenticationAPI
<<component>> AnalysisExecutor
<<component>> ComponentsDesigner
DataAPI
<<component>> Security
AnalysisJPA
DesignJPA
SecurityJPA
Managed with Spring JPA
<<component>> Persistency
IntegrationAPI
<<component>> RECLAIM Repository

Figure 13 - Reliability Analysis Tool component diagram

Table 9 - Filled-in software component template for the Reliability Analysis Tool

Component Type:	Software
Functionality:	The Reliability Analysis Tool implements statistic analysis to calculate machines reliability and average residual useful life. Starting from historical data (e.g. failure occurrences) of machine’s components and/or sub-systems, the Reliability Analysis Tool is able to build the related failures probability functions (2-parameter Weibull, 3-parameter Weibull etc.) and to calculate failure probability of the single components and of the overall machine. This tool has to be used to develop simple analysis in order to have a first overview of machines' reliability and status. It includes a simple interface supporting companies in providing the required data and in performing the analysis. The tool is composed by the following sub-components: -PluginManagement: Since it may be required to have custom distribution fitting functions, a plugin approach has been chosen. The component will handle all the aspects of managing the various plugins installed and to be installed on

	the modelling platform. -UserGUI: This component represents the whole GUI that the user will be presented with. Not only the graphic aspect but also the retrieval of information to be displayed on the latter. -AnalysisExecutor: Component in charge of executing the analysis upon user request. The executor will initialize the required function from the PluginManagement component and save the results on database, where they can be consulted by the user or stored into the RECLAIM repository. -ComponentsDesigner: This component will handle the design of the system structure, allowing the user to design a flow diagram that represents the components in the system. -Security: This component is in charge of authenticating the users and allowing them to operate only on their systems. -Persistency: The persistency component is in charge of handling all the aspects of the data storage of the platform. An embedded database like H2 or a server database like PostgreSQL will be used. The integration with the database and the rest of the application will likely be implemented using the Spring JPA interfaces. Most of the data will be saved within the tool database, while the data necessary for the calculation of the health index (T3.2) will be made available in the RECLAIM Repository.
Input Connections & Interfaces:	RECLAIM Repository (IntegrationAPI)
Output Connections & Interfaces:	RECLAIM Repository (IntegrationAPI), UserGUI (DesignerAPI, DataAPI, AuthenticationAPI)
Functional Requirements:	REC-3, REC-6, REC-7, REC-9
Non-Functional Requirements:	REC-12
Input parameters (attribute/parameter, short description, data type, data format, value range & frequency, origin):	1. MTTR Short description: Historical data on maintenance Data type: list of floats Data format: CSV Value range & frequency: >0 Data received from: database/file 2. MTTF Short description: Historical data on failure Data type: list of floats Data format: CSV

	Value range & frequency: >0 Data received from: database/file 3. MTBF Short description: Historical data on failure Data type: list of floats Data format: CSV Value range & frequency: >0 Data received from: database/file 4. Other data Short description: Other data that allows to calculate MTTR, MTTF or MTBF could be included (e.g. maintenance dates) Data type: - Data format: - Value range & frequency: - Data received from: -
Software Requirements / Development Language:	Python
Status of the development of the component:	Already Developed

3.1.8.3.2 Machinery Operational Profiling

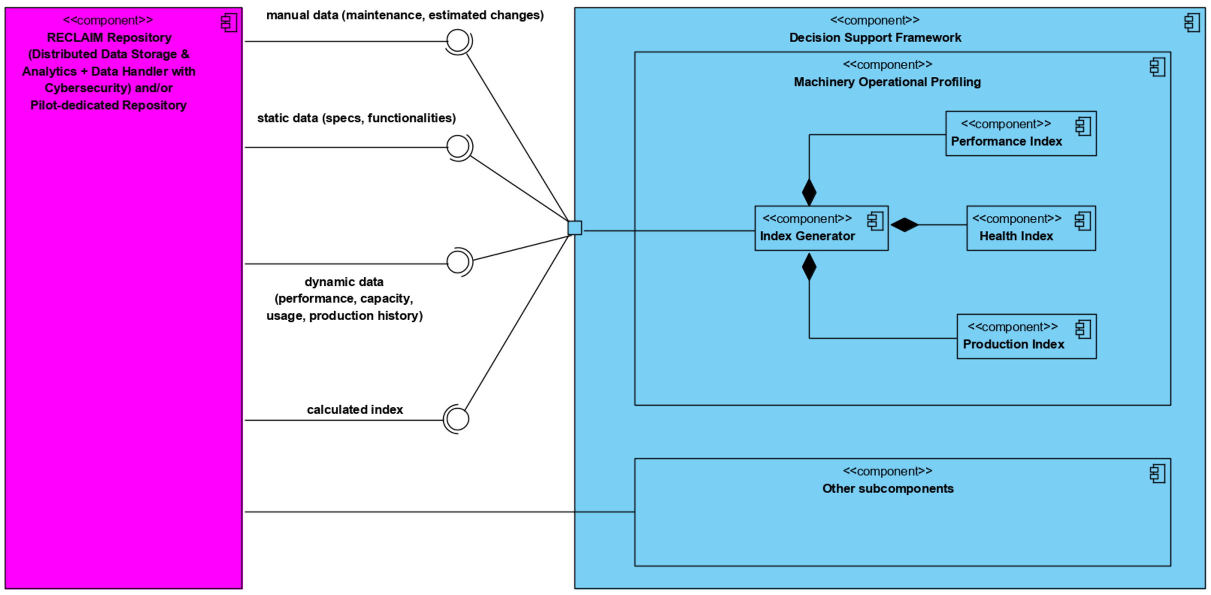

<<component>>
RECLAIM Repository (Distributed Data Storage & Analytics + Data Handler with Cybersecurity) and/or Pilot-dedicated Repository
manual data (maintenance, estimated changes)
static data (specs, functionalities)
dynamic data (performance, capacity, usage, production history)
calculated index
<<component>>
Decision Support Framework
<<component>>
Machinery Operational Profiling
<<component>>
Performance Index
<<component>>
Index Generator
<<component>>
Health Index
<<component>>
Production Index
<<component>>
Other subcomponents

Figure 14 - Machinery Operational Profiling component diagram

Table 10 - Filled-in software component template for the Machinery Operational Profiling

Component Type:	Software
Functionality:	The Machinery Operational Profiling (T3.2) algorithm calculates the three indexes (Health, Performance, Production) to generate the profile of the machine based on the data and parameters provided by the machines of the user partners. These data can either be manual (based on manually entered maintenance data or estimated changes), static (based on specification and functionalities from the machines) or dynamic (based on capacity, usage, history…). In terms of dependency, the Machinery Operational Profiling is part of the Decision Support Framework and the calculated indexes will then be stored in the RECLAIM Repository.
Input Connections & Interfaces:	REC-59 [Distributed data storage and analytics (DRy) (T3.2, ICE)] through REC-73 (Data Handler with Cybersecurity)
Output Connections & Interfaces:	REC-59 [Distributed data storage and analytics (DRy) (T3.2, ICE)] through REC-73 (Data Handler with Cybersecurity)
Functional Requirements:	REC-3, REC-34, REC-39, REC-52
Non-Functional Requirements:	REC-36
Input parameters (attribute/parameter, short description, data type, data format, value range & frequency, origin):	Data from machines, see template from D3.2
Output parameters (attribute/parameter, short description, data type, data format, value range & frequency, destination):	Health-index, Performance-index, Production-index
Software Requirements / Development	Ad-hoc development (most probably based either on Java or Python)

Language:	
Status of the development of the component:	To be developed from scratch

3.1.8.3.3 Fault Diagnosis and Predictive Maintenance Simulation Engine using Digital Twin

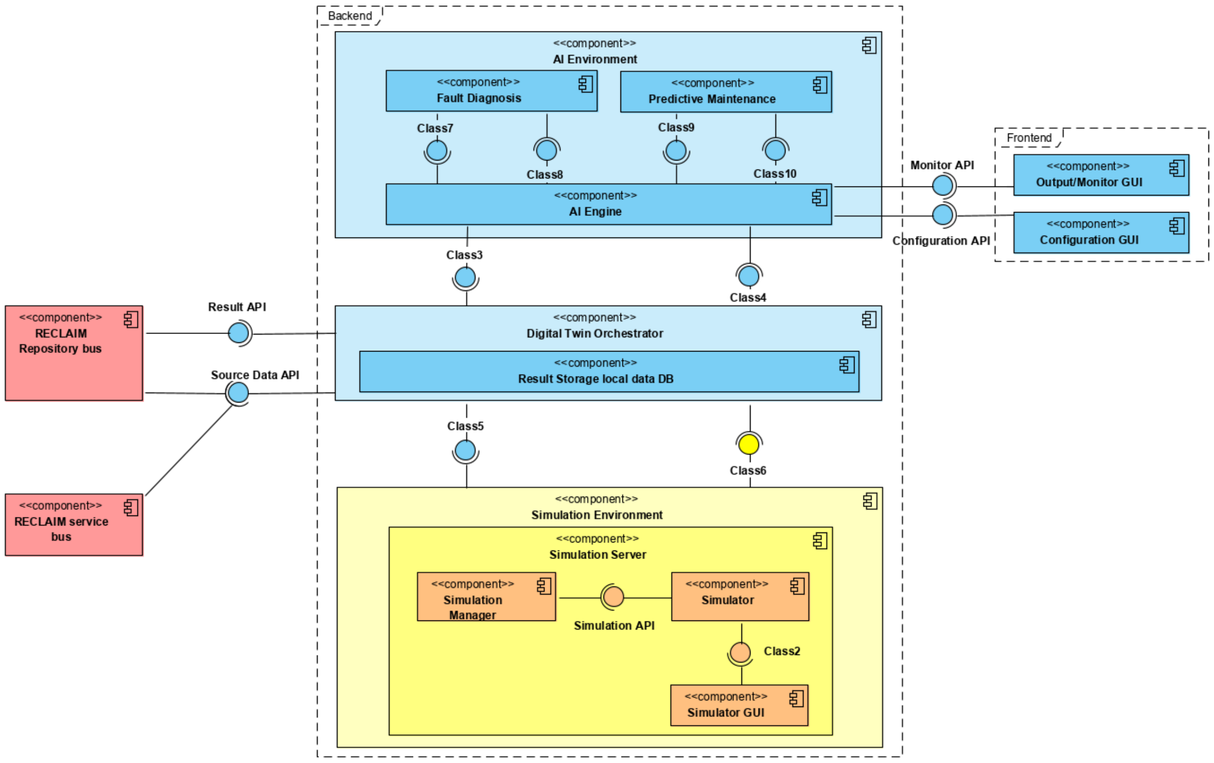


Figure 15 - Fault Diagnosis and Predictive Maintenance Simulation Engine using Digital Twin component diagram

Table 11 - Filled-in software component template for the Fault Diagnosis and Predictive Maintenance Simulation Engine using Digital Twin

Component Type:	Software
Functionality:	This is the component corresponding to T3.3 and building block 6. The main goal of this component is to create a Digital Twin of the factory environment and to use it to monitor and predict the performance and status of factory assets. This will allow providing to the user all the features needed to schedule the maintenance works on the machines to: avoid failures being predicted by the "Prognostic and Health" algorithms defined in the building block 5; to perform proper maintenance planning, optimizing the production throughput and reducing the production lines

	stoppages. This component includes the following sub-components: REC-61, REC-66 and REC-67. Furthermore, this component includes the following components: 1) AI environment - it is the engine leveraged to host and run the Fault Diagnosis and Predictive Maintenance algorithms (including REC-66 and REC-67) 2) AI engine - it is hosted in the AI environment and it is used to abstract the heterogeneous algorithms of Fault Diagnosis and Predictive Maintenance and to control their interactions. 3) Orchestrator - It is used to orchestrate all tasks of the component, coordinating the interactions among AI Engine and the distributed Simulation Environment. Furthermore, it receives the historical and real-time data from REC-72, store them and process them with data quality mechanisms. 4) Simulation Environment - it is a distributed environment composed by a set of distributed Digital Twin for simulation (REC-61) running on different machines, each one wrapped by a Simulation Manager that expose its features through common API.
Input Connections & Interfaces:	Distributed data storage and analytics REC-59 (through Data Handler - REC-73) IoT Gateway software stack (REC-72)
Output Connections & Interfaces:	Distributed data storage and analytics REC-59 (through Data Handler - REC-73), Optimization Toolkit for Refurbishment and Remanufacturing (REC-62)
Functional Requirements:	REC-3, REC-7, REC-8, REC-9, REC-10, REC-11, REC-35, REC-36
Input parameters (attribute/parameter, short description, data type, data format, value range & frequency, origin):	Raw data Short description: - Data type: dictionary/list Data format: JSON Value range & frequency: - Data received from: database
Output parameters (attribute/parameter, short description, data type, data format, value range & frequency,	Fault diagnosis and predictive mantainance predictions Short description: - Data type: dictionary/list Data format: JSON Value range & frequency: -

destination):	Data sent to: database
Software Requirements / Development Language:	Java, OSGi, Docker
Status of the development of the component:	Partially Developed

3.1.8.3.3.1 Digital Twin for Simulation

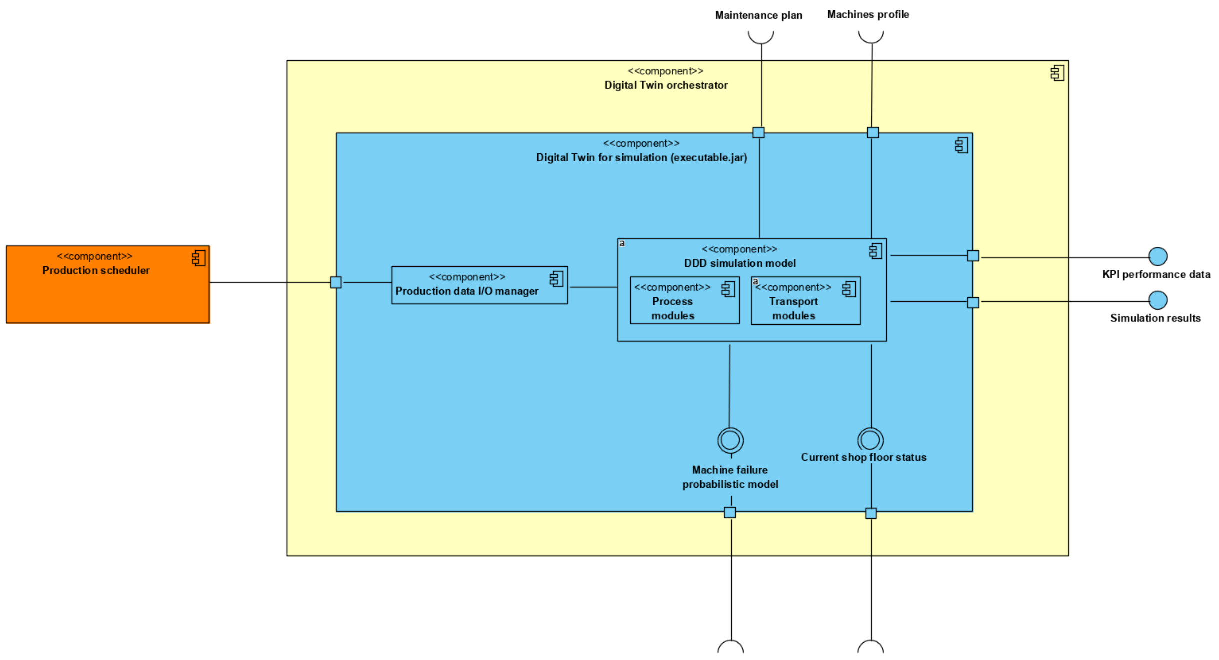


Figure 16 - Digital Twin for Simulation component diagram

Table 12 - Filled-in software component template for the Digital Twin for Simulation

Component Type:	Software
Functionality:	This component is part of the Fault Diagnosis and Predictive Maintenance Simulation Engine using Digital Twin to be developed in T3.3 and building block 6. The ultimate goal is to evaluate different maintenance scenarios and production strategies by means of discrete simulation to reduce the impact of maintenance activities on the performances of a production system. It is composed by 2 main modules: • Production data I/O manager that imports the production sequence over a period of time from a legacy system (Production scheduler) and converts it into a readable input for the DDD simulation model component.

	• DDD simulation model that allows to create the simulation model of a manufacturing shop floor. It receives as input: o the defined maintenance plan over a period of time, o the machine profile of each production resources to be simulated (from T3.2), o the machine failure probabilistic model as defined in the Predictive Maintenance component developed in T3.3, o the current shop floor status mainly in terms of WIP, machine/tool utilization time from the last failure, machine/resource availability, etc. It is composed by 2 sub-component types: o Process modules: to describe the production processes to be simulated along with control logics, priority, MTBF, MTTR, etc. o Transport modules: to describe the material flow in the shop floor along with the control logic, the routing rules The digital twin for simulation provides, as output, a list of KPIs, such as: average throughput of the shop floor, average utilization of each resource, average lead time, average WIP. Furthermore it can provide detailed simulation results such as the utilization time of each resource over a period of time, the duration of every failure happened during a simulation run, the number of failures of each resources, etc.
Input Connections & Interfaces:	Digital Twin orchestrator (machinery profiles, production plan, predictive maintenance model)
Output Connections & Interfaces:	Digital Twin orchestrator (production KPIs: resource saturation, productivity, production lead time,...)
Functional Requirements:	REC-55
Input parameters (attribute/parameter, short description, data type, data format, value range & frequency, origin):	1. machinery profiles Short description: Data about machinery capacity, maintenance, performances, etc. for each resource in a shop floor Data type: - Data format: - Value range & frequency: - Data received from: RECLAIM Repository 2. raw data Short description: Data about current machines status Data type: -

	Data format: - Value range & frequency: - Data received from: RECLAIM Repository 3. Production plan Short description: The production plan of a shop floor to be simulated Data type: - Data format: - Value range & frequency: - Data received from: RECLAIM Repository 4. Predictive maintenance model Short description: A model for each machine in the simulation model Data type: - Data format: - Value range & frequency: - Data received from: -
Output parameters (attribute/parameter, short description, data type, data format, value range & frequency, destination):	1. KPIs visualization Short description: Simulation results are shown to the end user Data type: - Data format: - Value range & frequency: - Data sent to: end users 2. Simulation model and result saving Short description: Simulation results are saved in the repository Data type: - Data format: - Value range & frequency: - Data sent to: RECLAIM Repository
Software Requirements / Development Language:	Java
Status of the development of the component:	Partially Developed

3.1.8.3.3.2 Anomaly Detection

Table 13 - Filled-in software component template for the Anomaly Detection

Component Type:	Software

Functionality:	Techniques to find abnormal behaviors that deviate from normal process conditions to raise warnings and find root causes for the problem. This algorithm will be feedback directly with sensor data (when possible and pertinent) or transformed data from the pilots in order to be more interpretable. Based on the analysis of data streaming, the algorithm should indicate if a warning should be sent to the key personnel to check the system. This algorithm is the first frontline of analysis from shop-floor components in order to understand machine's health. There might be some overlap with the ICE health index from T3.2, however different approaches might / will be used but further alignment is required.
Input Connections & Interfaces:	RECLAIM Repository; Fault Diagnosis and Predictive Maintenance Simulation Engine using Digital Twin (T3.3, LINKS)
Output Connections & Interfaces:	Fault Diagnosis and Predictive Maintenance Simulation Engine using Digital Twin (T3.3, LINKS)
Functional Requirements:	REC-3, REC-7, REC-8
Non-Functional Requirements:	REC-12
Input parameters (attribute/parameter, short description, data type, data format, value range & frequency, origin):	Normal process behavior Short description: normal process behavior in order to train an algorithm with such a pattern and further classification of data into normal / abnormal Data type: dictionary/list Data format: JSON Value range & frequency: - Data received from: database
Output parameters (attribute/parameter, short description, data type, data format, value range & frequency, destination):	Data classification Short description: classification into normal / abnormal data Data type: - Data format: - Value range & frequency: - Data sent to: end users
Software Requirements / Development Language:	Python 3.6

Communications:	RECLAIM Repository
Status of the development of the component:	Partially Developed

3.1.8.3.3.3 Predictive maintenance

Table 14 - Filled-in software component template for the Predictive Maintenance

Component Type:	Software
Functionality:	The predictive maintenance approach is composed of 1) a component failure prediction in the future (e.g. 48h); 2) Optimization module for scheduling future maintenance actions based on the existing scheduling. The main idea of this method is to predict what kind of maintenance will be required based on the failing component in the machine. With this, it will be possible to understand what changes need to be done in order to compensate the downtime of the failing machine.
Input Connections & Interfaces:	RECLAIM Repository; Fault Diagnosis and Predictive Maintenance Simulation Engine using Digital Twin (T3.3, LINKS)
Output Connections & Interfaces:	Optimization Toolkit for Refurbishment and Remanufacturing Planning (T3.4, FCY)
Functional Requirements:	REC-3, REC-7, REC-8
Non-Functional Requirements:	REC-12
Input parameters (attribute/parameter, short description, data type, data format, value range & frequency, origin):	1. process data from normal & failure behaviors Short description: process data for both normal and failure behavior in order to train machine learning algorithms to recognize those Data type: dictionary/list Data format: JSON Value range & frequency: - Data received from: database 2. failure components Short description: components that failed during the process for correlation with process data Data type: dictionary/list

	Data format: JSON Value range & frequency: - Data received from: database 3. existing maintenance activities Short description: scheduling of the maintenance activities for further scheduling optimization in case of component failure Data type: dictionary/list Data format: JSON Value range & frequency: - Data received from: database
Output parameters (attribute/parameter, short description, data type, data format, value range & frequency, destination):	Optimized maintenance schedule Short description: new proposal for maintenance activities towards cost minimization Data type: - Data format: - Value range & frequency: - Data sent to: end users
Software Requirements / Development Language:	Python 3.6
Communications:	RECLAIM Repository
Status of the development of the component:	Partially Developed

3.1.8.3.4 Optimization Toolkit for Refurbishment and Remanufacturing Planning

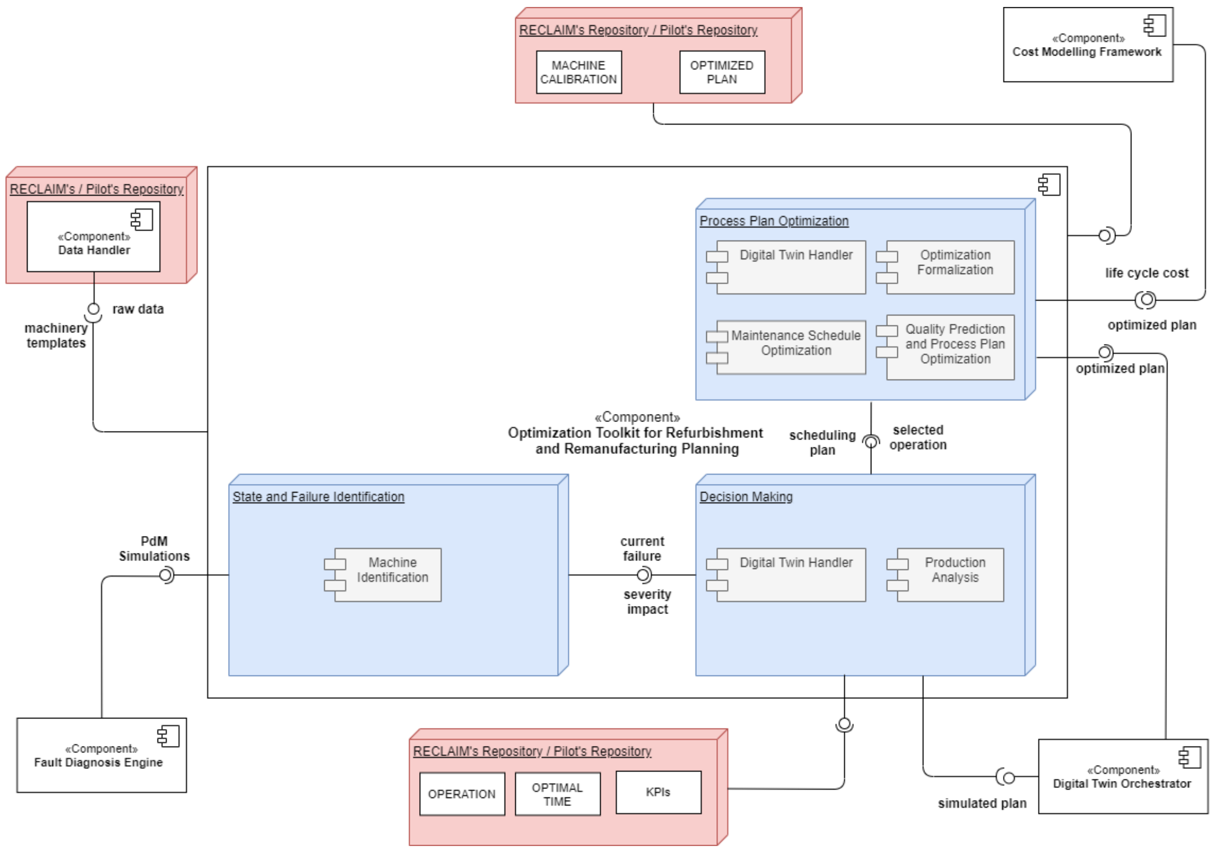

RECLAIM's Repository / Pilot's Repository
MACHINE CALIBRATION
OPTIMIZED PLAN
«Component» Cost Modelling Framework
RECLAIM's / Pilot's Repository
«Component» Data Handler
raw data
machinery templates
Process Plan Optimization
Digital Twin Handler
Optimization Formalization
Maintenance Schedule Optimization
Quality Prediction and Process Plan Optimization
life cycle cost
optimized plan
optimized plan
«Component» Optimization Toolkit for Refurbishment and Remanufacturing Planning
scheduling plan
selected operation
State and Failure Identification
Machine Identification
current failure
severity impact
Decision Making
Digital Twin Handler
Production Analysis
PdM Simulations
«Component» Fault Diagnosis Engine
RECLAIM's Repository / Pilot's Repository
OPERATION
OPTIMAL TIME
KPIs
simulated plan
«Component» Digital Twin Orchestrator

Figure 17 - Optimization Toolkit for Refurbishment and Remanufacturing Planning component diagram

Table 15 - Filled-in software component template for the Optimization Toolkit for Refurbishment and Remanufacturing Planning

Component Type:	Software
Functionality:	The Optimization Toolkit for Refurbishment and Remanufacturing Planning (T3.4, building block 7) aims to support planning optimization through multi-variable monitoring of the machines' operational parameters where the effects of variable changes will be possible to determine and combine learning methodologies for model-based plat-site/shop-floor control. Based on the multimodal data provided by the IoT infrastructure, new approaches of real-time production planning optimization algorithms will be developed to apply proven optimization methodologies to deliver measurable performance improvements. Also, this monitoring takes into consideration the data collected from the sensors network (Building Block 1), the machine profiles, production processes, and previous predictive maintenance simulations (Building Block 6). This toolkit is composed of three different blocks: State and Failure Identification, Decision Making, and Process Plan

	Optimization. The first block intends to identify the current state of the machine and characterize it according to its state and impact on the machine. This block will receive the trigger of a predicted failure from Digital Twin (T3.3) and will interpreter according to it within the Machine Identification module. The Decision-Making block receives the Severity Range from the previous block and evaluates its impact on the machine operation and performance. With the addition of the machinery profile, this block will propose and characterize different operations to the machine. Four different operations can be returned: "Machine OK" if the machine doesn't need any optimization, "Refurbishment", "Remanufacturing" or "Production Line Modification" according to the more efficient solution for the machine optimization. The final block will associate the proposed solution to the machine process plan and life cycle cost (received from Cost Framework), and generate the Process Planning according to it. It is expected that this plan considers all the different options and chooses the optimized solution according to machine performance and production. The proposed plan can be validated with simulation engines. Besides, a machine calibration proposal should be reported as a suggestion to parameter change.
Input Connections & Interfaces:	RECLAIM/Pilot Repository (Digital Retrofitting API, Machinery Production Processes, and Machinery Operational Profiling), Fault Diagnosis Engine (Predict Maintenance Simulations), Cost Modelling Framework (Life cycle cost of the machine), and Digital Twin Orchestrator (Simulated Plan)
Output Connections & Interfaces:	RECLAIM/Pilot Repository (Machine Process Plan and Machine Calibration Plan), Cost Modelling Framework (Optimized Plan), and Digital Twin Orchestrator (Optimized plan)
Functional Requirements:	REC-31, REC-32, REC-33, REC-34, REC-3, REC-4, REC-7, REC-8, REC-9, REC-10
Input parameters (attribute/parameter, short description, data type, data format, value range & frequency, origin):	1. Shop-floor sensors data (historic and real-time) (RECLAIM/Pilot Repositories) Short description: Time-series data with product information from earlier and later dates. Important to estimate some components by interpolation between values (historical and real-time). Data type: data frame with headers Data format: CSV

	Value range & frequency: dependent on time window prediction Data received from: database/file (Data Handler) 2. Machinery operational profiling and models (RECLAIM/Pilot Repositories) Short description: Time-series data with machinery information relevant to create production patterns which will be further used to distinguish normal from abnormal states. Data type: data frame with headers Data format: CSV Value range & frequency: dependent on time window prediction Data received from: database/file 3. Machinery production processes (RECLAIM/Pilot Repositories) Short description: Time-series data with machinery production processes relevant to understand how the different machines work and which failures have an impact on production. Data type: data frame with headers Data format: CSV Value range & frequency: 1 file per process Data received from: database/file 4. Predictive Maintenance Simulations (Fault Diagnosis Engine) Short description: Failure forecasts considering a simulated environment of the machines under analysis. The component will validate its veracity and identify which fault states are correctly identified. Data type: data frame with headers Data format: CSV Value range & frequency: Anything Data received from: database/file 5. Life Cycle Cost (Cost Modelling) Short description: Life Cycle Cost for each machine and its processes. Data type: data frame with headers Data format: CSV Value range & frequency: 1 per machine Data received from: database/file 6. default arguments Short description: configuration file Data type: object Data format: INI Value range & frequency: dependent on argument Data received from: defaults folder 7. non-default arguments Short description: changed default / other Data type: object

	Data format: command-line options Value range & frequency: dependent on argument Data received from: user / Docker image
Output parameters (attribute/parameter, short description, data type, data format, value range & frequency, destination):	1. Machine Process Plan Short description: Generated process plans regarding a specific machine. These plans will be an optimized version considering the possible alternatives, in order to obtain the Best Solution in the next interaction. Data type: data frame with headers Data format: CSV Value range & frequency: 1 plan per machine Data sent to: RECLAIM/Pilot Repositories 2. Machine Calibration Plan Short description: Generated machine calibration suggestions according to product quality and parameter updates to obtain a better performance. Data type: data frame with headers Data format: CSV Value range & frequency: 1 plan per machine Data sent to: RECLAIM/Pilot Repositories
Status of the development of the component:	To be developed from scratch

3.1.8.3.4.1 Algorithms for quality prediction and process parameter optimization

Table 16 - Filled-in software component template for the algorithms for quality prediction and process parameter optimization

Component Type:	Software
Functionality:	Algorithms to predict the final product quality in a specific machine based on the parameters used for the process. These methods are based in data, where both parameters and final quality should exist in order to train machine learning models, both regression and classification. Ultimately, based on the quality predictive model it is possible to estimate future machine parameters if a new product needs to be yielded.
Input Connections & Interfaces:	RECLAIM Repository; Fault Diagnosis and Predictive Maintenance Simulation Engine using Digital Twin (T3.3, LINKS)
Output Connections	Optimization Toolkit for Refurbishment and

& Interfaces:	Remanufacturing Planning (T3.4, FCY)
Functional Requirements:	REC-3, REC-7, REC-8
Input parameters (attribute/parameter, short description, data type, data format, value range & frequency, origin):	1. machine parameters Short description: machine parameters used in previous processes Data type: dictionary/list Data format: JSON Value range & frequency: - Data received from: database 2. product final quality / specifications Short description: product final quality in previous processes Data type: dictionary/list Data format: JSON Value range & frequency: - Data received from: database 3. target quality / specifications Short description: final desired quality / specifications of the new process to be made Data type: dictionary/list Data format: JSON Value range & frequency: - Data received from: database
Output parameters (attribute/parameter, short description, data type, data format, value range & frequency, destination):	Optimized machine parameters Short description: machine parameters to be used by the operator for the new process Data type: - Data format: - Value range & frequency: - Data sent to: end users
Software Requirements / Development Language:	Python 3.6
Communications:	RECLAIM Repository
Status of the development of the component:	Partially Developed

3.1.8.3.5 Prognostic and Health Management Toolkit

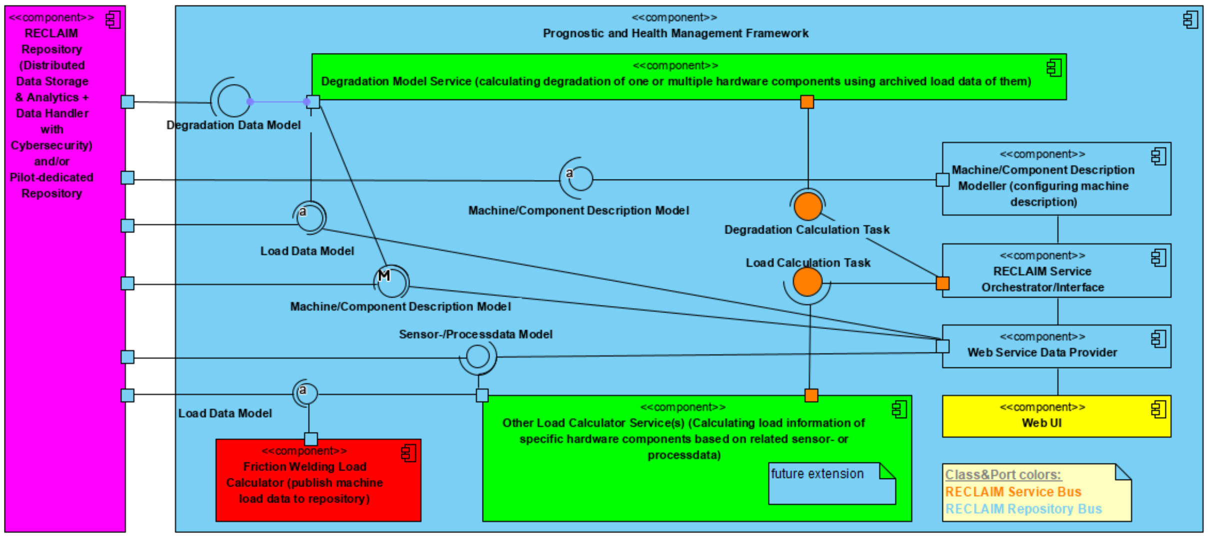

<<component>>
RECLAIM Repository (Distributed Data Storage & Analytics + Data Handler with Cybersecurity) and/or Pilot-dedicated Repository
<<component>>
Prognostic and Health Management Framework
<<component>>
Degradation Model Service (calculating degradation of one or multiple hardware components using archived load data of them)
Degradation Data Model
Machine/Component Description Model
Load Data Model
Machine/Component Description Model
Sensor-/Processdata Model
Load Data Model
Degradation Calculation Task
Load Calculation Task
<<component>>
Machine/Component Description Modeller (configuring machine description)
<<component>>
RECLAIM Service Orchestrator/Interface
<<component>>
Web Service Data Provider
<<component>>
Web UI
<<component>>
Friction Welding Load Calculator (publish machine load data to repository)
<<component>>
Other Load Calculator Service(s) (Calculating load information of specific hardware components based on related sensor- or processdata)
future extension
Class&Port colors:
RECLAIM Service Bus
RECLAIM Repository Bus

Figure 18 - Prognostic and Health Management Toolkit component diagram

Table 17 - Filled-in software component template for the Prognostic and Health Management Toolkit

Component Type:	Software
Functionality:	The aim of this toolkit is the development of a comprehensive framework for predictive and preventive control and management. The framework will contain a number of physical and virtual tools to be integrated for diagnostics and prognostics in manufacturing in order to provide a significant contribution on enhancing operations and maintenance intelligence. Exposed Services: - Degradation model service. - Machine Component Description Modeller Private Services: - FWM Load Calculator - Web-UI - Web Service Data Provider - RECLAIM Orchestrator
Input Connections & Interfaces:	RECLAIM Repository (Degradation Data, Load Data, Machine/Component Description, Sensor/Process Data)
Output Connections & Interfaces:	RECLAIM Repository (Degradation Data, Load Data, Machine/Component Description, Sensor/Process Data)
Functional	REC-23, REC-35, REC-37, REC-48, REC-49, REC-50, REC-51,

Requirements:	REC-52
Non-Functional Requirements:	REC-13, REC-36, REC-53, REC-54
Input parameters (attribute/parameter, short description, data type, data format, value range & frequency, origin):	machine part/component description, sensor/process data useful for load calculation, load data, degradation data (see data models in section A3.5 Data from the Prognostic and Health Management Toolkit)
Output parameters (attribute/parameter, short description, data type, data format, value range & frequency, destination):	machine part/component description, sensor/process data useful for load calculation, load data, degradation data (see data models in section A3.5 Data from the Prognostic and Health Management Toolkit)
Software Requirements / Development Language:	Javascript / C++
Hardware Requirements:	typical windows PC
Communications:	Ethernet / OPC-UA to machine
Status of the development of the component:	To be developed from scratch

3.1.8.3.5.1 Degradation models

Table 18 - Filled-in software component template for the Degradation models

Component Type:	Software
Functionality:	Machine-learning-based models to model the degradation patterns of certain components, and based on this, calculate the KPIs to classify the machine's health. The key difference between this technique and T2.5 is mainly on the detail they will provide, being the T2.5 much more high-level and only considering small aspects of the machine, like MTBF and similar. This algorithm will take into account

	different dimensions like 1) Machine Load; 2) Time and 3) Sensor Data. Based on this, the proposed algorithm will provide more insights into machine degradation than T2.5. This method is very related with the one presented by HWH, and we will work together with HWH on the implementation and development of this method, both specifically for HWH and generic for others to use.
Input Connections & Interfaces:	RECLAIM service orchestrator / interface; Web Service data provider; IoT Gateway with AI acceleration (T3.1, FINT); RECLAIM Repository.
Output Connections & Interfaces:	RECLAIM service orchestrator / interface; Web Service data provider;
Functional Requirements:	REC-3, REC-7, REC-8
Input parameters (attribute/parameter, short description, data type, data format, value range & frequency, origin):	1. process data Short description: historical process data Data type: dictionary/list Data format: JSON Value range & frequency: - Data received from: database 2. historical data of failures Short description: list of annotated failures through time Data type: dictionary/list Data format: JSON Value range & frequency: - Data received from: database 3. product quality Short description: final quality of the product Data type: dictionary/list Data format: JSON Value range & frequency: - Data received from: database 4. machine parameters used Short description: historical machine parameters Data type: dictionary/list Data format: JSON Value range & frequency: - Data received from: database
Output parameters (attribute/parameter, short description, data type, data format, value range & frequency,	Prediction machine degradation Short description: Models to predict different failures in the future according to different processes and parameters used Data type: - Data format: - Value range & frequency: -

destination):	Data sent to: end users
Software Requirements / Development Language:	Python 3.6
Communications:	RECLAIM Repository
Status of the development of the component:	Partially Developed

3.1.8.3.6 Cost Modelling and Financial Analysis Toolkit

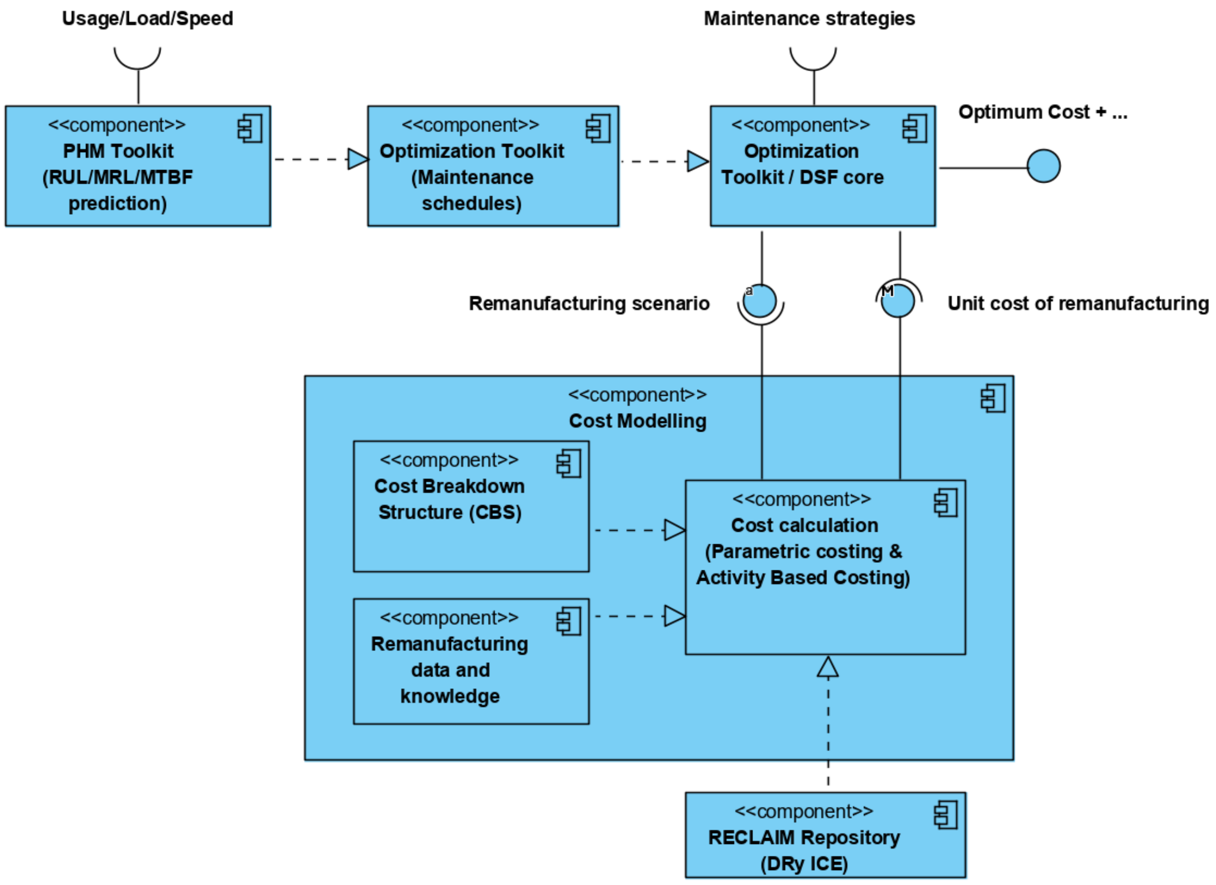

Usage/Load/Speed
Maintenance strategies
<<component>> PHM Toolkit (RUL/MRL/MTBF prediction)
<<component>> Optimization Toolkit (Maintenance schedules)
<<component>> Optimization Toolkit / DSF core
Optimum Cost + ...
Remanufacturing scenario
Unit cost of remanufacturing
<<component>> Cost Modelling
<<component>> Cost Breakdown Structure (CBS)
<<component>> Cost calculation (Parametric costing & Activity Based Costing)
<<component>> Remanufacturing data and knowledge
<<component>> RECLAIM Repository (DRy ICE)

Figure 19 - Cost Modelling and Financial Analysis Toolkit component diagram

Table 19 - Filled-in software component template for the Cost Modelling and Financial Analysis Toolkit

Component Type:	Software
Functionality:	The component (cost model) will estimate the unit cost of remanufacturing, refurbishment, repair strategies for

	equipment or their systems/sub-systems/components, which will support the Decision Support Framework (T4.4).
Input Connections & Interfaces:	Associated remanufacturing scenarios (e.g. what equipment/sybsystem/component, what remanufacturing strategy, when to remanufacture) from pilot partners directly or via T4.4 DSF
Output Connections & Interfaces:	Unit cost of a remanufacturing strategy.
Functional Requirements:	REC-31

Input parameters (attribute/parameter, short description, data type, data format, value range & frequency, origin):	What equipment/sybsystem/component;
	
	What remanufacturing strategy;
	
	When cost needs to be estimated for.

Output parameters (attribute/parameter, short description, data type, data format, value range & frequency, destination):	Unit cost of reamanufacturing, digit number
Software Requirements / Development Language:	Excel, C

3.1.8.3.7 Integrated Decision Support Framework (DSF) for Refurbishment & Remanufacturing Optimization - core component

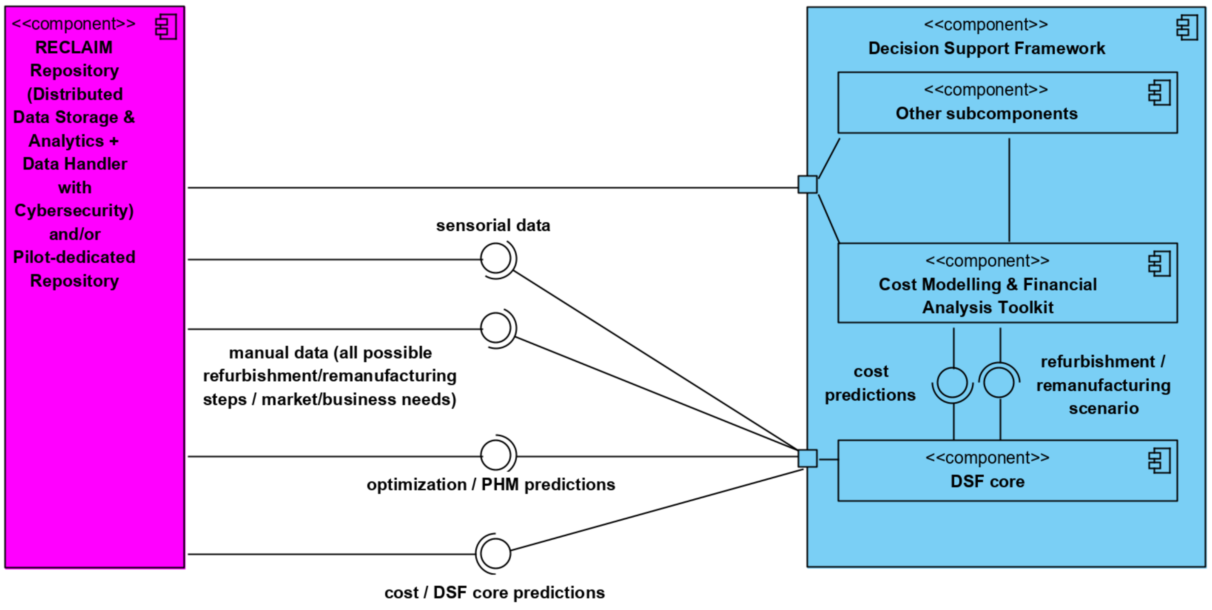

<<component>>
RECLAIM Repository (Distributed Data Storage & Analytics + Data Handler with Cybersecurity) and/or Pilot-dedicated Repository
<<component>>
Decision Support Framework
<<component>>
Other subcomponents
<<component>>
Cost Modelling & Financial Analysis Toolkit
<<component>>
DSF core
sensorial data
manual data (all possible refurbishment/remanufacturing steps / market/business needs)
optimization / PHM predictions
cost / DSF core predictions
cost predictions
refurbishment / remanufacturing scenario

Figure 20 - Integrated Decision Support Framework (DSF) for Refurbishment & Remanufacturing Optimization component diagram

Table 20 - Filled-in software component template for the Integrated Decision Support Framework (DSF) for Refurbishment & Remanufacturing Optimization

Component Type:	Software
Functionality:	Based on evaluation metrics to be defined, raw data from T3.1, the output of data analysis components from T3.2-T3.4, T4.2 and T4.3, as well as lifetime extension strategies from T4.1, the DSF will infer 1) the most suitable remanufacturing/refurbishment strategy, 2) the preferable timeframe for the implementation of the strategy, 3) the right components to be remanufactured/refurbished, 4) the optimal design alternative. In contrast with the Optimization Toolkit of T3.4, which performs only operational optimization in single machines, T4.4 performs operational optimization globally, i.e. in whole production lines or set of machines of each pilot use case. This component will probably use the REST protocol for communicating both with the repository and the Cost Modelling and Financial Analysis Toolkit.
Input Connections & Interfaces:	RECLAIM Repository (Digital Retrofitting API), Cost Modelling and Financial Analysis Toolkit
Output Connections	RECLAIM Repository (Digital Retrofitting API), Cost Modelling

& Interfaces:	and Financial Analysis Toolkit
Functional Requirements:	REC-3, REC-7, REC-8, REC-9, REC-10, REC-31, REC-50
Non-Functional Requirements:	REC-12, REC-13, REC-14
Input parameters (attribute/parameter, short description, data type, data format, value range & frequency, origin):	1. raw data Short description: sensorial/manual data Data type: dictionary/list Data format: JSON Value range & frequency: - Data received from: database 2. PHM predictions Short description: - Data type: dictionary/list Data format: JSON Value range & frequency: - Data received from: database 3. Cost predictions Short description: - Data type: dictionary/list Data format: JSON Value range & frequency: - Data received from: Cost Modelling and Financial Analysis Toolkit 4. T3.4 optimization predictions Short description: - Data type: dictionary/list

	Data format: JSON Value range & frequency: - Data received from: database
Output parameters (attribute/parameter, short description, data type, data format, value range & frequency, destination):	1. Refurbishment/remanufacturing scenario Short description: scenario for which the KPIs of this component will be computed Data type: dictionary/list Data format: JSON Value range & frequency: - Data sent to: Cost Modelling and Financial Analysis Toolkit 2. DSF predictions Short description: - Data type: dictionary/list Data format: JSON Value range & frequency: - Data sent to: database 3. Cost predictions Short description: optimal cost and cost in case that no action is taken Data type: dictionary/list Data format: JSON Value range & frequency: - Data sent to: database
Software Requirements / Development Language:	GitLab / Python 3.6
Status of the development of the component:	To be developed from scratch

3.1.8.4 In-Situ Repair Data Analytics

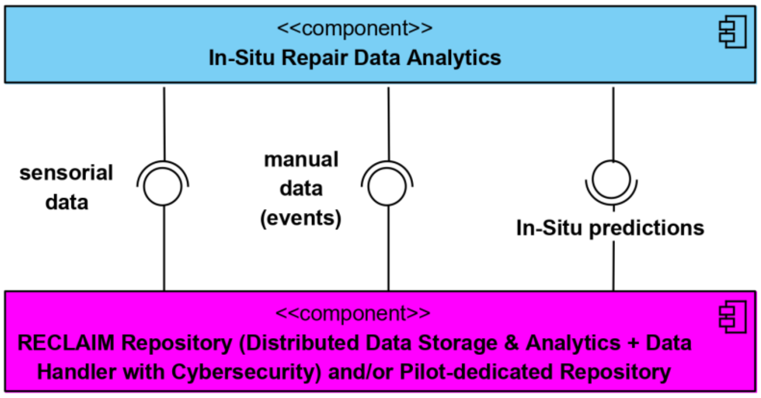

<<component>>
In-Situ Repair Data Analytics
sensorial data
manual data (events)
In-Situ predictions
<<component>>
RECLAIM Repository (Distributed Data Storage & Analytics + Data Handler with Cybersecurity) and/or Pilot-dedicated Repository

Figure 21 - In-Situ Repair Data Analytics component diagram

Table 21 - Filled-in software component template for the In-Situ Repair Data Analytics

Component Type:	Software
Functionality:	This is the component corresponding to T5.2 and building block 8. The exact role of it will depend on the pilot needs. In any case, it will consist of algorithms and visual analytics. One possible option is that a camera or laser sensor that will be taking 2D or 3D data from the product is installed, and an image processing algorithm (supervised or unsupervised, depending on the presence or absence of ground truth data respectively) will be comparing it with the ideal form of the product and based on that will be inferring (in the supervised case) what action should be taken on the equipment producing it. Another probable option is to use process data from machinery data collectors as input and perform a signal processing algorithm (e.g. Fourier analysis) to monitor the state of machine components producing the products.
Input Connections & Interfaces:	RECLAIM Repository (Digital Retrofitting API)
Output Connections & Interfaces:	RECLAIM Repository (Digital Retrofitting API)
Functional Requirements:	REC-3, REC-4, REC-7, REC-8, REC-9, REC-10, REC-24, REC-50
Non-Functional Requirements:	REC-12, REC-13, REC-14
Input parameters (attribute/parameter,	1. model input time series data (historical/real-time) Short description: 3D data of the product, or process data

short description, data type, data format, value range & frequency, origin):	Data type: dictionary / list / data frame of floats Data format: JSON/CSV Value range & frequency: anything Data received from: database/file 2. model target time series data (historical/real-time) Short description: event data (refurbishment etc.) Data type: dictionary / list / data frame of floats Data format: JSON/CSV Value range & frequency: anything Data received from: database/file 3. statistical / machine learning models Short description: descriptive/predictive analytics Data type: - Data format: - Value range & frequency: - Data received from: models folder
Output parameters (attribute/parameter, short description, data type, data format, value range & frequency, destination):	1. statistical / machine learning models Short description: descriptive/predictive analytics Data type: - Data format: - Value range & frequency: - Data sent to: models folder 2. predictions Short description: descriptive/predictive/prescriptive analytics Data type: - Data format: - Value range & frequency: - Data sent to: database/file
Status of the development of the component:	Partially Developed

3.1.8.5 AR mechanisms

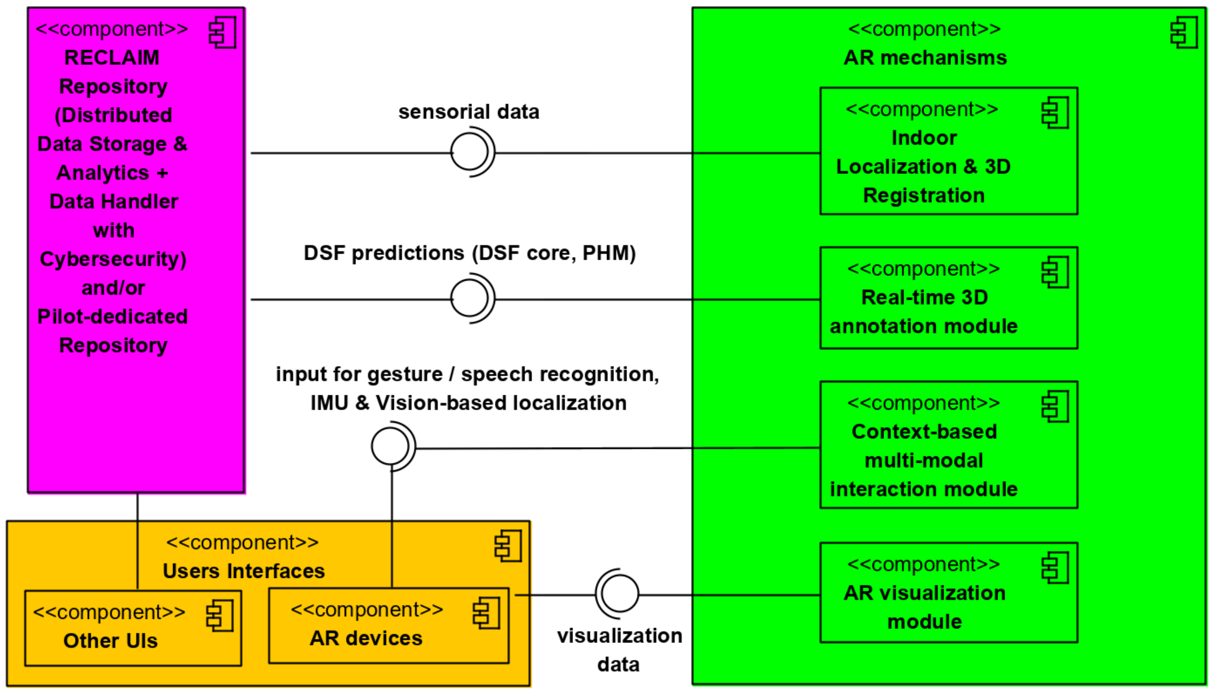

<<component>>
RECLAIM Repository (Distributed Data Storage & Analytics + Data Handler with Cybersecurity) and/or Pilot-dedicated Repository
sensorial data
DSF predictions (DSF core, PHM)
input for gesture / speech recognition, IMU & Vision-based localization
visualization data
<<component>>
AR mechanisms
<<component>>
Indoor Localization & 3D Registration
<<component>>
Real-time 3D annotation module
<<component>>
Context-based multi-modal interaction module
<<component>>
AR visualization module
<<component>>
Users Interfaces
<<component>>
Other UIs
<<component>>
AR devices

Figure 22 - AR mechanisms component diagram

Table 22 - Filled-in software component template for the AR mechanisms

Component Type:	Software
Functionality:	1. Develop novel Localization Mechanism: a) Research Inside-out 3D registration [based on Inertial Measurement Unit (IMU) and Vision hybrid methods] b) Research Outside-in 3D registration (based on Bluetooth/WiFi/Zigbee etc. triangulation) c) Implement hybrid indoor localization using above methodologies 2. Develop novel AR Visualization Mechanism: a) Implement head-mounted display (HMD) and Mobile device-based AR Visualization pipeline b) Integrate Indoors localization to AR 3D visualization c) Implement real-time Localised 3D Annotation module d) Research multimodal (gesture+voice) interaction techniques 3. Integrate with Adaptive Sensorial Network: Use the ASN to provide context-aware instructions and notifications 4. Integrate with DSF: Integrate DSF proposed solutions in instructions and notifications
Input Connections &	RECLAIM Repository (Digital Retrofitting API), AR devices

Interfaces:	(AR UI)
Output Connections & Interfaces:	RECLAIM Repository (Digital Retrofitting API), AR devices (AR UI)
Functional Requirements:	REC-11, REC-15, REC-16, REC-17, REC-18, REC-19, REC-20, REC-21, REC-22
Non-Functional Requirements:	REC-12, REC-13, REC-14
Input parameters (attribute/parameter, short description, data type, data format, value range & frequency, origin):	1. machinery health data Short description: - Data type: - Data format: - Value range & frequency: - Data received from: database 2. proposed actions / parts IDs (from DSF) Short description: - Data type: - Data format: - Value range & frequency: - Data received from: database 3. network-based localization Short description: - Data type: - Data format: - Value range & frequency: - Data received from: database 4. IMU & vision-based localization Short description: IMU = Inertial Measurement Unit Data type: - Data format: - Value range & frequency: - Data received from: AR devices 5. input for gesture / speech recognition Short description: - Data type: - Data format: - Value range & frequency: - Data received from: AR devices 6. gesture / speech recognition model Short description: - Data type: - Data format: - Value range & frequency: - Data received from: database
Output parameters	1. textual recommendations

(attribute/parameter, short description, data type, data format, value range & frequency, destination):	Short description: steps of optimal actions Data type: - Data format: - Value range & frequency: - Data sent to: end users 2. coordinates/orientation Short description: coordinates/orientation of user & components Data type: 3D vectors of floats Data format: JSON Value range & frequency: |point|<105, |pose|<180 Data sent to: end users
Software Requirements / Development Language:	Python, Keras, OpenCV, C#, Unity3D
Hardware Requirements:	Modern computer with 16GB RAM and with a dedicated graphics card, AR Smart Glasses with RGB-D camera and all the necessary sensors
Status of the development of the component:	Partially Developed

3.1.8.6 Life Cycle Assessment

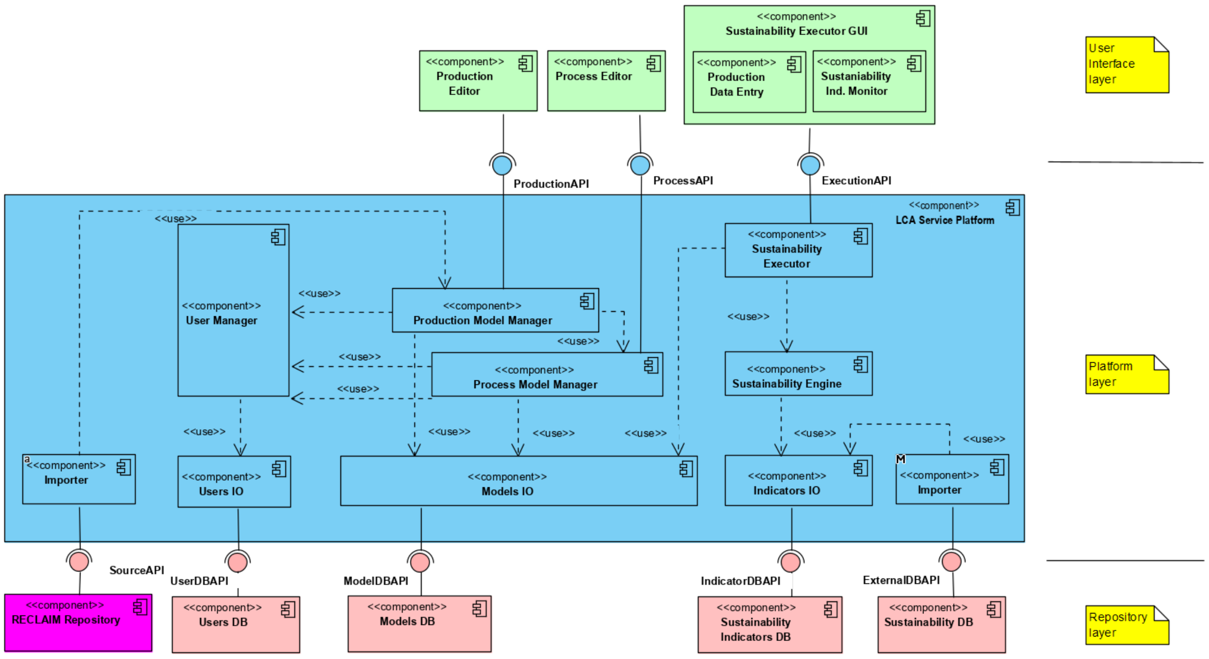


Figure 23 - Life Cycle Assessment component diagram

Table 23 - Filled-in software component template for the Life Cycle Assessment

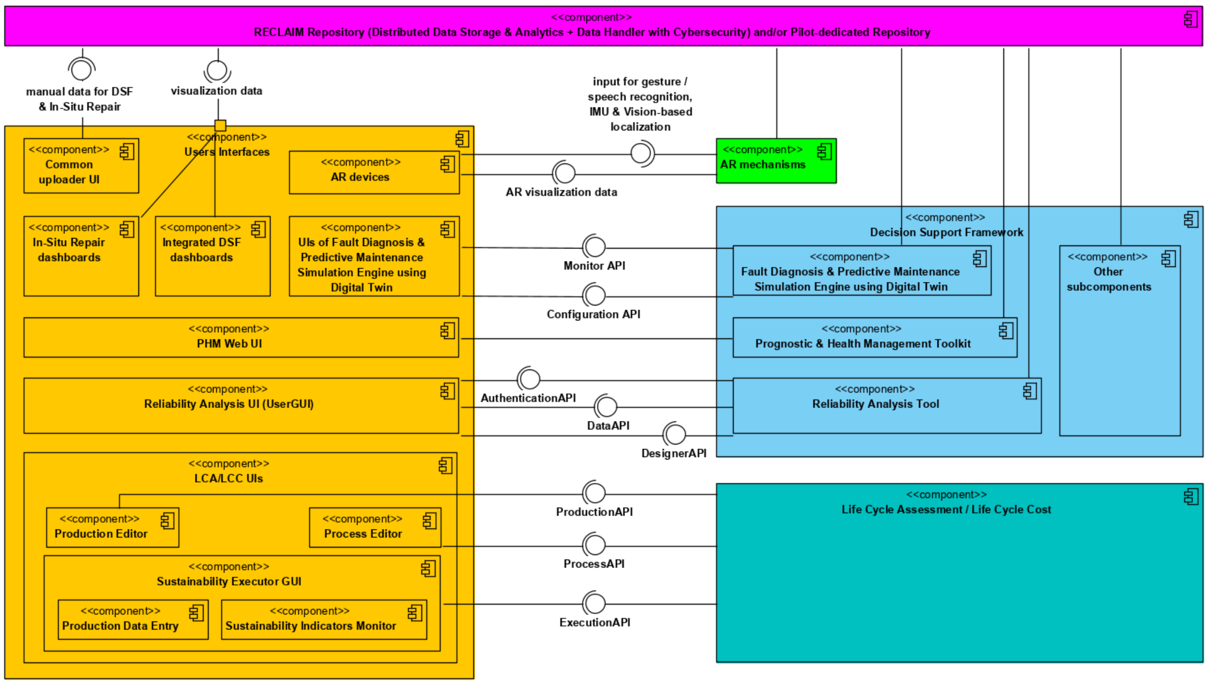
Component Type	Web-based application
Functionality	The main functionalities of the Life Cycle Assessment Tool and its dashboard are the following: • real-time assessment of the sustainability performances • generation of machine use/refurbishment scenarios • comparison of the identified scenarios • visualization of assessment results The tool is composed by the following subcomponents, divided in three layers: repository, LCA platform, user interface. Repository layer • Users DB: the component stores the data related to users accessing the platform and makes them available to the User Manager whenever necessary; • Models DB: the component is deputed to store the models of processes, production systems or supply chains designed inside the platform. The Models DB exchanges data by means of the Models IO to the related Model Managers within the LCA platform. • Sustainability indicators DB: this component is dedicated to the storage of the KPIs and/or methodologies that are used by the Sustainability Engine to calculate the environmental, social and economic LCA impacts; • Sustainability DB: this component is dedicated to the integration and storage of external databases necessary for LCA impacts computation. LCA service platform • User Manager: it manages the profiles of the users of the platform • Process Model Manager, Production Model Manager: these two components are dedicated to the design and editing of respectively production processes (e.g. milling) or group of processes linked to produce a component (e.g. production line); • User IO, Models IO, Indicators IO, Importers: all these component are dedicated to the dispatching to the other internal components of the LCA platform of data coming from external sources or DBs. • Sustainability engine: it is the core of the platform and stores the algorithms required to calculate the LCA impacts; • Sustainability executor: it uses the defined process/production models and the calculation logics embedded in the sustainability engine to provide as output the impacts of each analyzed production/process.

	User interface layer • Process Editor: this is the interface that allows the user to edit a new process; • Production editor: this is the interface that allows the user to interlink a set of processes in a production system; • Sustainability executor GUI: this is the main user interface that provides the visualization means of the calculations performed by the platform. It is composed by two subcomponents: production data entry and Sustainability ind. Monitor. The first is dedicated to the data entry of process/production specific data, required to feed the LCA platform engine. The latter visualizes the sustainability related performances of the system and allows the user to work on the provided data.
Input Connections & Interfaces	RECLAIM Repository
Output Connections & Interfaces	Production Editor (ProductionAPI), Process Editor (ProcessAPI), Sustainability Executor GUI (ExecutionAPI)
Functional Requirements	REC-6, REC-38, REC-39, REC-40, REC-41, REC-42, REC-43, REC-45, REC-46
Input parameters (attribute/parameter, short description, data type, data format, value range & frequency, origin)	1. raw data Short description: Inventory data coming from machines running. These could be already embedded into digital twin models. Data type: dictionary (impact key, impact value, unit of measure) Data format: JSON Value range & frequency: Data ranges are different for each impact, but already validated as coming from trusted data source Data received from: external, production machines 2. Life-Cycle predictions Short description: - Data type: - Data format: JSON Value range & frequency: - Data received from: database 3. Digital Twin predictions Short description: Predicted data coming from digital models Data type: - Data format: JSON Value range & frequency: - Data received from: database

	4. Other data Short description: Predicted data coming from optimization models Data type: - Data format: JSON Value range & frequency: - Data received from: database
Output parameters (attribute/parameter, short description, data type, data format, value range & frequency, destination)	Visualized data Short description: LCA Dashboard Data type: - Data format: - Value range & frequency: - Data sent to: end users
Software Requirements / Development Language	Web based application (Spring framework for back-end and HTML/CSS for front-end) developed by means of Java.
Hardware Requirements	Server to host the docker image.
Communications	Still to be defined, really dependent on the hosting server.
Status of the development of the component	Partially Developed

3.1.8.7 Users Interfaces for the Integrated Decision Support Framework, the In-Situ Repair Data Analytics and the AR mechanisms

The component diagram below shows all envisioned Users Interfaces.



<<component>> RECLAIM Repository (Distributed Data Storage & Analytics + Data Handler with Cybersecurity) and/or Pilot-dedicated Repository
manual data for DSF & In-Situ Repair
visualization data
input for gesture / speech recognition, IMU & Vision-based localization
<<component>> Users Interfaces
<<component>> Common uploader UI
<<component>> AR devices
<<component>> AR mechanisms
AR visualization data
<<component>> In-Situ Repair dashboards
<<component>> Integrated DSF dashboards
<<component>> UIs of Fault Diagnosis & Predictive Maintenance Simulation Engine using Digital Twin
<<component>> Decision Support Framework
<<component>> Fault Diagnosis & Predictive Maintenance Simulation Engine using Digital Twin
<<component>> Other subcomponents
Monitor API
Configuration API
<<component>> PHM Web UI
<<component>> Prognostic & Health Management Toolkit
<<component>> Reliability Analysis UI (UserGUI)
AuthenticationAPI
DataAPI
DesignerAPI
<<component>> Reliability Analysis Tool
<<component>> LCA/LCC UIs
<<component>> Production Editor
<<component>> Process Editor
<<component>> Sustainability Executor GUI
<<component>> Production Data Entry
<<component>> Sustainability Indicators Monitor
ProductionAPI
ProcessAPI
ExecutionAPI
<<component>> Life Cycle Assessment / Life Cycle Cost

Figure 24 - Users Interfaces component diagram

The template below refers only to the UIs for the DSF core, the In-Situ Repair Data Analytics and the AR mechanisms. The others have been described previously in the templates of the respective services/frameworks.

Table 24 - Filled-in software component template for the Users Interfaces for the DSF core, the In-Situ Repair Data Analytics and the AR mechanisms

Component Type:	Software
Functionality:	This component consists of the visualization panels of the RECLAIM platform corresponding to the output of the Decision Support Framework, the In-situ Repair Data Analytics Toolkit and the AR mechanisms, and will appear on relevant hardware (AR devices for AR mechanisms, and usually personal computers for the rest). The component will also allow the end users to provide manual input for the algorithms of the DSF and the In-situ Repair Data Analytics using the Common uploader UI, under the holistic data model supported by the Data Handler. The visualization provided by these UIs is going to be based on Angular 8, which appears to have more capabilities than Grafana, that had been used in previous projects. These UIs will probably support Websocket or also MQTT protocol, but this will also depend on the interfaces of the interacting components.
Input Connections &	RECLAIM Repository (RECLAIM 3rd party API), AR

Interfaces:	mechanisms (AR UI)
Output Connections & Interfaces:	Data Handler with Cybersecurity, AR mechanisms
Functional Requirements:	REC-3, REC-9, REC-10, REC-19, REC-24, REC-48, REC-49, REC-51
Non-Functional Requirements:	REC-12, REC-13, REC-14
Input parameters (attribute/parameter, short description, data type, data format, value range & frequency, origin):	Visualization data Short description: - Data type: dictionary/list Data format: JSON Value range & frequency: - Data received from: RECLAIM Repository, AR mechanisms
Output parameters (attribute/parameter, short description, data type, data format, value range & frequency, destination):	1. manual data for Decision Support Framework and In-Situ Repair Data Analytics Short description: events / specifications / functionalities / experts' estimations / market or business needs / financial knowledge / all possible refurbishment & remanufacturing steps etc. Data type: files Data format: JSON/CSV/XLSX/TXT/XML Value range & frequency: - Data sent to: RECLAIM Repository 2. input for gesture/speech recognition, IMU & Vision-based localization Short description: - Data type: - Data format: - Value range & frequency: - Data sent to: AR mechanisms
Software Requirements / Development Language:	Docker, Angular, Node.js, Apache server / HTML, JavaScript, Python, Java
Hardware Requirements:	computer screens, mobile phones, tablets, AR glasses
Status of the development of the component:	To be developed from scratch

3.1.9 Common information model

As shown by the information view diagram and by the component diagrams of the previous subsections, many components send data to a repository. In case of the RECLAIM Repository, there will be a subcomponent, named “Data Handler with Cybersecurity”, which will transform the data into a common format (“common information model”, or, in other words, “holistic data model”) before sending them to the RECLAIM Repository database (Distributed Data Storage and Analytics) for storage. In order that the common information model (CIM) is created, it was asked by owners of components which send data to the RECLAIM Repository to provide the format of the output of their component as a JSON[3] file. These individual data models may be found in Annex 3: Individual data models.

The CIM created for RECLAIM was based on the OGC SensorThings standard of the Open Geospatial Consortium (OGC) (OGC, 2017), which is one of the data modelling standards used generally in manufacturing and adopted by a recent manufacturing-related EU project (COMPOSITION D3.3, 10/2018). The data are defined in JSON format and an associated object-oriented UML class diagram is provided (OGC SensorThings API Documentation, n.d.). However, the original OGC SensorThings standard was slightly modified in order to fit more the needs of RECLAIM, taking into account the individual data models.

The current version of RECLAIM CIM is shown below.

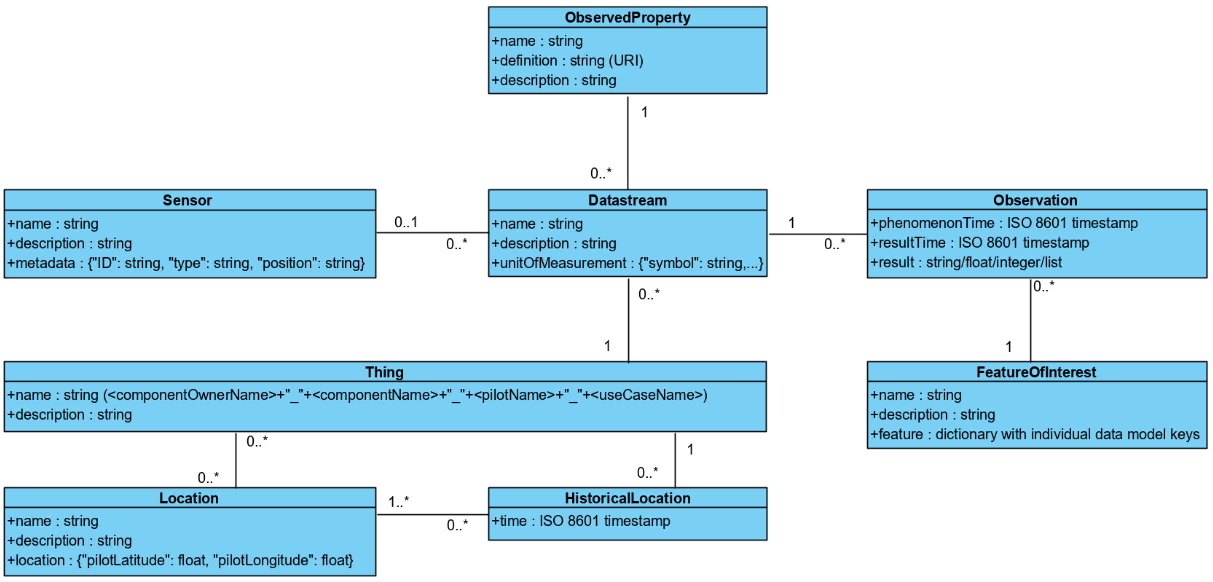

ObservedProperty
+name : string
+definition : string (URI)
+description : string
1
0..*
Sensor
+name : string
+description : string
+metadata : {"ID": string, "type": string, "position": string}
0..1
0..*
Datastream
+name : string
+description : string
+unitOfMeasurement : {"symbol": string,...}
1
0..*
Observation
+phenomenonTime : ISO 8601 timestamp
+resultTime : ISO 8601 timestamp
+result : string/float/integer/list
0..*
0..*
1
1
Thing
+name : string (<componentOwnerName>+"_"+<componentName>+"_"+<pilotName>+"_"+<useCaseName>)
+description : string
FeatureOfInterest
+name : string
+description : string
+feature : dictionary with individual data model keys
0..*
0..*
1
0..*
Location
+name : string
+description : string
+location : {"pilotLatitude": float, "pilotLongitude": float}
1..*
0..*
HistoricalLocation
+time : ISO 8601 timestamp

Figure 25 - UML class diagram of RECLAIM common information model

This CIM will be subject to updates until the next update of this deliverable.

All classes and attributes of this figure also exist in the OGC SensorThings data model and thus they have already been explained by OGC (OGC SensorThings API Documentation, n.d.). On the other hand, some attributes of the OGC SensorThings

[3] The JSON format (as well as XML) is commonly used for data exchange (COMPOSITION D3.3, 10/2018). From the experience of CERTH from previous projects, JSON proved to be more user-friendly compared to XML.

data model were not considered as useful for RECLAIM, so they were omitted. The values of some attributes have been particularized here, so that their potentially necessary content for RECLAIM is clarified. Anyway, all classes and attributes are explained below, so that their specific meaning for RECLAIM is shown:

- *Thing:* This class defines the source of data. It contains the following attributes:
    - *name:* information about the owner of the component that produces the associated data (componentOwnerName), the name of the component (componentName), the name of the pilot organization the data are associated with (pilotName), as well as the pilot use case name (useCaseName)
    - *description:* other potential textual details about this class
- *Location:* This optional class specifies the pilot location. It contains the following attributes:
    - *name:* name of this class
    - *description:* other potential textual details about this class
    - *location:* the latitude (pilotLatitude) and longitude (pilotLongitude) of the pilot premises where the associated data are collected
- *HistoricalLocation:* This optional class provides the following attribute:
    - *time:* time of the current (i.e., last known) or a potentially previous location of a pilot
- *Datastream:* This class represents a time series of a particular variable, the measurements of which are produced by the same Thing. It contains the following attributes:
    - *name:* name of this class
    - *description:* other potential textual details about this class
    - *unitOfMeasurement:* unit of measurement of the measured values; at least the symbol of the unit of measurement must be specified
- *Sensor:* This class represents an instrument that observes a property or phenomenon in order to produce an estimate of its value, as in the SensorThings API. It contains the following attributes:
    - *name:* name of this class
    - *description:* other potential textual details about this class
    - *metadata:* the sensor ID (string), type (string between “GenericSensor” and “VisionSensor”) and position (string), as they have been defined in the individual data models for the ASN sensor and the Machine Vision System (section A3.1 **Data from/to the Adaptive Sensorial Network**)
- *ObservedProperty:* This class defines the variable measured. It contains the following attributes:
    - *name:* name of this class
    - *definition:* definition of the variable provided in a URI link
    - *description:* other potential textual details about this class

- *Observation:* This class contains a measurement of a variable. It contains the following attributes:
    - *phenomenonTime:* the time corresponding to the measured value
    - *resultTime:* the time when this measurement is communicated from the sensor to another component
    - *result:* the value of the measurement
- *FeatureOfInterest:* This class contains the characteristics and other information accompanying a measurement:
    - *name:* name of this class
    - *description:* other potential textual details about this class
    - *feature:* dictionary the keys and values of which are the attribute names and their values respectively; the currently envisioned attributes correspond to the information included in the individual data models that cannot be described in any other class of the data model standard (e.g. machine, decision, message, arguments)

Compared with the OGC SensorThings data model, the multiplicities of the associations are almost the same. In fact, the only change is that in the RECLAIM CIM a datastream does not necessarily have to be associated with a sensor, because RECLAIM data are generated not only by sensors, but also by services/frameworks and the users themselves.

Based on the input received in the component templates, the high-level diagram below summarizes the main communication protocols that are going to be used. This figure is going to be elaborated in the 2nd iteration of T2.3.

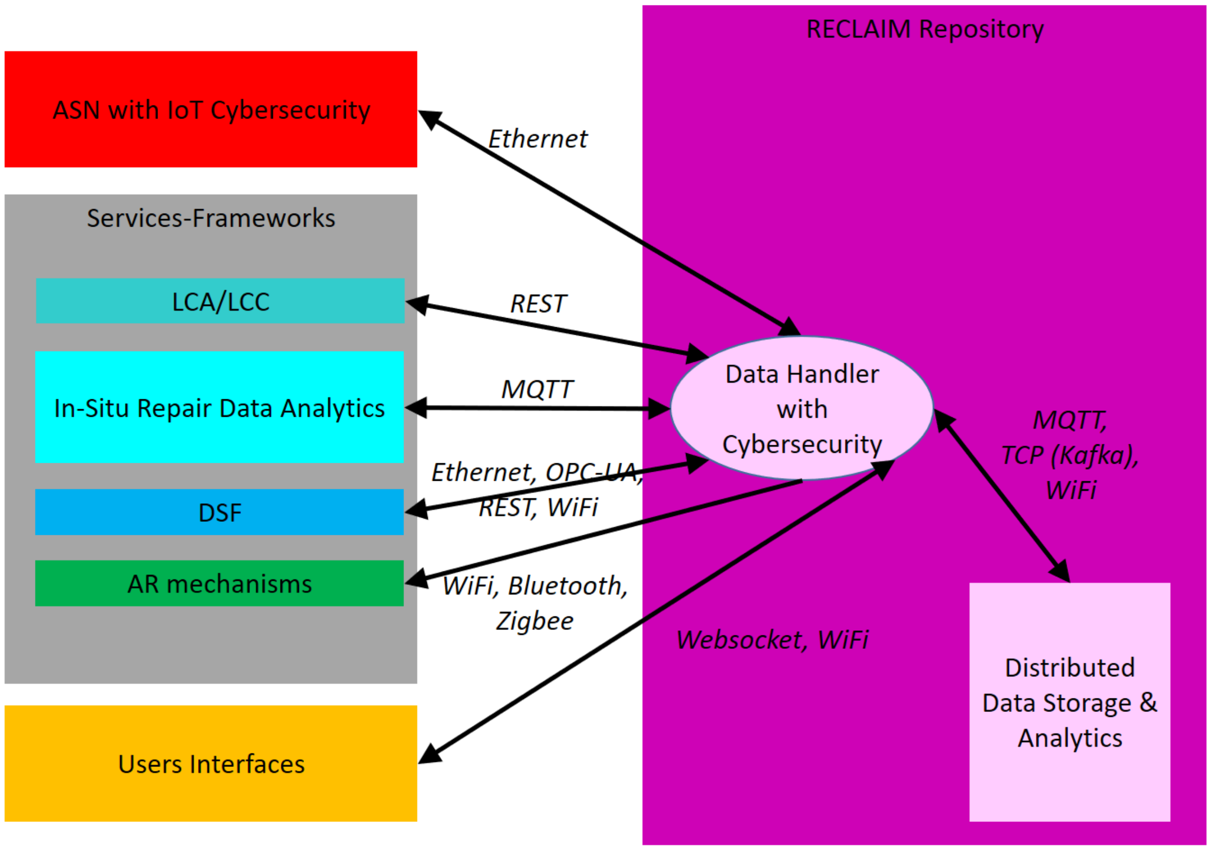


Figure 26 - High-level view of communication protocols to be used in RECLAIM

3.2 Deployment view

The deployment view focuses on aspects of the system which are important after it has been tested and is ready to be used live. By this view, the physical environment in which the system is intended to run is defined, including:

- Required hardware environment (such as processing nodes and network interconnections)
- Technical environment requirements for every node
- Mapping software elements to the runtime environment
- Third-party software requirements
- Network requirements

The deployment view needs to describe the required deployment environment of the RECLAIM platform, which depends on the pilot areas and their topology. This subsection provides a first overview, covering the known hardware requirements of the software modules and the used tools.

The partners who filled in the hardware templates mentioned above contributed to the following deployment view diagram. The corresponding hardware is shown inside the Adaptive Sensorial Network Infrastructure, whereas the main software components correspond to the execution environments of the upper part of the diagram. This diagram is expected to be updated and elaborated in the 2nd version of the architecture, including detailed information about the “Other hardware” to be specified between pilots and hardware providers, and providing more accurate information about the necessary number of devices that will host the RECLAIM components.

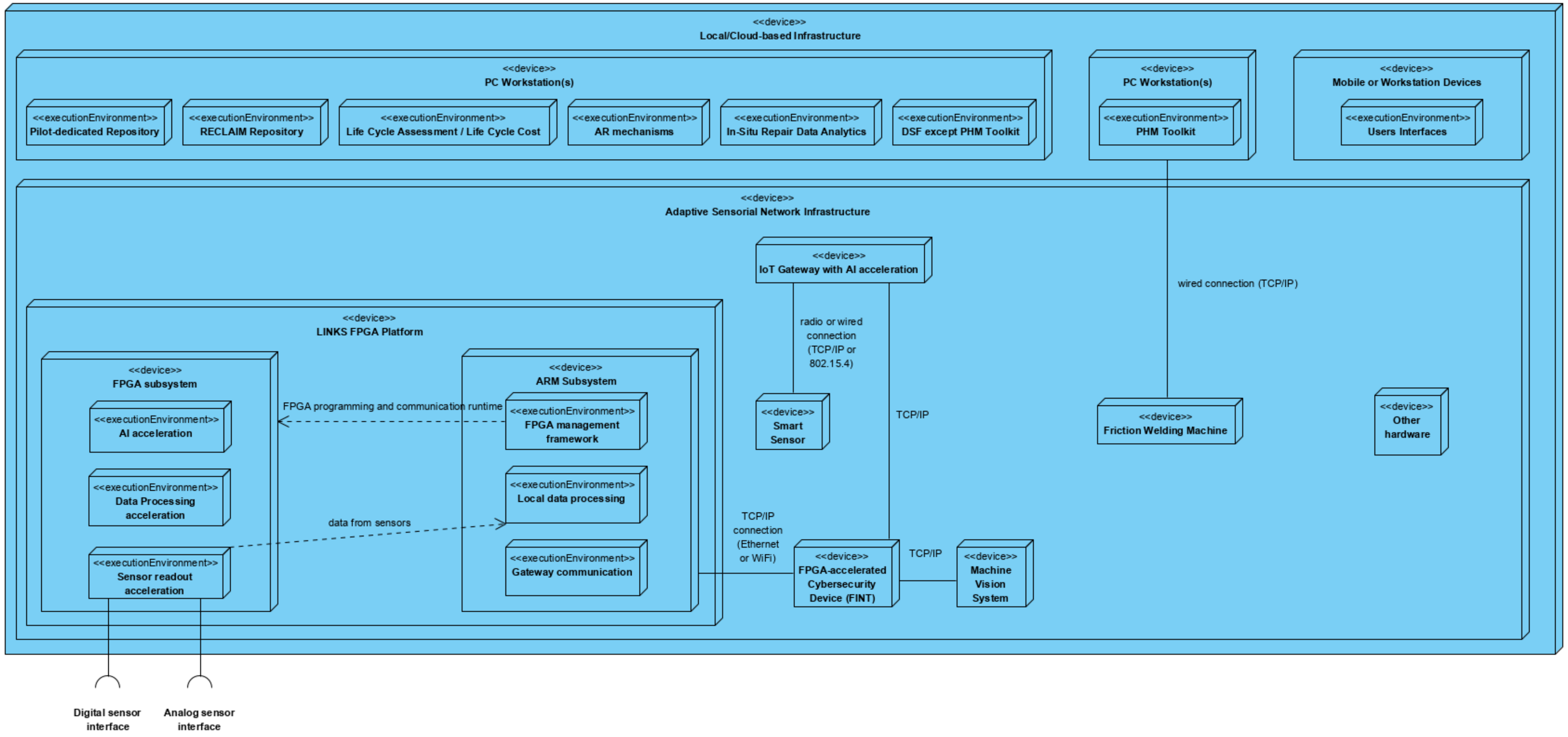


Figure 27 - RECLAIM deployment view

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 869884

3.3 Development view

A considerable amount of planning and design of the development environment is often required to support the design and build of software for complex systems. Some aspects to consider include code structure and dependencies, build and configuration management of deliverables, system-wide design constraints, and system-wide standards to ensure technical integrity. It is the role of the Development view to address these aspects of the system development process.

The remainder of this subsection presents the initial considerations for deployment, whereas related UML diagrams are going to be prepared for a future iteration of the architecture activities.

3.3.1 Code organization

Code organization is a very crucial part in the development stage of the components. Git is an open-source distributed version-control system and is widely used because it offers various tools for software development and other version control tasks.

The main features that make Git so successful are:

- Allows the use of multiple local branches that can be entirely independent of each other.
- Huge speed advantage, as all operations are performed locally.
- Git is distributed, meaning every user has a backup of the main server, thus offering multiple different workflows that can be implemented.
- A staging area makes it possible to choose the modifications of the files to be committed.
- Cryptographic integrity of every bit of data committed.

The source code of all the components is managed in a Git repository server. It is preferable for every change in the code to be committed immediately, so that other developers work on the latest versions of the components.

Regarding code quality, all developers should adhere to coding guidelines and conventions in order to improve the readability and maintenance of the code. Of

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 869884

course, each programming language has its own coding standards and style guides. To automate this process various tools will be utilized, such as Checkstyle for Java and Pylint for Python.

3.3.2 Continuous Integration and Deployment

The definition of Continuous Integration refers to the development practice, where developers integrate code into a shared repository frequently. Automated build and automated tests can then detect errors quickly and locate them more easily. Additionally, Continuous Delivery and Continuous Deployment are common practices used alongside Continuous Integration. They refer to keeping the applications deployable at any point and even pushing them into production automatically.

Some of the benefits of Continuous Integration and Deployment are:

- Reduced time and effort for integrations of different code changes.
- Quick feedback mechanism on every change.
- Earlier and quicker detection of errors.
- Reduced manual testing effort.
- Prevents divergence in different branches as they are integrated regularly.

3.3.3 Tools

Gitlab is chosen as the platform for source code management (SCM), continuous integration (CI), continuous deployment (CD) and monitoring. Gitlab SCM comes with a web-based Git-repository manager, providing wiki, issue-tracking and grouping and sub-grouping or repositories. A crucial benefit of using Gitlab is that it integrates CI services by default in all projects. A CI/CD pipeline can be configured to run in every commit or push to a particular branch. In this way, the need of integration of different tools and pipelines is tackled, as the entire CI pipeline can be configured in one simple script file.

A Gitlab infrastructure will be hosted in CERTH's servers. Each component will be managed in each own repository, by its respective developers. As there are multiple categories of components, such as predictive functions, visualization and deployment tools, all components will be placed inside their corresponding group and sub-group. Developers will be granted permissions and access only to their own repositories.

Docker is an open-source project that utilizes OS-level virtualization to deliver software in packages called containers. These containers are portable, self-sufficient, and isolated from each other. All containers are run by a single operating system kernel, thus using fewer resources than virtual machines. Furthermore, they are designed to run on a wide range of platforms, including Windows, Linux and Mac Desktops, Azure cloud services and Windows, Debian, and Ubuntu servers.

By running the components in isolated containers, the management of the configuration and dependencies of each component becomes very easy. Each container represents a whole separate operating system, with its own software, libraries and configuration installed inside. Moreover, Docker containers are very lightweight and require few resources for their creation. Thus, the integration and deployment of multiple different containerized components in a single host machine bears no effect on the system.

Lastly, Docker provides continuous monitoring and logging of containers, by utilizing various dedicated web management tools, such as Portainer. Depending on its user's permissions, Portainer allows to create, run, publish and remove containers, build and delete Docker images and manage Docker volumes and networks. As a result, it is a versatile tool for both the components developers and the integration and deployment experts.

4. Conclusions

This was the 1st version of the deliverable of the RECLAIM project regarding the architecture specification, submitted in M10 (7/2020). It provided the initial considerations about the architectural viewpoints of the RECLAIM framework (conceptual, communication, functional, deployment, development), as well as a high-level illustrative consolidated view diagram. More especially, the information view was accompanied by the common information model to be used for data exchange with the RECLAIM Repository. Also, numerous system requirements and some hardware specifications were defined in T2.3 and presented in this deliverable.

There will be two revisions of this deliverable in months 20 (5/2021) and 30 (3/2022), as an outcome of the remaining two iterations of the architecture task T2.3 respectively. Apart from updates in the content of this document, the next revised version will include also the consolidated view instantiation per pilot, detailed UML sequence diagrams and the context view derived from them, which will link more clearly the RECLAIM architecture with the technical use cases.

References

1471-2000 - IEEE Recommended Practice for Architectural Description for Software-Intensive Systems. (2000, October 9). IEEE.

B. Schmidt-Belz, D. F. (1999). Scenario-based System Validation by Users. *Human-Computer Interaction - INTERACT*. Edinburgh: Swindon: British Computer Society.

COMPOSITION D3.3. (10/2018). *Digital Factory Model II*.

D2.1. (7/2020). *Initial requirements specification*.

D2.2. (7/2020). *RECLAIM Use Cases Definition & Operational Requirements #1*.

D2.4. (1/2022). *Lessons Learned and updated requirements report*.

D2.6. (3/2021). *RECLAIM Use Cases Definition & Operational Requirements #2*.

D2.7. (3/2022). *RECLAIM Use Cases Definition & Operational Requirements #3*.

ISO/IEC/IEEE 42010: Systems and Software Engineering - Architecture Description. (2011, December 1). Retrieved from http://cabibbo.dia.uniroma3.it/asw/altrui/iso-iec-ieee-42010-2011.pdf

N. Rozanski, E. W. (2012). *Software systems architecture: working with stakeholders using viewpoints and perspectives*. books.google.com.

OGC. (2017). *Open Geospatial Consortium*. Retrieved from http://www.opengeospatial.org

OGC SensorThings API Documentation. (n.d.). Retrieved from http://developers.sensorup.com/docs

S. Robertson, J. R. (1999). *Mastering the requirement process*. London: ACM Press Books.

Sommerville, I. (2011). *Software Engeneering*. Boston: Addison-Wesley.

Annex 1: System requirements list

The system requirements that have been created and successfully passed the quality check follow below. These are the requirements which have been mentioned as satisfied by the software components in the corresponding templates. Only the requirement REC-47 is still open and is going to be assessed in the 2nd iteration, when all requirements will be subject to potential updates, whereas new requirements may be added as well.

Table 25 - Filled-in system requirement templates

[REC-1] The data scientist can use SQL-like queries for structured datasets.	
Requirement Type:	Functional
Reporter:	CERTH

Implementation Assignee:	ICE
Quality Check Assignee:	LINKS
RECLAIM Component(s):	Distributed data storage and analytics (DRy)
Status:	part of specification
Rationale:	In order to avoid inefficient extract/transform/load (ETL) operations over very large dataset, it should be possible to query data without the need to load the data, create and maintain schemas, or transform the data before they can be processed. Query language should be based on well-established SQL standard.
Source:	MONSOON (relevant EU project)
Fit Criterion:	The Distributed Database provides SQL-like interface for structured datasets for "on place" querying in various data formats.

[REC-3] The data scientist has access to a visualization of data stored in the project's repository.	
Issue Links:	**Relates** relates to REC-5 Data originating in the production si... relates to REC-35 Data are homogeneously stored in a ce... relates to REC-37 Transformation of data for their deli... relates to REC-5 Data originating in the production si...
Requirement Type:	Functional
Reporter:	CERTH
Implementation Assignee:	CERTH, HWH, LINKS, SUPSI

Quality Check Assignee:	HWH
RECLAIM Component(s):	Algorithms for quality prediction and process parameter optimization, Anomaly Detection, AR mechanisms, Cost Modelling and Financial Analysis Toolkit, Data handler, Degradation models, Digital Twin for simulation, Distributed data storage and analytics (DRy), Fault Diagnosis and Predictive Maintenance Simulation Engine using Digital Twin, In-Situ Repair Data Analytics, Integrated Decision Support Framework (DSF) for Refurbishment & Remanufacturing Optimization, Life cycle assessment + LCA dashboard, Machine Operational Profiling, Optimization Toolkit for Refurbishment and Remanufacturing Planning, Predictive Maintenance, Prognostic and Health Management Toolkit, Reliability Analysis Tool, Users Interfaces
Status:	part of specification
Rationale:	In order to get an overview of the raw and analyzed data in the project's repository, the data scientist must be able to visualize them in an appropriate manner, such as a dashboard.
Source:	MONSOON (relevant EU project)
Fit Criterion:	Raw, predicted and evaluation data from RECLAIM Repository can be vizualized.
Description:	The leaders of tasks T2.5 (SUPSI), T3.3 (LINKS), T4.2 (HWH), T4.4 (CERTH), T5.2 (CERTH), T5.5 (CERTH) and T7.4 (SUPSI) will most probably develop dedicated Graphical User Interfaces. CERTH may visualize also the output of T3.2 (raw and analyzed data), T3.4 and T4.3 through the GUI for T4.4, if there are no dedicated GUIs for the aforementioned tasks. To be discussed among partners.

[REC-4] The data scientist is able to serialize machine learning models.	
Requirement Type:	Functional
Reporter:	CERTH
Implementation Assignee:	ASTON, CERTH, FCY, FEUP, HWH, ICE, LINKS, SUPSI

Quality Check Assignee:	LINKS
RECLAIM Component(s):	Algorithms for quality prediction and process parameter optimization, Anomaly Detection, AR mechanisms, Cost Modelling and Financial Analysis Toolkit, Degradation models, Fault Diagnosis and Predictive Maintenance Simulation Engine using Digital Twin, In-Situ Repair Data Analytics, Optimization Toolkit for Refurbishment and Remanufacturing Planning, Predictive Maintenance, Prognostic and Health Management Toolkit, Reliability Analysis Tool
Status:	part of specification
Rationale:	Once trained, machine learning models must be serialized and exported in order to be applied to life data from the shop floor.
Source:	MONSOON (relevant EU project)
Fit Criterion:	All trained machine learning models can be serialized and exported to the working directory of the respective code.
Description:	Each organization with data scientists developing machine learning algorithms (T2.5, T3.2, T3.3, T4.2, T4.3, T5.2, T5.5) will be responsible to implement this requirement for its algorithms. Each machine learning model needs to be accessible for execution only by the respective component. If one component needs to execute a model of another component, it should call that component, which will read the model by itself.

[REC-5] Data originating in the production sites must be able to be transferred to RECLAIM Repository.

Issue Links:	**Relates**		
	relates to	REC-6	The data scientist must have access t...
	relates to	REC-3	The data scientist has access to a vi...
	relates to	REC-6	The data scientist must have access t...
	relates to	REC-35	Data are homogeneously stored in a ce...
	relates to	REC-37	Transformation of data for their deli...

	relates to REC-3 The data scientist has access to a vi...
Requirement Type:	Functional
Reporter:	CERTH
Implementation Assignee:	ADV, CERTH, CTCR, FCY, FEUP, FINT, FLUCHOS, GORENJE, HWH, ICE, LINKS, PODIUM, TECNALIA, ZORLUTEKS
Quality Check Assignee:	HWH
RECLAIM Component(s):	Adaptive Sensorial Network & Infrastructure for Digital Retrofitting, Data handler, Distributed data storage and analytics (DRy)
Status:	part of specification
Rationale:	In order to transfer the data from the production sites to the project's repository, it is mandatory that all necessary ports are opened on production sites (pilots).
Source:	MONSOON (relevant EU project)
Fit Criterion:	The necessary network communication ports are open, data from the plants are available in RECLAIM Repository.

[REC-6] The data scientist must have access to the data from the project's repository.	
Issue Links:	**Relates**
	relates to REC-5 Data originating in the production si...
	relates to REC-35 Data are homogeneously stored in a ce...
	relates to REC-5 Data originating in the production si...
Requirement Type:	Functional
Reporter:	CERTH

Implementation Assignee:	CERTH, ICE
Quality Check Assignee:	ICE
RECLAIM Component(s):	Data handler, Distributed data storage and analytics (DRy)
Status:	part of specification
Rationale:	There must be access to the stored data from the project's repository in order to train the machine learning algorithms or make other calculations.
Source:	MONSOON (relevant EU project)
Fit Criterion:	Access to the data from RECLAIM Repository is available.
Description:	The data stored in ICE's database (DRy) will be transferred to the Data handler, for which CERTH is responsible, and after converted to the appropriate format they will be transferred to the data analytics components of the RECLAIM platform. The data scientist shall be able to browse the data stored in the project's repository and, upon selection of data, these will be transferred to the Data handler.

[REC-7] The administrator can manage the platform via an integrated web-user management console.	
Requirement Type:	Functional
Reporter:	CERTH
Implementation Assignee:	CERTH, FEUP, HWH, ICE, LINKS, SUPSI
Quality Check Assignee:	HWH
RECLAIM Component(s):	Algorithms for quality prediction and process parameter optimization, Anomaly Detection, AR mechanisms, Cost Modelling and Financial Analysis Toolkit, Degradation models, Digital Twin for simulation, Distributed data storage and analytics (DRy), Fault

	Diagnosis and Predictive Maintenance Simulation Engine using Digital Twin, In-Situ Repair Data Analytics, Integrated Decision Support Framework (DSF) for Refurbishment & Remanufacturing Optimization, Life cycle assessment + LCA dashboard, Machine Operational Profiling, Optimization Toolkit for Refurbishment and Remanufacturing Planning, Predictive Maintenance, Prognostic and Health Management Toolkit, Reliability Analysis Tool
Status:	part of specification
Rationale:	For system administrators, it is more convenient to manage the whole platform "from one place" using the integrated management console which provides overall view of all platform services.
Source:	MONSOON (relevant EU project)
Fit Criterion:	The Management Tools provide integrated web-user management console for whole platform.
Description:	Analogous responsibilities as for REC-3 for the data analysis components, ICE and FEUP are responsible for the RECLAIM Repository, and Edge Device with Digital Twin integration and data processing algorithms, respectively. The resources, status, logs etc. will be shown for every service or component separately. The simplest way is to use Portainer (https://www.portainer.io/), a common tool for Docker administration and passive monitoring. This requirement can be fulfilled either in the same server as the one that the related services will use, or remotely in another server. The position of the management interface in the architecture will depend on the way that the deployment will be done.

[REC-8] The administrator can use active monitoring of the project's platform services.	
Requirement Type:	Functional
Reporter:	CERTH

Implementation Assignee:	CERTH, HWH, ICE, LINKS, SUPSI
Quality Check Assignee:	HWH
RECLAIM Component(s):	Algorithms for quality prediction and process parameter optimization, Anomaly Detection, AR mechanisms, Cost Modelling and Financial Analysis Toolkit, Degradation models, Digital Twin for simulation, Distributed data storage and analytics (DRy), Fault Diagnosis and Predictive Maintenance Simulation Engine using Digital Twin, In-Situ Repair Data Analytics, Integrated Decision Support Framework (DSF) for Refurbishment & Remanufacturing Optimization, Life cycle assessment + LCA dashboard, Machine Operational Profiling, Optimization Toolkit for Refurbishment and Remanufacturing Planning, Predictive Maintenance, Prognostic and Health Management Toolkit
Status:	part of specification
Rationale:	Since the project's platform consists of many services, it is necessary to provide tools for monitoring of the whole environment where the administrators will have detailed information about the status of the services and environment.
Source:	MONSOON (relevant EU project)
Fit Criterion:	The Management Tools support active monitoring of all platform services.
Description:	Analogous responsibilities as for REC-3 for the data analysis components, ICE will be responsible for the RECLAIM Repository.

[REC-9] The end user has the ability to edit dashboard panels.	
Requirement Type:	Functional
Reporter:	CERTH
Implementation Assignee:	CERTH, HWH, LINKS, SUPSI
Quality Check Assignee:	HWH

RECLAIM Component(s):	Users Interfaces
Status:	part of specification
Rationale:	The end user should be able to edit dashboards in order to focus on specific information displayed on them.
Source:	MONSOON (relevant EU project)
Fit Criterion:	The dashboards are editable. For example, the time range visualized in a time series and the visible metric plots should be configurable.
Description:	Similar responsibilities as for REC-3.

[REC-10] Dashboards can be printed.	
Requirement Type:	Functional
Reporter:	CERTH
Implementation Assignee:	CERTH, HWH, LINKS, SUPSI
Quality Check Assignee:	HWH
RECLAIM Component(s):	Users Interfaces
Status:	part of specification
Rationale:	Exporting dashboards allows them to be preserved when migrating or upgrading the data visualization service.
Source:	MONSOON (relevant EU project)
Fit Criterion:	Dashboards can be printed or exported.
Description:	Similar responsibilities as for REC-3.

[REC-11] Ability for WiFi connection at any time.	
Requirement Type:	Functional
Reporter:	CERTH
Implementation Assignee:	ADV, CTCR, FEUP, FINT, FLUCHOS, GORENJE, HWH, ICE, LINKS, PODIUM, TECNALIA, ZORLUTEKS
Quality Check Assignee:	HWH
RECLAIM Component(s):	AR mechanisms, Data handler, Distributed data storage and analytics (DRy), Fault Diagnosis and Predictive Maintenance Simulation Engine using Digital Twin, In-Situ Repair Data Analytics, Integrated Decision Support Framework (DSF) for Refurbishment & Remanufacturing Optimization, Users Interfaces
Status:	part of specification
Rationale:	The raw data and data analytics are updated real-time, and the end-user should often be immediately be informed about sudden changes (e.g. anomalies).
Source:	Documents about components and requirements of a previous Augmented Reality toolkit developed by CERTH.
Fit Criterion:	The percentage of time that the RECLAIM components are connected to WiFi.
Description:	Each pilot is responsible for itself.

[REC-12] The dashboards of the RECLAIM Platform must provide a friendly user interface.	
Requirement Type:	Non-Functional
Reporter:	CERTH
Implementation Assignee:	CERTH, HWH, LINKS, SUPSI
Quality Check	HWH

Assignee:	
RECLAIM Component(s):	Users Interfaces
Status:	part of specification
Rationale:	Most end users do not have technical knowledge.
Source:	Documents about components and requirements of a previous Augmented Reality toolkit developed by CERTH.
Fit Criterion:	End users state through some questionnaire that they are satisfied enough regarding the user-friendliness of the dashboards.
Description:	Similar responsibilities as for REC-3.

[REC-13] The Users Interfaces of the RECLAIM Platform must support multiple languages.	
Requirement Type:	Non-Functional
Reporter:	CERTH
Implementation Assignee:	CERTH, HWH, LINKS, SUPSI
Quality Check Assignee:	HWH
RECLAIM Component(s):	Users Interfaces
Status:	part of specification
Rationale:	Multiple pilots are involved in the project, and some end users may not have good knowledge of English.
Source:	Documents about components and requirements of a previous Augmented Reality toolkit developed by CERTH.
Fit Criterion:	End users state through some questionnaire that they do not have any language-related difficulties to understand the dashboards.

Description:	Similar responsibilities as for REC-3.

[REC-14] The Users Interfaces of the RECLAIM Platform must include a main page with a menu of its dashboards.	
Requirement Type:	Non-Functional
Reporter:	CERTH
Implementation Assignee:	CERTH, HWH, LINKS, SUPSI
Quality Check Assignee:	HWH
RECLAIM Component(s):	Users Interfaces
Status:	part of specification
Rationale:	The end users should have a wide view of RECLAIM platform.
Source:	Documents about components and requirements of a previous Augmented Reality toolkit developed by CERTH.
Fit Criterion:	The main page is in place.
Description:	Similar responsibilities as for REC-3.

[REC-15] The AR mechanisms must provide information for every mapped component of the equipment of interest.	
Requirement Type:	Functional
Reporter:	CERTH
Implementation Assignee:	CERTH
Quality Check Assignee:	SCM

RECLAIM Component(s):	AR mechanisms
Status:	part of specification
Rationale:	This requirement is related to the purpose of the real-time 3D annotation module of the AR mechanisms.
Source:	Documents about components and requirements of a previous Augmented Reality toolkit developed by CERTH.
Fit Criterion:	The AR mechanisms provide information for every mapped component of the equipment of interest.

[REC-16] The AR mechanisms must be able to update information data of every component.

Requirement Type:	Functional
Reporter:	CERTH
Implementation Assignee:	CERTH
Quality Check Assignee:	SCM
RECLAIM Component(s):	AR mechanisms
Status:	part of specification
Rationale:	The AR mechanisms are interactive.
Source:	Documents about components and requirements of a previous Augmented Reality toolkit developed by CERTH.
Fit Criterion:	The AR mechanisms are able to update information data of every component.

[REC-17] The AR mechanisms must be able to register and annotate equipment components that have not been recognized or registered.

Requirement Type:	Functional
Reporter:	CERTH
Implementation Assignee:	CERTH
Quality Check Assignee:	SCM
RECLAIM Component(s):	AR mechanisms
Status:	part of specification
Rationale:	The AR mechanisms must be as automatic as possible.
Source:	Documents about components and requirements of a previous Augmented Reality toolkit developed by CERTH.
Fit Criterion:	The percentage of registered and annotated components, based on domain expert's ground truth information.

[REC-18] The AR mechanisms must be able to recognize equipment components.	
Requirement Type:	Functional
Reporter:	CERTH
Implementation Assignee:	CERTH
Quality Check Assignee:	SCM
RECLAIM Component(s):	AR mechanisms
Status:	part of specification
Rationale:	The AR mechanisms must be as automatic as possible.

Source:	Documents about components and requirements of a previous Augmented Reality toolkit developed by CERTH.
Fit Criterion:	The percentage of recognized components, based on domain expert's ground truth information.

[REC-19] The AR mechanisms must be able to exchange data with the RECLAIM Repository and the Users Interfaces.	
Requirement Type:	Functional
Reporter:	CERTH
Implementation Assignee:	CERTH
Quality Check Assignee:	SCM
RECLAIM Component(s):	AR mechanisms, Data handler, Users Interfaces
Status:	part of specification
Rationale:	The need for this requirement results directly from the envisioned functionality of the AR mechanisms.
Source:	Grant Agreement, documents about components and requirements of a previous Augmented Reality toolkit developed by CERTH.
Fit Criterion:	The AR mechanisms are able to store the semantic 3D model, the user input for annotation in RECLAIM Repository and use it in the future, and the Digital Twin predictions can be read from RECLAIM Repository.

[REC-20] The AR mechanisms must be able to receive data regarding all the possible and the optimal refurbishment / remanufacturing steps, through the Refurbishment & Remanufacturing Framework.	
Requirement Type:	Functional

Reporter:	CERTH
Implementation Assignee:	CERTH, HWH, SCM
Quality Check Assignee:	SCM
RECLAIM Component(s):	AR mechanisms
Status:	part of specification
Rationale:	These steps will enable the AR mechanisms to indicate the parts to be refurbished or remanufactured and the corresponding instructions to end users.
Source:	Grant Agreement
Fit Criterion:	The steps have been inserted in the software of the AR mechanisms.

[REC-21] The AR mechanisms must be able to determine user's position and orientation.	
Requirement Type:	Functional
Reporter:	CERTH
Implementation Assignee:	CERTH
Quality Check Assignee:	SCM
RECLAIM Component(s):	AR mechanisms
Status:	part of specification
Rationale:	This requirement is related to the purpose of the Indoor localization and 3D registration module of the AR mechanisms.
Source:	Documents about components and requirements of a previous

	Augmented Reality toolkit developed by CERTH.
Fit Criterion:	The error in the estimation of user's position and orientation, based on ground truth information.

[REC-22] The AR mechanisms must be able to locate every recognized characteristic of the equipment.	
Requirement Type:	Functional
Reporter:	CERTH
Implementation Assignee:	CERTH
Quality Check Assignee:	SCM
RECLAIM Component(s):	AR mechanisms
Status:	part of specification
Rationale:	This requirement is related to the purpose of the Indoor localization and 3D registration module of the AR mechanisms.
Source:	Documents about components and requirements of a previous Augmented Reality toolkit developed by CERTH.
Fit Criterion:	The error in the estimation of the position of the characteristics, based on ground truth information.

[REC-23] Streaming real-time data from a messaging service of RECLAIM Repository to a distributed data processing framework of RECLAIM Repository.	
Requirement Type:	Functional
Reporter:	CERTH
Implementation Assignee:	ADV, CERTH, CTCR, FCY, FEUP, FINT, HWH, ICE, LINKS, TECNALIA

Quality Check Assignee:	LINKS
RECLAIM Component(s):	Adaptive Sensorial Network & Infrastructure for Digital Retrofitting, Data handler, Distributed data storage and analytics (DRy), IoT Gateway software stack, Prognostic and Health Management Toolkit
Status:	part of specification
Rationale:	By integration of the Messaging Service and the Distributed Data Processing framework, it will be possible to incrementally update or validate predictive models or apply them on new operational data directly on the project's platform in real time.
Source:	MONSOON (related EU project)
Fit Criterion:	The Messaging Service of RECLAIM Repository supports programming interface for streaming real-time data directly to the Distributed Data Processing framework.

[REC-24] Support for MQTT protocol.	
Requirement Type:	Functional
Reporter:	CERTH
Implementation Assignee:	ADV, CERTH, CTCR, FCY, FEUP, FINT, HWH, ICE, LINKS, TECNALIA
Quality Check Assignee:	LINKS
RECLAIM Component(s):	Adaptive Sensorial Network & Infrastructure for Digital Retrofitting, Data handler, Distributed data storage and analytics (DRy), IoT Gateway software stack
Status:	part of specification
Rationale:	Message Queue Telemetry Transport (MQTT) protocol is one of the most adopted standards for IoT communication. The platform should support direct streaming of data from the IoT environment.

Source:	MONSOON (relevant EU project)
Fit Criterion:	Support for MQTT protocol is provided.

[REC-26] Support for queries over big data file formats.	
Requirement Type:	Functional
Reporter:	CERTH
Implementation Assignee:	ICE
Quality Check Assignee:	ICE
RECLAIM Component(s):	Distributed data storage and analytics (DRy)
Status:	part of specification
Rationale:	Big data file formats are especially designed for efficient storage and query of very large data sets.
Source:	MONSOON (relevant EU project)
Fit Criterion:	The Distributed data storage and analytics component supports direct query over big data file formats.

[REC-27] Support for batch processing of large data sets.	
Requirement Type:	Functional
Reporter:	CERTH
Implementation Assignee:	ICE
Quality Check Assignee:	ICE

RECLAIM Component(s):	Distributed data storage and analytics (DRy)
Status:	part of specification
Rationale:	Since historical data can be stored in very large data sets, where the size of the data exceed the size of the memory of common server configuration (i.e. more than 512GB - 1TB), it is necessary to support distributed computing on multiple servers.
Source:	MONSOON (related EU project)
Fit Criterion:	The Distributed Data Processing Framework is able to process very large data sets.

[REC-31] Define the optimal time to carry out the suggested operation (refurbishment/remanufacturing) for each machine.

Requirement Type:	Functional
Reporter:	FCY
Implementation Assignee:	FCY
Quality Check Assignee:	LINKS
RECLAIM Component(s):	Optimization Toolkit for Refurbishment and Remanufacturing Planning
Status:	part of specification
Rationale:	The system should provide optimal time to carry out the suggested operation (refurbishment, remanufacturing) for each machine, in real-time, in order to make actions about machine lifecycle. In addition to the solution, an understandable rational should be provided.
Source:	RECLAIM Documentation
Fit Criterion:	Real-time machine optimal time to carry out refurbishment/remanufacturing data sent to RECLAIM Repository.

[REC-32] The Optimization Toolkit models should support real-time calibration/fine-tuning.	
Requirement Type:	Functional
Reporter:	FCY
Implementation Assignee:	FCY
Quality Check Assignee:	LINKS
RECLAIM Component(s):	Optimization Toolkit for Refurbishment and Remanufacturing Planning
Status:	part of specification
Rationale:	The optimization models for each machine must support calibration/fine-tuning in real time, using sensorial network information (T3.1)
Source:	RECLAIM Documentation
Fit Criterion:	Real-time fine-tuning optimization data sent to RECLAIM Repository.

[REC-33] The Optimization Toolkit should integrate machine prediction failures and severity range patterns classifications.	
Requirement Type:	Functional
Reporter:	FCY
Implementation Assignee:	FCY
Quality Check Assignee:	LINKS
RECLAIM	Optimization Toolkit for Refurbishment and Remanufacturing

Component(s):	Planning
Status:	part of specification
Rationale:	The system must integrate the machine predictions for failures/malfunctions and severity range patterns classifications from Digital Twin (T3.3).
Source:	RECLAIM Documentation
Fit Criterion:	Full integration of predictions and classifications

[REC-34] Integrate different machinery data and propose solutions for different severity ranges.	
Requirement Type:	Functional
Reporter:	FCY
Implementation Assignee:	FCY
Quality Check Assignee:	LINKS
RECLAIM Component(s):	Optimization Toolkit for Refurbishment and Remanufacturing Planning
Status:	part of specification
Rationale:	The toolkit should integrate real-time data (T3.1), machinery and processes profile (T3.2), and predictive maintenance simulations related to a fault analysis (T3.3), in order to generate the Optimized Process Plan according to the severity range for the machine. Also, the proposed plan should be simulated and validated in the Digital Twin.
Source:	RECLAIM Documentation
Fit Criterion:	Full integration of different machinery data and proposed solutions for each severity range.

[REC-35] Data are homogeneously stored in a central or distributed database.	
Issue Links:	Relates relates to REC-6 The data scientist must have access t... relates to REC-37 Transformation of data for their deli... relates to REC-3 The data scientist has access to a vi... relates to REC-5 Data originating in the production si...
Requirement Type:	Functional
Reporter:	LINKS
Implementation Assignee:	ASTON, CERTH, CTCR, FLUCHOS, GORENJE, ICE, LINKS, PODIUM, ROBOTEH, SUPSI, TTS, ZORLUTEKS
Quality Check Assignee:	CERTH
RECLAIM Component(s):	Adaptive Sensorial Network & Infrastructure for Digital Retrofitting, Data handler, Distributed data storage and analytics (DRy)
Status:	part of specification
Rationale:	In order to have a fast and easy access to data, data should be stored in the db using a common data model among the different sources and pilots.
Source:	MONSOON (relevant EU project)
Fit Criterion:	The Distributed Database contains homogeneous data.

[REC-36] Clear description of data	
Requirement Type:	Non-Functional
Reporter:	LINKS
Implementation	ADV, ASTON, CERTH, CTCR, ESCI, FCY, FEUP, FINT, FLUCHOS,

Assignee:	GORENJE, HWH, ICE, LINKS, PODIUM, ROBOTEH, SCM, SEZ, SUPSI, TECNALIA, TTS, ZORLUTEKS
Quality Check Assignee:	CERTH
RECLAIM Component(s):	Distributed data storage and analytics (DRy)
Status:	under implementation
Rationale:	Data should have a clear description and association with the process
Source:	MONSOON (relevant EU project)
Fit Criterion:	Data documentation is clear

[REC-37] Transformation of data for their delivery from the RECLAIM Repository database to the Users Interfaces.

Issue Links:	**Relates** relates to REC-3 The data scientist has access to a vi... relates to REC-5 Data originating in the production si... relates to REC-35 Data are homogeneously stored in a ce...
Requirement Type:	Functional
Reporter:	CERTH
Implementation Assignee:	ASTON, CERTH, CTCR, FLUCHOS, GORENJE, ICE, LINKS, PODIUM, ROBOTEH, SUPSI, TTS, ZORLUTEKS
Quality Check Assignee:	ICE
RECLAIM Component(s):	Data handler, Distributed data storage and analytics (DRy), Users Interfaces
Status:	part of specification

Rationale:	Raw data and algorithmic outputs will have a holistic format in the RECLAIM Repository database, but they will need to be transformed to a format acceptable by each pilot's users interfaces.
Source:	RECLAIM Grant Agreement
Fit Criterion:	Desired data are properly depicted in the users interfaces.
Description:	CERTH will lead the transformation of visualization data from the database (DRy) to Users Interfaces through the Data Handler.

[REC-38] Process modelling	
Issue Links:	**Relates** relates to REC-39 Machine modelling relates to REC-40 Impacts computation relates to REC-43 LCA scenarios comparison relates to REC-46 LCA service providing
Requirement Type:	Functional
Reporter:	SUPSI
Implementation Assignee:	SUPSI
Quality Check Assignee:	CERTH
RECLAIM Component(s):	Data handler, Distributed data storage and analytics (DRy), Life cycle assessment + LCA dashboard
Status:	part of specification
Rationale:	The LCA tool has to model the machine(s) middle of life in analysis.
Source:	MANU-SQUARE (H2020 project)

Fit Criterion:	Tool development. Potentially already existing process formalisms might be useful.
Description:	The process modelling is a functionality provided by the LCA tool where users need to specify information like process input and output (like materials, parts and components) and process characteristics (like the involved machine). The modelled processes represent part of the input for the logics of the sustainability assessment.

[REC-39] Machine modelling

Issue Links:	**Relates**		
	relates to	REC-40	Impacts computation
	relates to	REC-41	Integration of the digital twin data
	relates to	REC-43	LCA scenarios comparison
	relates to	REC-46	LCA service providing
	relates to	REC-38	Process modelling
Requirement Type:	Functional		
Reporter:	SUPSI		
Implementation Assignee:	SUPSI		
Quality Check Assignee:	CERTH		
RECLAIM Component(s):	Data handler, Distributed data storage and analytics (DRy), Life cycle assessment + LCA dashboard, Machine Operational Profiling		
Status:	part of specification		
Rationale:	The LCA tool has to support the modelling of different machines middle of life scenarios.		
Source:	MANU-SQUARE (H2020 project)		

Fit Criterion:	Tool development. Potentially already existing machine formalisms might be useful.
Description:	The machine modelling is a functionality provided by the LCA tool where users need need to specify information like the machine capabilities, potential connected processes, etc. The modelled machines represent part of the input the process modelling.

[REC-40] Impacts computation			
Issue Links:	**Relates**		
	relates to	REC-43	LCA scenarios comparison
	relates to	REC-45	Saving of the assessment result
	relates to	REC-46	LCA service providing
	relates to	REC-38	Process modelling
	relates to	REC-39	Machine modelling
Requirement Type:	Functional		
Reporter:	SUPSI		
Implementation Assignee:	SUPSI		
Quality Check Assignee:	CERTH		
RECLAIM Component(s):	Data handler, Distributed data storage and analytics (DRy), Life cycle assessment + LCA dashboard		
Status:	part of specification		
Rationale:	The LCA tool has to compute impact data related to defined machines middle of life scenarios.		
Source:	MANU-SQUARE (H2020 project), MANUTELLIGENCE (FP7 project)		
Fit Criterion:	Developers are able to write code/libraries from a development environment. Developers have basic knowledge on the LCA		

	methodology.

[REC-41] Integration of the digital twin data			
Issue Links:	**Relates**		
	relates to	REC-43	LCA scenarios comparison
	relates to	REC-46	LCA service providing
	relates to	REC-39	Machine modelling
Requirement Type:	Functional		
Reporter:	SUPSI		
Implementation Assignee:	SUPSI		
Quality Check Assignee:	CERTH		
RECLAIM Component(s):	Data handler, Distributed data storage and analytics (DRy), Life cycle assessment + LCA dashboard		
Status:	part of specification		
Rationale:	The LCA tool has to connect with the digital twin to feed the models with middle of life data.		
Fit Criterion:	Tool development.		
Description:	The integration of the LCA tool with the digital twin enables the forecast of the sustainability performances and comparison of the different scenarios which differ on the lifetime extension methods.		

[REC-42] Retrieving of the predicted data	
Issue Links:	**Relates**

	relates to REC-46 LCA service providing
Requirement Type:	Functional
Reporter:	SUPSI
Implementation Assignee:	SUPSI
Quality Check Assignee:	CERTH
RECLAIM Component(s):	Data handler, Distributed data storage and analytics (DRy), Life cycle assessment + LCA dashboard
Status:	part of specification
Rationale:	LCA tool has to retrieve data from prediction engines to support impact calculation of forecasting scenarios.
Fit Criterion:	Developers are able to write code/libraries from a development environment.

[REC-43] LCA scenarios comparison

Issue Links:	**Relates**		
	relates to	REC-46	LCA service providing
	relates to	REC-38	Process modelling
	relates to	REC-39	Machine modelling
	relates to	REC-40	Impacts computation
	relates to	REC-41	Integration of the digital twin data
Requirement Type:	Functional		
Reporter:	SUPSI		
Implementation	SUPSI		

Assignee:	
Quality Check Assignee:	CERTH
RECLAIM Component(s):	Data handler, Distributed data storage and analytics (DRy), Life cycle assessment + LCA dashboard
Status:	part of specification
Rationale:	The LCA tool has to connect with the digital twin to feed the models with middle of life data.
Fit Criterion:	Tool development.

[REC-45] Saving of the assessment result	
Issue Links:	**Relates** relates to REC-46 LCA service providing relates to REC-40 Impacts computation
Requirement Type:	Functional
Reporter:	SUPSI
Implementation Assignee:	SUPSI
Quality Check Assignee:	CERTH
RECLAIM Component(s):	Data handler, Distributed data storage and analytics (DRy), Life cycle assessment + LCA dashboard
Status:	part of specification
Rationale:	The LCA tool has to visualize assessment results and scenarios comparisons. The LCA tool has to be able to save all computed impacts into the RECLAIM repository.
Fit Criterion:	Developers are able to write code/libraries from a development

	environment.

[REC-46] LCA service providing	
Issue Links:	**Relates** relates to REC-38 Process modelling relates to REC-39 Machine modelling relates to REC-40 Impacts computation relates to REC-41 Integration of the digital twin data relates to REC-42 Retrieving of the predicted data relates to REC-43 LCA scenarios comparison relates to REC-45 Saving of the assessment result
Requirement Type:	Functional
Reporter:	SUPSI
Implementation Assignee:	SUPSI
Quality Check Assignee:	CERTH
RECLAIM Component(s):	Data handler, Distributed data storage and analytics (DRy), Life cycle assessment + LCA dashboard
Status:	part of specification
Rationale:	The execution of the assessment logics (belonging to the LCA tool) need to be made available through service endpoints.
Fit Criterion:	Tool development.
Description:	In order to guarantee its integration, the LCA tool needs to expose a set of web services (for example by means of the REST paradigm) meant to provide some functionalities to the rest of the platform.

[REC-47] Cybersecurity monitoring and protection of the deployed embedded systems and IT infrastructure.	
Issue Links:	Relates relates to REC-71 FPGA-accelerated cyber security modul... relates to REC-81 FPGA-accelerated cyber security devic...
Requirement Type:	Functional
Reporter:	FINT
Implementation Assignee:	FINT
Quality Check Assignee:	CERTH
RECLAIM Component(s):	FPGA-accelerated cyber security module
Status:	open
Rationale:	The protection of the deployed embedded systems and IT infrastructure from external and internal cybersecurity threats is critical for ensuring the uninterrupted and correct operation of the deployed assets. To achieve this level of protection, mechanisms providing the monitoring, detection and mitigation of the threats need to be in place.
Source:	RECLAIM Grant Agreement, Security By Design, Best practices in Cyber security, NIST SP 800-53, ISO 27002 "Code practice for information security controls"
Fit Criterion:	Perform some form of penetration testing (focused on the threats the detection and analysis modules claims to detect) and verify that a) the threat is detected; b) the the threat is controlled (e.g. blocked by the firewall); and c) the event is logged and reported. The tests should check the implemented functionalities (detection, control, reporting) and also some quality aspects like performance (i.e. speed).

Description:	Cybersecurity monitoring and protection of the deployed embedded systems and IT infrastructure.

[REC-48] Shop floor visualization	
Requirement Type:	Functional
Reporter:	HWH
Implementation Assignee:	HWH
Quality Check Assignee:	FEUP
RECLAIM Component(s):	AR mechanisms, Users Interfaces
Status:	part of specification
Rationale:	Relevant maintenance data must be available on shop floor at the machine's HMI.
Fit Criterion:	Dashboards need to be embeddable and presentable fullscreen.

[REC-49] Shop floor visualization (small screen type)	
Requirement Type:	Functional
Reporter:	HWH
Implementation Assignee:	HWH
Quality Check Assignee:	FEUP
RECLAIM Component(s):	AR mechanisms, Users Interfaces
Status:	part of specification

Rationale:	Relevant maintenance data must be available on shop floor in a compressed format also on mobile devices (panel, smart phone).
Fit Criterion:	The dashboard system should be mobile compatible.

[REC-50] Big data analysis	
Requirement Type:	Functional
Reporter:	HWH
Implementation Assignee:	HWH
Quality Check Assignee:	FEUP
RECLAIM Component(s):	Adaptive Sensorial Network & Infrastructure for Digital Retrofitting, Algorithms for quality prediction and process parameter optimization, Anomaly Detection, Cost Modelling and Financial Analysis Toolkit, Data handler, Degradation models, Digital Twin for simulation, Distributed data storage and analytics (DRy), Fault Diagnosis and Predictive Maintenance Simulation Engine using Digital Twin, In-Situ Repair Data Analytics, Integrated Decision Support Framework (DSF) for Refurbishment & Remanufacturing Optimization, IoT Gateway software stack, Life cycle assessment + LCA dashboard, Machine Operational Profiling, Optimization Toolkit for Refurbishment and Remanufacturing Planning, Predictive Maintenance, Prognostic and Health Management Toolkit, Reliability Analysis Tool
Status:	part of specification
Rationale:	Big data analysis shall be done on the basis of existing data sets and shall be able to be extended in case of new data and new data types.
Fit Criterion:	Well-defined interfaces and data storage in the RECLAIM architecture and repository.

[REC-51] Big data visualization

Requirement Type:	Functional
Reporter:	HWH
Implementation Assignee:	HWH
Quality Check Assignee:	FEUP
RECLAIM Component(s):	Users Interfaces
Status:	part of specification
Rationale:	Big data analysis shall be presented on system level, e.g. cloud infrastructure which is hosted within or outside of a factory environment.
Fit Criterion:	Consistent analysis presentation for all components keeping a visual project identity.

[REC-52] Machine's degradation state	
Requirement Type:	Functional
Reporter:	HWH
Implementation Assignee:	HWH
Quality Check Assignee:	FEUP
RECLAIM Component(s):	Degradation models
Status:	part of specification
Rationale:	The degradation state of the machine's components and of the entire machine shall be predicted based on relevant process information.

Source:	REBORN project
Fit Criterion:	Unit-Test: Different degradation states for worn-out / new process information.

[REC-53] Ease of usage, simple setup	
Requirement Type:	Non-Functional
Reporter:	HWH
Implementation Assignee:	HWH
Quality Check Assignee:	FEUP
RECLAIM Component(s):	Adaptive Sensorial Network & Infrastructure for Digital Retrofitting, Algorithms for quality prediction and process parameter optimization, Anomaly Detection, AR mechanisms, Data handler, Degradation models, Digital Twin for simulation, Fault Diagnosis and Predictive Maintenance Simulation Engine using Digital Twin, In-Situ Repair Data Analytics, Life cycle assessment + LCA dashboard, Machine Operational Profiling, Predictive Maintenance, Reliability Analysis Tool, Users Interfaces
Status:	part of specification
Rationale:	The setup of any system within the RECLAIM architecture shall be as easy as possible. Training, expert knowledge, etc. shall not be required for setting up a component. Any system and any component shall be able for self-learning e.g. by using AI e.g. in order to identify setup parameters, to distinguish between regular and malfunction operations.
Fit Criterion:	Questionnaire/evaluation with pilots and non-technical partners.

[REC-54] Economic infrastructure	
Requirement Type:	Non-Functional

Reporter:	HWH
Implementation Assignee:	HWH
Quality Check Assignee:	FEUP
RECLAIM Component(s):	Adaptive Sensorial Network & Infrastructure for Digital Retrofitting, FPGA-accelerated cyber security module
Status:	part of specification
Rationale:	Additional hardware such as sensors or actuators are often costly in purchasing and maintaining. Furthermore, they are often difficult to attach to machines and are additional potential failure sources. As a consequence, additional hardware shall be used only if there are no other options for delivering a required output.
Fit Criterion:	An attempt to reuse computing and sensing equipment already present was made.

[REC-55] Evaluation of production system performances in terms of throughput, assets utilization, etc.

Requirement Type:	Functional
Reporter:	TTS
Implementation Assignee:	TTS
Quality Check Assignee:	LINKS
RECLAIM Component(s):	Digital Twin for simulation
Status:	part of specification
Rationale:	The Digital twin simulates the shop floor and provides KPI to evaluate the performances of a production system considering different production and maintenance scenarios.

Annex 2: Hardware components list

A2.1 Machine Vision System

Table 26 - Filled-in hardware specifications template for the Machine Vision System

Issue Links:	**Relates** relates to REC-82 Machine Vision Toolkit (T3.1, FINT)
Description and Functionality	
Name:	Machine Vision System
Measurement:	The part ID identified or NONE if unknown, the alignment status of the part (true: correctly aligned/false: incorrectly aligned
Functionality:	Production line enhancement and predictive maintenance
Physical Characteristics	
Length (mm):	200
Width (mm):	200
Height (mm):	100
Weight (kg):	2
Material:	aluminium, plastic, stainless steel
Mounting:	wall mount, pole mount, custom
Operational Characteristics	
Measurement Range:	frame size: 50x50cm, frame rate: 1 - 5fps
Measurement Resolution:	TBA
Accuracy:	TBA

Zero Error:	TBA
Temperature:	+5°C to +45°C
Humidity:	20% - 80%
Lifetime:	5 years
Hardware Requirements	
Power Requirements:	230V AC, 25W
Data Connections:	Ethernet, WiFi
Data Format:	NGSI, jpg file
Data Rate:	every 200msec minimum, dependable on production line speed
Data Availability:	continuous, on demand
Software Requirements	
Software Required:	Yes
Software Details:	Machine Vision software stack based on ML (OpenCV, Tensorflow, Keras etc.)

A2.2 IoT Gateway with AI acceleration

Table 27 - Filled-in hardware specifications template for the IoT Gateway with AI acceleration

Issue Links:	**Relates** relates to REC-72 IoT Gateway software stack (T3.1, FIN... relates to REC-79 ASN Sensor Protocol Adaptation (T3.1,... relates to REC-80 ASN Actuator Protocol Adaptation (T3....
Description and Functionality	

Name:	IoT Gateway
Functionality:	Operates as a border router for wireless sensors, caches and processes sensor data using ML algorithms when needed, optional machine vision camera interface
Physical Characteristics	
Length (mm):	200
Width (mm):	200
Height (mm):	100
Weight (kg):	1.5
Material:	aluminium
Mounting:	wall mount, pole mount
Operational Characteristics	
Temperature:	-10°C to +50°C
Humidity:	10% - 90%
Lifetime:	10 years
Hardware Requirements	
Power Requirements:	230V AC, 20W
Data Connections:	Ethernet, WiFi, LoRa (optional), 6LoWPAN, MODBUS
Data Format:	NGSI
Data Rate:	limited by underlying communication protocol only
Data Availability:	continuous, periodic, on demand
Software Requirements	
Software	Yes

Required:	
Software Details:	WSN session services and brokers, caching services (database), ML software stack

A2.3 LINKS FPGA platform

Table 28 - Filled-in hardware specifications template for the LINKS FPGA platform

Description and Functionality	
Name:	LINKS Gateway
Short description:	Flexible platform to be adapted to specific requirments
Functionality:	Edge Industrial gateway: flexible gateway that can connect different machines, sensors, cameras, etc., and hosts different type of algorithms using an ARM+FPGA or ARM+GPU approach. AI algorithms can be optimized to work on FPGA/GPU to be faster and consume less power
Software requirements	
Software required:	No
Software details:	The software will depend on the requirements and will be based on open-source projects and/or on SW directly developed by LINKS. FPGA accelerators will be based on open source libraries or on IP cores freely available from FPGA vendor.

LINKS is a research center and does not have a product but can customize the board to fulfill the requirements of the application.

This is true for both HW and SW including the support for external sensors and the handling of different types of data transport protocols (currently at least MQTT, AMQP and Websocket are supported).

A2.4 FPGA-accelerated cyber security device

Table 29 - Filled-in hardware specifications template for the FPGA-accelerated cyber security device

Issue Links:	Relates relates to REC-47 Cybersecurity monitoring and protecti...
Description and Functionality	
Name:	Cyber security GW
Functionality:	Monitors and analyses network traffic destined to or stemming from the assets for detecting threats and anomalous patterns; Blocks the network connections that are evaluated as threats; logs related security events (e.g. to be use later from an auditing mechanism); Provides acceleration to the sophisticated cyber threat detection and analysis algorithms.
Physical Characteristics	
Material:	aluminium, plastic, stainless steel
Mounting:	Rack mount
Operational Characteristics	
Temperature:	0°C to 70°C
Hardware Requirements	
Power Requirements:	220V AC
Data Connections:	Ethernet, USB, Serial
Data Rate:	Max 1Gbps (Ethernet)
Software Requirements	
Software Required:	Yes
Software Details:	The SW modules are needed for implementing a) the management modules of the Cyber security GW; b) the cyber threat detection and analysis modules; c) the threat mitigation/control modules

	that will block the threats; and d) the communication/interfacing modules that will enable for the interaction with other external components (e.g. central logging server, software agents, etc.).

A2.5 Friction Welding Machine

Table 30 - Filled-in hardware specifications template for the Friction Welding Machine

Description and Functionality	
Name:	Friction Welding Machine
Measurement:	several process data (distance, speed, air pressure/forces, times), average load
Digital/Analog Signals:	digital distance measurement, digital io nio output analog pressure, speed input/output
Functionality:	manual friction welding machine
Physical Characteristics	
Length (mm):	2,000
Width (mm):	1,200
Height (mm):	2,200
Weight (kg):	2,000
Material:	aluminium, plastic, stainless steel, copper
Mounting:	free standing on floor with feet
Operational Characteristics	
Measurement Range:	approx. 500 Hz measurement frequency
Measurement Resolution:	2ms
Accuracy:	TBA

Zero Error:	TBA
Temperature:	+5°C to +45°C
Humidity:	20% - 80%
Lifetime:	10 years
Hardware Requirements	
Power Requirements:	380V 32A AC, 12kW
Data Connections:	Ethernet
Data Format:	raw data
Data Rate:	every 2msec minimum, dependable on production line speed
Data Availability:	continuous
Software Requirements	
Software Required:	Yes
Software Details:	Machine operating system based on FreeBSD for monitoring/communication and embedded software stack for control.

Annex 3: Individual data models

The data models corresponding to data transferred through the RECLAIM Repository follow in JSON format. By these data models the holistic data model, which is the common information model presented in Section 3.1.9 Common information model, has been derived. In this version of the deliverable, not all data models could be specified, so only the available ones were considered.

A3.1 Data from/to the Adaptive Sensorial Network

The data transferred from the ASN to the Data Handler are produced by the Sensor Protocol Adaptation and the Machine Vision Toolkit subcomponents of the ASN.

A3.1.1 ASN Sensor

```
[
  {
    "id": "a4cc9834-f4dc-44b4-83d2-2d06f8ed6def",
    "type": "Machine",
    "TimeInstant": {
      "type": "DateTime",
      "value": "2020-06-19T12:06:08.00Z"
    },
    "owner": {
      "type": "Text",
      "value": "OWNER"
    },
    "transducers": {
      "type": "Array",
      "value": [
        "046c48c2-4d23-416d-9efe-64344e19faba",
      ]
    }
  },
  {
    "id": "046c48c2-4d23-416d-9efe-64344e19faba",
    "type": "GenericSensor",
    "TimeInstant": {
      "type": "DateTime",
      "value": "2020-06-19T12:06:08.00Z"
    },
    "name": {
      "type": "Text",
      "value": "My Sensor 22"
    },
    "position": {
      "type": "Text",
      "value": "POSITION"
    },
    "units": {
      "type": "Text",
      "value": "UNITS"
    },
    "value": {
      "type": "Text",
      "value": "VALUE"
    }
  }
]
```

A3.1.2 ASN Actuator

The idea for the ASN Actuator is to receive data from the RECLAIM Repository instead of sending data to it. Despite this, it was considered that the corresponding individual data model should also be compatible with the CIM. During the 2nd phase of T2.3 it will be examined if there will be appropriate devices for data communication to the ASN Actuator.

```
[
  {
    "id": "a4cc9834-f4dc-44b4-83d2-2d06f8ed6def",
    "type": "Machine",
    "TimeInstant": {
      "type": "DateTime",
      "value": "2020-06-19T12:06:08.00Z"
    },
    "owner": {
      "type": "Text",
      "value": "OWNER"
    },
    "transducers": {
      "type": "Array",
      "value": [
        "d270715e-1c22-4e74-97fe-e6dbf0e3f840"
      ]
    }
  },
  {
    "id": "d270715e-1c22-4e74-97fe-e6dbf0e3f840",
    "type": "GenericActuator",
    "TimeInstant": {
      "type": "DateTime",
      "value": "2020-06-19T12:06:08.00Z"
      },
      "name": {
        "type": "Text",
        "value": "My Actuator 123"
      },
      "position": {
        "type": "Text",
        "value": "POSITION"
      },
      "units": {
        "type": "Text",
        "value": "UNITS"
      },
      "value": {
        "type": "Text",
        "value": "VALUE"
      }
    }
]
```

A3.1.3 Machine Vision System

```
[
  {
    "id": "a4cc9834-f4dc-44b4-83d2-2d06f8ed6def",
    "type": "Machine",
    "TimeInstant": {
      "type": "DateTime",
      "value": "2020-06-19T12:06:08.00Z"
    },
    "owner": {
      "type": "Text",
      "value": "OWNER"
    },
    "transducers": {
      "type": "Array",
      "value": [
        "5877bcdc-e0f9-4187-a65b-aa8b5c80e0b0"
      ]
    }
  },
  {
    "id": "5877bcdc-e0f9-4187-a65b-aa8b5c80e0b0",
    "type": "VisionSensor",
    "TimeInstant": {
      "type": "DateTime",
      "value": "2020-06-19T12:06:08.00Z"
    },
    "name": {
      "type": "Text",
      "value": "My Vision Sensor 2"
    },
    "position": {
      "type": "Text",
      "value": "POSITION"
    },
    "requestedPart": {
      "type": "Text",
      "value": "PART_ID"
    },
    "identifiedPart": {
      "type": "Text",
      "value": "PART_ID"
    },
    "partMatch": {
      "type": "Boolean",
      "value": true
    },
    "partAligned": {
      "type": "Boolean",
      "value": false
    }
  },
]
```

A3.2 Data from the Reliability Analysis Tool

```
{
  "systemName": <str>,
  "systemId": <int>,
  "workingHours": <float>,
  "currentReliability": <float>,
  "meanTimeBetweenFailures": <float>,
  "analysisExecutionTimestamp" : <ISO/UNIX timestamp>,
  "components": [
    {
      "componentName": <str>,
      "repairable": <boolean>,
      "componentId": <int>,
      "meanFailureTime" : <float>,
      "stdDeviation" : <float>,
      "currentReliability": <float>,
    }
    {
      "componentName": <str>,
      "repairable": <boolean>,
      "componentId": <int>,
      "meanFailureTime" : <float>,
      "stdDeviation" : <float>,
      "currentReliability": <float>,
    }
    {
      "..." : "..."
    }
  ]
}
```

A3.3 Data from the Machinery Operational Profiling

```
{
    "TimeStamp": <ISO timestamp>,
    "Machine": {
       "MachineID": <str>,
       "Owner": <str>,
       "KPIs": [
          {"Kpi": {
             "@name": <str>,
             "@units": <str - measurement unit>,
             "#text": <str - number>
          }},…
       ],
       "Index": {
          "Health": {
             "@timestamp": <ISO timestamp>,
             "#text": <str - number>
          },
          "Performance": {
             "@timestamp": <ISO timestamp>,
             "#text": <str - number>
          },
          "Production": {
             "@timestamp": <ISO timestamp>,
             "#text": <str - number>
          }
       }
    }
 }
```

A3.4 Data from the Optimization Toolkit for Refurbishment and Remanufacturing Planning

```
{
  "output_name":
"FCY.OptimizationToolkit."+<pilot_name>+"."+<use_case_name>,
  "present_time": <ISO/UNIX timestamp>,
  "machine": <str>,
  "kpi_values": {
    "perfomance_"+<individual_kpi_1>: <float>,...
                 "perfomance_"+<individual_kpi_n>: <float>,
                 "operation_"+<individual_kpi_1>: <float>,...
                 "operation_"+<individual_kpi_n>: <float>,
                 "production_"+<individual_kpi_1>: <float>,...
                 "production_"+<individual_kpi_n>: <float>,
                 "process_"+<individual_kpi_1>: <float>,...
                 "process_"+<individual_kpi_n>: <float>
      },
  "operation": {
    "component": <str>,
    "sensor": <str>,
    "machine_process": <str>,
    "suggested_operation": <str>,
    "optimal_time_to_perfom_action": <ISO/UNIX timestamp>,
  },
  "generated_plan": [
    {
        "component": <str>,
        "sensor": <str>,

        "machine_process": <str>,

        "suggested_action": <str>,

        "optimal_time_to_perfom_action": <ISO/UNIX timestamp>,

    },

    {

        "component": <str>,

        "sensor": <str>,

        "machine_process": <str>,

        "suggested_action": <str>,

        "optimal_time_to_perfom_action": <ISO/UNIX timestamp>,

    },

    ...,

  ],

  "machine_calibration_suggestions": [

   {

      "component": <str>,

       "sensor": <str>,

       "machine_process": <str>,

       "parameter_value_expected_1": <float>, ...,

         "parameter_value_expected_n": <float>, ...,

   },...,

   {

      "component": <str>,

       "sensor": <str>,

       "machine_process": <str>,

       "parameter_value_expected_1": <float>, ...,

         "parameter_value_expected_n": <float>, ...,
    }
  ],
  "message": <str - This is free text notifying the end users
about relevant information>,
  "arguments": {
    "toolkit_configuration": {
      "kpis_weights":
        {
          "kpi_1": <float>,...
          "kpi_n": <float>
        }
    },
"adaptive_sensorial_network_and_infrastructure_digital_retrofi
tting": <dict>,
    "machinery_operational_profiling": <dict>,
"fault_diagnosis_predictive_maintenance_simulation_digital_twi
n": <dict>}
}
```

A3.5 Data from the Prognostic and Health Management Toolkit

A3.5.1 Description of machine part

```
// MachinePartItem
{
  "machinePartId": {
    "type": "uuid"

    "description": "identifier of a machine part"

  },

  "componentId": {

    "type": "uuid"

    "description": "identifier of a hardware component"

  },

  "WeibullWeight": {

    "type": "float",

    "min": "0.0",

    "max": "100.0",

    "default": "1.0",

    "description": "weight of reliability value of the
component in the referenced machine part"

  }

}

// MachinePart, combines machine-part-items to a whole chain

{

  "machinePartId": {

    "type": "uuid"

    "description": "identifier of a machine part"

  },

  "machinePartItems": {

    "type": "array",

    "minLength": "1",

    "maxLength": "infinite"

    "description": "list of MachinePartItems objects"

  }
```

```
}
```

A3.5.2 Description of machine component

```
// Hardware Component

// Static information of a single hardware component necessary
for degradation calculation

// which should be store in RECLAIM Repository

{

  "componentId": {

    "type": "uuid",

    "description": "identifier of the component,

  },

  "Model": {

    "type": "string",

    "default": "Model_XYZ",

    "description": "model identfier",

    "required": true

  },

  "SerialNumber": {

    "type": "string",

    "default": "12345678",

    "description": "serial number",

  },

  "InstallationTime": {

    "type": "datetime",

    "default": "",

    "description": "time of installation of the machine part
(start of usage)",
  },
  "LastServiceTime": {
    "type": "datetime",
    "default": "",
    "description": "time of last maintenance",
  },
  "DescriptionModel": {
    "type": "string",
    "default": "Weibull",
    "description": "type of degradation model to use",
   },
  "TimeUnit": {
    "type": "string",
    "default": "Second",
    "description": "time resolution of load and degradation
calculation",
  },
  "WeibullBeta": {
    "type": "float",
    "min": "0.1",
    "max": "50",
    "default": 3.0,
    "description": "Weibull beta parameter. Miminal and
maximal values are extreme values where beta still could make
sense."
  },
  "WeibullEta" : {
    "type": "uint32"
    "min": "0",
    "max": "1e9",
    "default": "1e9"
    "description": "Weibull eta parameter in seconds.",
  },
}
```

A3.5.3 Sensor/process data

```
// Sensor/process data
// This can be any data from process which can be referenced
// by table/column combination from RECLAIM Repository
// to a specific load calculator service
{
  "componentId": {
    "type": "uuid",
    "description": "Identifier of a machine part",
  },
  "...": {
    "type": "float|float[]",
    "min": "FLT_MIN",
    "max": "FLT_MAX",
    "default": "0.0",
    "description": "a sensor or processdata column which is
necessary for load calulcation",
  },
}
```

A3.5.4 Load data

```
// Load Caclulation Result

{

  "componentId": {

    "type": "uuid",

    "description": "Identifier of a machine part",

    "required": true,

  },

  "AverageLoad": {

    "type": "float",

    "min": "0.0",

    "max": "1000000.0",

    "default": "1.0",

    "description": "Equipment load can vary in a broad range.
Maximal value is here limited by 1e6."

  }

  "EstimateTime": {

    "type": "uint32",

    "min": "0",

    "max": "4294967295",

    "default": "0",

    "description": "Calculated life time of the component in
TimeUnit of component",

  }

}
```

A3.5.5 Degradation data

```
// Degradation Model Result
```

```
{
  "machinePartId": {
    "type": "uuid"
    "description": "identifier of a machine part "
  },
  "MTTF" : {
    "type": "uint32"
    "min": "0",
    "max": "1e9",
    "default": "1e9"
    "description": "mean time to failure in seconds. Maximum
value of 1 billion corresponds to about 30 years.",
  },
  "SigmaMTTF" : {
    "type": "uint32"
    "min": "0",
    "max": "1e9",
    "default": "1e9"
    "description": "Square root of second moment (standard
deviation) of time to failure (TTF) distribution. Maximum
value of 1 billion corresponds to about 30 years.",
  },
  "MTBF" : {
    "type": "uint32"
    "min": "0",
    "max": "1e9",
    "default": "1e9"
    "description": "mean time between failures in seconds.
Maximum value of 1 billion corresponds to about 30 years.",
  },
  "MRL": {
    "type": "uint32",
    "min": "0",
    "max": "1e9",
    "default": "1e9"
    "description": "mean residual life of used component in
seconds. Maximum value of 1 billion corresponds to about 30
years."
  },
  "Hazard": {
    "type": "uint32",
    "min": "0",
    "max": "1e9",
    "default": "0"
    "description": "hazard function in ppb/second (part per
billion in second). It is probability to fail within next time
unit (e.g. second). Maximum value of 1 billion corresponds to
failure probability of 100% 1 in second."
  },
  "ReliableLife": {
    "type": "uint32",
    "min": "0",
    "max": "1e9",
    "default": 1e9,
    "description": "Lifetime for survival probability of 90%
in seconds. Maximum value of 1 billion corresponds to about 30
years."
  },
  "Wear": {
    "type": "uint32",
    "min": "0",

    "max": "1e9",

    "default": 0,

    "description": "Wear of equipment. Maximum value of 1
billion (1e9) corresponds to 100. Wear percentage is obtained
by following factor: x1e-7"

  },

}
```

A3.6 Data from the Integrated Decision Support Framework for Refurbishment & Remanufacturing Optimization core component

```
{"output_name":
"CERTH.DSFCore."+<pilot_name>+"."+<use_case_name>,

 "present_time": <ISO/UNIX timestamp>,

 "kpi_values": {"with_optimization_"+<individual_kpi_1>:
<float>,

                "with_optimization_"+<individual_kpi_2>:
<float>,...

                "with_optimization_"+<individual_kpi_n>:
<float>,

                "with_optimization_total": <float>,

                "without_optimization_"+<individual_kpi_1>:
<float>,

                "without_optimization_"+<individual_kpi_2>:
<float>,...

                "without_optimization_"+<individual_kpi_n>:
<float>,

                "without_optimization_total": <float>},

 "decisions": [{"machine": <str>,
               "component": <str>,

               "action": <str>,

               "optimal_continuous_variables":
{"parameter_name": <float>,


"parameter_name": <float>,...},

               "start_time": <ISO/UNIX timestamp>,

               "end_time": <ISO/UNIX timestamp>,

               "design_strategy",<str>},

              {"machine": <str>,

               "component": <str>,

               "action": <str>,

               "optimal_continuous_variables":
{"parameter_name": <float>,


"parameter_name": <float>,...},

               "start_time": <ISO/UNIX timestamp>,

               "end_time": <ISO/UNIX timestamp>,

               "design_strategy",<str>},...],

 "message": <str -

              This is free text notifying the end users about
the

              a) most suitable remanufacturing/refurbishment
strategies,

              b) preferable timeframe for implementation of
the strategies,

              c) right components to be
remanufactured/refurbished,

              d) optimal design alternatives.>,

 "arguments": {"dsfcore": {"optimization_horizon_years":
<float>,

                            "kpis_weights": {"kpi_1": <float>,
                                                "kpi_2":
<float>,...
                                                "kpi_n": <float>}

"number_of_optimization_iterations": <int>,
                          },
                "machinery_operational_profiling": <dict>,
                "fault_disgnosis_predictive_maintenance":
<dict>,

"optimization_toolkit_refurbishment_remanufacturing_planning":
<dict>,
                "prognostic_health_management": <dict>,
                "cost_modelling_financial_analysis_toolkit":
<dict>}}
```