




## DIGITAL MANUFACTURING PLATFORMS FOR CONNECTED SMART FACTORIES

### D3.14 Library of Integrated, Interoperable Digital Enablers (Final Version)

Deliverable Id:	D3.14
Deliverable Name:	Library of Integrated, Interoperable Digital Enablers (Final Version)
Status:	Final
Dissemination Level:	PU
Due date of deliverable:	30/06/2020
Actual submission date:	30/07/2020
Work Package:	WP3
Organization name of lead contractor for this deliverable:	INTRASOFT International S.A.
Author(s):	John Soldatos, Nikos Kefalakis
Partner(s) contributing:	EPFL, TTT, MON, MGEP, ENG, ATOS, VTT, FHG

**Abstract:** This deliverable presents the project's approach to integrating different enablers, including the packaging, distribution, and software code management infrastructures to be used. Moreover, it provides a list of digital enablers that will make use of these infrastructures to ease integration and distribution. Furthermore, the deliverable illustrates the project's approach to the interoperability of different automation platforms and components, including semantic interoperability.




	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

## Contents

HISTORY .....	4
Executive Summary .....	5
1 Introduction.....	6
1.1 Scope and Purpose of the Deliverable .....	6
1.2 Relation to Other Deliverables.....	7
1.3 Updates since the last deliverable version.....	7
1.4 Deliverable Structure .....	8
2 Packaging & Availability of Standalone & Integrated QU4LITY Digital Enablers.	9
2.1 Packaging & Integration .....	11
2.1.1 Software Packaging with Docker images.....	11
2.1.2 Container Tool with Docker Compose. ....	11
2.1.3 Code Management with GitHub. ....	12
2.1.4 Repository Management with Docker Hub.....	12
2.2 Standalone Digital Enablers .....	13
2.2.1 DataCROP DDA Platform .....	13
2.2.2 Data Transformation Platform .....	16
2.2.3 Semi-Supervised Fault Identification.....	21
2.2.4 Edge Computing Enabler .....	22
2.2.5 One-click deployment of a disaggregated 5G cloud-native E2E network	23
2.2.6 QU4LITY Cloud Bridge.....	26
2.2.7 Q-Ontology Enabler .....	27
2.2.8 VTT OpenVA .....	28
2.2.9 Secure Identity Directory (SID).....	29
2.2.10 Secure Messaging Board (SMB).....	31
2.2.11 Quality Clearing House (QCH) .....	32
2.2.12 Security Privacy and Trust Framework .....	33
2.2.13 XL-SIEM.....	34
2.2.14 Anonymization Component .....	37
2.3 Integrated Digital Enablers .....	39
2.3.1 Data Transformation Platform + DataCROP .....	39
2.3.2 DataCROP + LDM + Fault Identification.....	41
2.3.3 DataCROP + LDM + QARMA Analytics .....	42

<b>QU4LITY</b>	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

2.3.4	DataCROP + LDM + Secure Messaging Board (SMB) .....	44
2.3.5	QU4LITY Ontology + Visual Component Simulation.....	45
2.3.6	Anomaly Detection for Assembly Process / Shift-In Movement .....	46
2.4	QU4LITY Open-Source Digital Enablers .....	47
3	QU4LITY Platform Interoperability .....	49
3.1	QU4LITY Common Standards used .....	49
4	QU4LITY Semantic Interoperability .....	52
4.1	QU4LITY Semantic Data Models and Ontologies.....	52
4.1.1	QU4LITY Semantic Data Models .....	52
4.1.2	QU4LITY Ontologies.....	53
4.1.3	Application of QU4LITY Data Models and Ontologies .....	55
4.1.4	Semantic Models and Vocabularies implementation .....	57
4.2	Lightweight Digital Models for ZDM (FAR-EDGE/PROPHECY data model) ..	58
4.3	Lightweight Digital Models .....	58
4.3.1	Description and Usage .....	58
4.3.2	Relation with the Reference Architecture .....	59
4.3.3	Dependencies .....	59
4.3.4	Availability .....	59
4.3.5	Installation guidelines .....	60
5	Conclusions .....	61
6	References .....	62
	List of figures .....	63
	List of tables .....	64
	List of Abbreviations .....	65
	Appendix I – Digital Enablers’ Docker Compose Scripts .....	66
	DataCROP.....	66
	Cloud Infrastructure.....	70
	Appendix II – Native Installation Scripts .....	72
	XL-SIEM .....	72
	Partners: .....	75

	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

## HISTORY

Version	Date	Modification reason	Modified by
0.1	17/05/2021	Initial Table of Contents presented to the involved partners.	INTRA
0.2	31/05/2021	First Draft of contribution	EPFL
0.3	31/05/2021	First Draft of contributions	TTT
0.4	03/06/2021	Integrated contributions, added packaging and integration	INTRA
0.5	11/06/2021	Added Cloud Bridge, Q-Ontology, SID, SMB, QCH Q4LITY enablers	ENG
0.6	11/06/2021	Second version of contributions	TTT, FHG
0.7	14/06/2021	Integrated contributions, added DataCROP and Lightweight Digital Models	INTRA
0.8	15/06/2021	Added Semi-Supervised Fault identification digital enabler	MON
0.9	20/06/2021	Integrated contributions, added Reference Architecture content and introduction	INTRA
0.10	24/06/2021	Added QU4LITY Ontology + Visual Component Simulation, common standards survey	EPFL
0.11	29/06/2021	Added VTT OpenVA, KUBE5G	INTRA, VTT, TID
0.12	30/06/2021	Added Data Transformation Platform, DTP + DataCROP Integration	MGEP
0.13	01/07/2021	Integrated contributions, added integrated components for DataCROP, LDM, Fault Identification and QARMA	INTRA
0.14	16/07/2021	Added SPT and XL-SIEM	ATOS
0.15	18/07/2021	Integrated contributions, added executive summary and conclusions	INTRA
0.16	21/07/2021	Preparation of version of quality control by QU4LITY partners	INTRA
0.17	28/07/2021	Internal review of the document	FHG-ILT
0.18	29/07/2021	Implementation of Revisions following quality control and reviews; Preparation of version for delivery to the Project Coordinator & Submission	INTRA
0.19	/07/2021	Approved and submitted from project coordinator	ATOS

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

## Executive Summary

QU4LITY has specified and implement a novel approach to quality management and Zero Defects Manufacturing (ZDM), namely an intelligent and Autonomous Quality (AQ) paradigm. The realization of these paradigm relies on the deployment and usage of a diverse collection of components which are mapped in different domains of the QU4LITY reference architecture. To enable the reusability of these components a packaging & integration methodology have been established which is provided in this deliverable along with platform and semantic interoperability directions. Moreover, in this deliverable we list the different WP3 digital enablers most of which are following the packaging and integration methodology described. Additionally, we provide examples of integrated digital enablers which are combined to offer a more complete solution. Several of the digital enablers are also offered as open-source software thru the Git repository that have been established for the QU4LITY project.

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

## 1 Introduction

### 1.1 Scope and Purpose of the Deliverable

QU4LITY is developing and validating a pioneering approach to quality management and Zero-Defect Manufacturing (ZDM), which is characterized by autonomy and intelligence that minimize manual error prone operations. The project's approach is empowered by the notion of a fully digital shopfloor, where digital data and ICT technologies are exploited to increase the accuracy and proactive of quality management and ZDM processes. In this direction, WP3 of the project is developing and validating a set of digital enablers, which empower the transformation of conventional quality management processes to fully digital processes i.e., processes that are driven and control in the cyber part of modern Industry 4.0 compliant factories.

As part of earlier deliverables of WP3, various digital enablers for ZDM have been specified, designed, and sometimes implemented, including Bigdata platforms, Machine Learning and AI (Artificial Intelligence) algorithms, Edge/Fog Nodes and Devices, Blockchain infrastructures for ZDM and more. Each of these enablers is destined to provide a subset of functionality of the project's Autonomous Quality (AQ) system, such as the extraction of knowledge about quality processes, the execution of automation and control functions close to the field, the sharing of data in secure and trustworthy ways, and more. However, none of these enablers is enough for implementing full-fledged, end-to-end AQ solutions in-line with the QUALITY Reference Architecture (RA). Rather, the implementation of end-to-end solutions requires the packaging and integration of more than one enabler in a single solution configuration. In several cases, this packaging may include additional manufacturing components and applications such as digital simulations or augmented reality components for human centred manufacturing. Hence, there is a need for an infrastructure that will facilitate:

- The integration of two or more digital enablers in ZDM/AQ solutions, in ways that facilitate the deployment, release, packaging and distribution of the results.
- The interoperability across different modules and platforms that comprise a ZDM/AQ solution, given that the various components/platforms tend to produce digital data in different formats and based on different semantics.

The present deliverable is aimed at describing the project's solution for integration, packaging, and distribution of QU4LITY's digital manufacturing solutions, as well as the project's approach to the interoperability of different platforms and components. Specifically, the deliverable presents:

- A library of digital enablers that will following the integration and packaging principles of the project, including the specified infrastructures and tools. This library will provide inputs to several other development activities of the project

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

such as the market platform (in WP8) and the pilots that will use enablers from the library (in WP7).

- The QU4LITY Interoperability concepts and solutions, including (common) digital models for semantic interoperability of distributed components and middleware solutions for the syntactic interoperability of diverse platforms.

## 1.2 Relation to Other Deliverables

The deliverable is closely linked to all the WP3 deliverables that produce digital enablers for ZDM. These digital enablers will be integrated and packaged following the guidelines and using the tools established as part of the first version of this deliverable (D3.13). As such, the deliverable is relevant to the following WP3 deliverables that have been already delivered:

- D3.2 Connectivity Technologies for Autonomous Quality (Final version).
- D3.4 HPC and Cloud Resources for ZDM (Final version).
- D3.6 BigData and Analytics Infrastructure (Final version).
- D3.8 Fog Nodes and Edge Gateways for ZDM deployments (Final version).
- D3.10 QU4LITY SPT Framework (Final version).
- D3.12 Permissioned Blockchain for ZDM (Final version).

Also, the deliverable leverages inputs and results from:

- D2.8 Standards Compliance and Interoperability Specification (Final Version) Report illustrating standards compliance and interoperability needs.
- D2.10 QU4LITY Digital Models and Vocabularies (Final Version), which prescribes digital models and vocabularies used in the QU4LITY systems and pilots. These digital models will also provide a foundation for the project's semantic interoperability approach.
- D2.12 Reference Architecture and Blueprints (Final Version), which provides the overall ZDM system integration concept of the project. The results of the present deliverable are compatible with D2.12 and support the integration of solutions in-line with the RA of the project.

## 1.3 Updates since the last deliverable version

This is the second and final version of the "Library of Integrated, Interoperable Digital Enablers" deliverable. In this version almost all of the content is new or updated from the previous one. More specifically the following changes have been performed:

Removed content:

- Packaging & Integration of QU4LITY Digital Enablers section have been mostly removed. In order to get information for the following subjects please visit Section 2.1, 2.2 and 2.4 respectively from D3.13.
  - Enablers integration concept and elements of the integration solutions
  - Integration Requirements with partners preferences on development, deployment and integration infrastructures.

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

- Deployment infrastructure for edge computing
- Deployment infrastructure for DLT Services

Updated, new content:

- Updated the Integration Infrastructure description.
- Updated QU4LITY integration concept.
- A complete update of the list of wp3 standalone digital enablers with a short description, availability, installation guidelines and relevant documentation.
- Added a list of integrated Digital Enablers with examples of interoperable enablers their possible usage, integration methodology and installation guidelines.
- A complete update of the QU4LITY platform interoperability
- A complete update of the QU4LITY semantic interoperability

## 1.4 Deliverable Structure

The deliverable is structured as follows:

- Section 2 following this introductory section presents the packaging and availability of standalone and integrated digital enablers along with the packaging and integration methodology.
- Section 3 provides a description of the project's platform interoperability solution focusing on the common services, infrastructures and standards.
- Section 4 focuses on the presentation of the project's semantic interoperability solutions based on the semantic digital models that are specified and used in the project.
- Section 5 is the concluding section of the deliverable.



QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

## 2 Packaging & Availability of Standalone & Integrated QU4LITY Digital Enablers

Deliverable D2.12 presented the last version of the QU4LITY reference architecture (QU4LITY-RA), which is also illustrated in Figure 1 below. This architecture provides a high-level blueprint for the integration of different solutions, over digital infrastructures. The integration approach of the project does not pose any restriction regarding the integration of different components based on the RA. Specifically, the QU4LITY software packaging and integration approach will leverage OS-level virtualization to deliver different software modules in the form of packages ("containers"). Individual QU4LITY components will be therefore isolated from one another within specific containers, each one bundling their own software, libraries and configuration files.

Using the proposed infrastructure, any set of QU4LITY modules can communicate with each other through well-defined channels and without restrictions. As such the integration infrastructure and tools of the previous paragraphs provide flexibility in integrating different modules in the scope of industrial use cases.

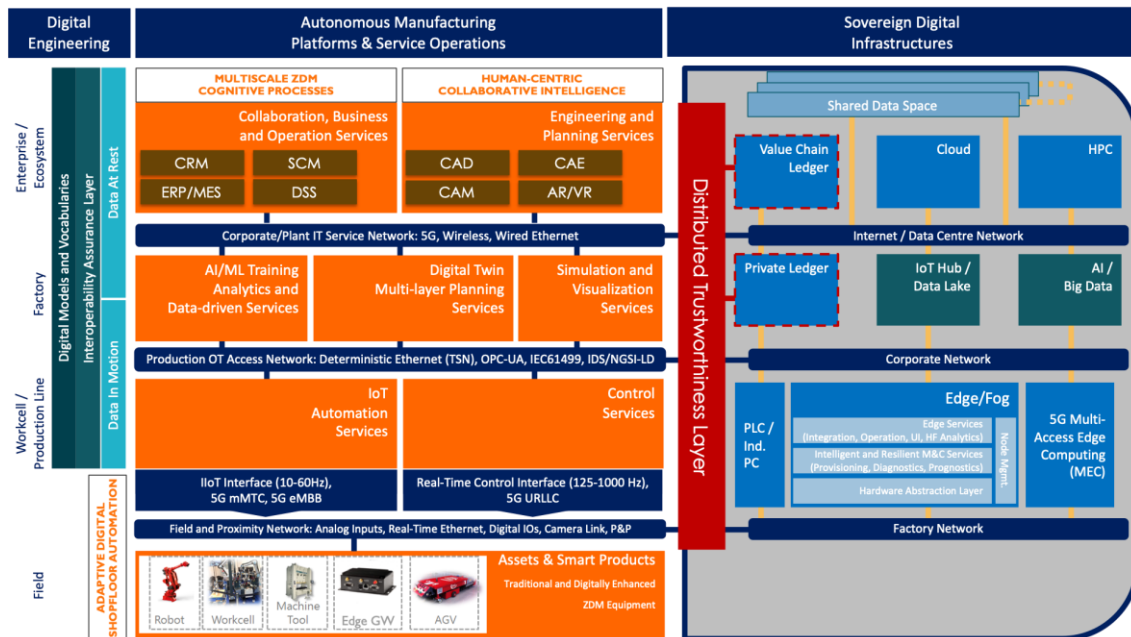


Figure 1 QU4LITY Reference Architecture (Q-RA) (D2.12)

In Table 1 below we can see an overview of the different digital enablers offered in the context of WP3 along with their QU4LITY Reference Architecture Mapping.

Component Name	Owner	Task	QU4LITY Mapping	RA	License
One-click deployment of a disaggregated 5G cloud-native E2E network	TID	T3.1	Corporate/Plant Service Network	IT	Apache-2.0

<b>QU4LITY</b>	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

Component Name	Owner	Task	QU4LITY Mapping	RA	License
TSN fieldbus network	CEA	T3.1	Field and Proximity Network		SaaS/ Proprietary
Visual quality control Training-aaS on HPC	JSI	T3.2	Data Lake / Big Data Analytics Infrastructure		Open Source
QU4LITY Cloud Infrastructure	ENG	T3.2	Cloud		AGPL-3.0
Q-Ontology Enabler	ENG	T3.2	Cloud		Open Source
DataCrop DDA Platform	INTRA	T3.3	Data-driven Modelling and Learning Services		Apache-2.0
Data Driven RUL Calculation	ATLAS	T3.3	Data-driven Modelling and Learning Services		Proprietary
Model Driven RUL Calculation	ATLAS	T3.3	Data-driven Modelling and Learning Services		Proprietary
Quantitative Association Rule Mining (QARMA)	INTRA	T3.3	Data-driven Modelling and Learning Services		Proprietary
Anomaly Detection for Quality Control	TNO	T3.3	Data-driven Modelling and Learning Services		Proprietary
Analytics for In-Line Sensor Data Visualization	FHG-IGD	T3.3	Data-driven Modelling and Learning Services		Proprietary
Image analyzer for surface inspection	FHG-ILT	T3.3	Data-driven Modelling and Learning Services		TBD
Improved Failure Classification Enabler	TUDO	T3.3	Data-driven Modelling and Learning Services		Proprietary
OpenVA	VTT	T3.3	Simulation and Human-centric Visualization Srv.		BSD/ Proprietary
Fault Identification	MGEP	T3.3	AI and Big Data		Proprietary
Nerve Blue	TTT	T3.4	Access & Smart Products		Proprietary
Savvy Smart Box	IDEKO	T3.4	Access & Smart Products		Proprietary
FOOTPRINT	UNP	T3.4	Access & Smart Products		TBD
Security Privacy and Trust Framework	ATOS	T3.5	Distributed Trustworthiness Middleware		TBD
XL-SIEM	ATOS	T3.5	Distributed Trustworthiness Middleware		GPL/ Proprietary
Context Awareness Framework	ATB	T3.5	Converters for Interoperability		Proprietary

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

Component Name	Owner	Task	QU4LITY Mapping	RA	License
MASAI	ATOS	T3.5	Converters Interoperability	for	TBD
Data Transformation Platform	MGEP	T3.5	AI and Big Data		MIT
DApp: Secure Identity Directory (SID)	ENG	T3.6	Value Chain Ledger Private Ledger		Apache-2.0
DApp: Secure Messaging Board (SMB)	ENG	T3.6	Value Chain Ledger Private Ledger		Apache-2.0
DApp: Quality Clearing House (QCH)	ENG	T3.6	Value Chain Ledger		Apache-2.0

Table 1 List of the QU4LITY digital enablers

The following sections provides an overview of the proposed packaging and integration solution along with the list of QU4LITY digital enablers in standalone and integrated interoperable flavours that are offering it. A short description of the digital enablers/integrated solutions is provided along with their reference architecture placement, their availability and installation instructions where available.

## 2.1 Packaging & Integration


In this section you can find an overview of the packaging and integration plan identified in the first version of this deliverable (D3.13) and more specifically in section 2.3.

### 2.1.1 Software Packaging with Docker images.

The preferred packaging methodology is Docker containers which are used, in most of the cases, to package and distribute the different Digital Enablers. Docker is an open platform for developing, shipping, and running applications. With Docker, an infrastructure can be managed in the same way's applications are managed. Docker [1] offers shipping, testing, and deploying methodologies easily and quickly, where time between writing code and running it in production can be significantly reduced.

### 2.1.2 Container Tool with Docker Compose.

As briefly mentioned above Docker Compose is a tool for defining and running multi-container Docker applications. It uses YAML files to configure the application's services and performs the creation and start-up process of all the containers with a single command. The docker-compose.yml file is used to define an application's services and includes configuration options. In quality as the preferred container runtime management method was Docker Compose every digital enabler will be accompanied by a docker-compose.yml file which will facilitate its installation. Additionally, different collections of interoperable digital enablers that will be used as solutions for the QU4LITY use cases will be provided as ready to install docker-compose.yml files.

	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

Information on how to edit a docker-compose.yml file can be found at Docker Docs [1] and more specifically at the Get started with Docker Compose<sup>1</sup>

### 2.1.3 Code Management with GitHub.

The Open Source (OS) Digital Enablers source code is offered thru GitHub<sup>2</sup> repository. A QU4LITY organization has been set up at GitHub<sup>3</sup> where every OS Digital Enabler which is developed within the context of QU4LITY project can have its own repository.

Guidelines on how to use GitHub can be found in the GitHub Guides<sup>4</sup> and more specifically for a beginner the Hello World guide<sup>5</sup> could be used.

In most of the cases the different Digital Enablers had already existing code repositories (available in other GitHub branches). In that cases repository mirroring mechanisms have been employed to ensure access to the components from the project's GitHub. Mirroring<sup>6</sup> a repository from another location enables the user to maintain an updated copy of the active workspace including getting updates from the original. Mirroring can be performed periodically in order to push any updates from the mirrored repository.

To generate a new repository under the QU4LITY organization a Digital Enabler owner should contact Mr. Nikos Kefalakis<sup>7</sup> with their GitHub username (or list of user names if they wish to generate a dedicated team for their organization) and the name of the repository(s) to be generated in order to become their Administrators (level Admin), Maintainers (level Maintain) or Authors (level Write) depending on the team organization.

### 2.1.4 Repository Management with Docker Hub.

The Open Source (OS) Digital Enablers containers can be hosted in Docker Hub<sup>8</sup>. A QU4LITY organization has been set up at Docker Hub<sup>9</sup> where every OS Digital Enabler which is developed within the context of QU4LITY project can have its own repository.

To generate a new artifact repository under the QU4LITY organization a Digital Enabler owner should contact Mr. Nikos Kefalakis<sup>10</sup> with their Docker Hub username (or list of usernames if they wish to generate a dedicated team for their organization) and the name of the repository(s) to be generated in order to become their Administrators (level Admin) or Authors (level Read-Write) depending on the team organization.

<sup>1</sup> <https://docs.docker.com/compose/gettingstarted/>

<sup>2</sup> <https://github.com/>

<sup>3</sup> <https://github.com/qu4lity>

<sup>4</sup> <https://guides.github.com/>

<sup>5</sup> <https://guides.github.com/activities/hello-world/>

<sup>6</sup> <https://docs.github.com/en/github/creating-cloning-and-archiving-repositories/creating-a-repository-on-github/duplicating-a-repository#mirroring-a-repository-in-another-location>

<sup>7</sup> [Nikos.KEFALAKIS@intrasoft-intl.com](mailto:Nikos.KEFALAKIS@intrasoft-intl.com)

<sup>8</sup> <https://hub.docker.com/>

<sup>9</sup> <https://hub.docker.com/orgs/qu4lity>

<sup>10</sup> [Nikos.KEFALAKIS@intrasoft-intl.com](mailto:Nikos.KEFALAKIS@intrasoft-intl.com)

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

## 2.2 Standalone Digital Enablers

### 2.2.1 DataCROP DDA Platform

#### 2.2.1.1 Description and Usage

The DataCROP (Data Collection Routing & Processing) platform, initially developed in the H2020 FAR-EDGE & PROPHECY projects, is an IoT platform which enables the collection, routing, pre-processing and annotation of collected data to facilitate the configuration and management of analytics algorithms. DataCROP supports multiple Edge Gateways for local data collection, manipulation and processing and a cloud tier for global data manipulation and processing for data coming from the different Edge Gateways to provide more complex and consolidated analytics from the whole infrastructure (multiple Edge Gateways).

A lightweight version of the DataCROP solution (single Edge Gateway) have been used in the RiaStone pilot which was deployed to facilitate the QARMA4Industry algorithm for estimating RUL. More details on the DataCROP platform can be found in section 3.1 of QU4LITY deliverable D3.6 "BigData and Analytics Infrastructure (Final Version)".

#### 2.2.1.2 Relation with the Reference Architecture

As mentioned in D2.12 the DataCROP analytics management platform is placed at the "AI and Big Data" layer (see Figure 1 above) based on its cross-cutting functions associated with data routing and industrial analysis.

#### 2.2.1.3 Dependencies

DataCROP platform has the following environment and software dependencies:

- Confluent Platform  $\geq 4.1.1$
- MongoDB  $\geq 3.6.4$
- Node.js  $\geq 10.1.0$
- npm  $\geq 5.6.0$

#### 2.2.1.4 Availability

DataCROP is an Open-Source software and is offered both as source code at GitHub<sup>11</sup> but as well as a containerized solution at Docker Hub<sup>12</sup>. In the following two sub-sections, we provide details for this availability.

##### 2.2.1.4.1 GitHub Distribution

DataCROP source code is offered under the Apache License 2.0 and its components are available from GitHub<sup>13</sup>. DataCROP is consisted of the following components:

<sup>11</sup> <https://github.com/qu4lity/data-crop>

<sup>12</sup> <https://hub.docker.com/u/faredge>

<sup>13</sup> <https://github.com/qu4lity/data-crop>

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

- Open API for Analytics
- Edge Analytics Engine
- Model Repository
- Digital Models
- Analytics Processors
- Analytics Dashboard
- MQTT Data Publishers
  - File
  - Random Data

#### 2.2.1.4.2 Docker Distribution

DataCROP dockerized components are offered thru Docker Hub and various deployment options are also available by offering the equivalent YAML files for Docker Compose and Docker Swarm.

DataCROP Docker Hub availability is provided below:

- Open API for Analytics:
  - <https://hub.docker.com/repository/docker/faredge/open-api-for-analytics>
- Edge Analytics Engine:
  - <https://hub.docker.com/repository/docker/faredge/edge-analytics-engine>
- Analytics Dashboard:
  - <https://hub.docker.com/repository/docker/faredge/analytics-dashboard>
- Model Repository
  - <https://hub.docker.com/repository/docker/faredge/model-repository>
- MQTT Data Publishers:
  - Publish data from File:
    - <https://hub.docker.com/repository/docker/faredge/mqtt-file-data-publisher>
  - Publish Random Data:
    - <https://hub.docker.com/repository/docker/faredge/mqtt-random-data-publisher>

#### 2.2.1.5 Installation guidelines

The core infrastructure of DataCROP is deployed effortlessly by taking advantage of the facilities offered by Docker. The platform deployment scripts can be found on GitLab<sup>14</sup>.

For a test deployment the following scripts and sample data are offered:

- **INSTRUCTIONS.md**: a text file containing instructions on how to deploy the platform

<sup>14</sup> <https://github.com/qu4lity/data-crop>

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

- **test.yml:** a YAML file containing the configurations of the various Docker containers (Docker images, environment variables, networking, data volumes configuration etc.) see Table 2 in Appendix I – Digital Enablers’ Docker Compose Scripts below.
- **processors:** a folder containing sample algorithms imitating the ML toolkit’s algorithms behavior.
- **data:** folder containing sample data to prepopulate the databases with for demonstration purposes

By following the instructions, one may both deploy and undeploy the various components DataCROP infrastructure. To do so, Docker Compose is being used. Compose is a tool for defining and running multi-container Docker applications. A YAML file is being employed to configure all application’s services. Then, with a single command, one may create and start all the services from the aforementioned configuration<sup>15</sup>.

#### Start Everything.

Run the following command.

```
docker-compose -f test.yml -p test up -d
```

#### Deployment Test.

Run the following command.

```
docker ps --filter name=test --format "table {{.Image}}\t{{.Names}}"
```

#### Stop Everything.

Run the following command.

```
docker-compose -f test.yml -p test down
```

#### Clean Everything.

Run the following command. (\*\*at your own risk\*\*).

```
docker system prune --volumes
```

Please note that this command erases all the unused system volume so execute it only if you are sure useful system volumes are not going to be deleted.

<sup>15</sup> <https://docs.docker.com/compose/>

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

## 2.2.2 Data Transformation Platform

### 2.2.2.1 Description and Usage

The Data Transformation Platform consist of several components/servers that allow the construction of converters for interoperability. These converters for interoperability are software artifacts that enable syntactic interoperability between heterogeneous systems by means of protocol and data format translation. The digital enabler offered in this deliverable is an infrastructure solution compose of three servers. In addition, two APIs converters are provided as an example on the usage of the infrastructure.

The infrastructure of the data transformation platform consists of a Docker solution with three servers:

1. An Enterprise Service Bus (ESB) server that enables the deployment of software artifacts for the conversion of data between APIs, IoT gateways and platforms. The solution proposes the usage of WSO2 (*Web Services Oxygenated 2*).
2. An edge broker server for message publication and subscription. This broker enables event driven architectures for the converters that need to integrate APIs that need such infrastructure. The solution proposes the usage of RabbitMQ.
3. A Node-RED server is also provided. Node-RED has similar integration capabilities to those of WSO2 but presents a much more user-friendly interface (visual representation) with a Web browser and enables the integration with OT technology such as OPC-UA.

The Data Transformation Platform is used in the Mondragon pilot which along the MongoDB repository presented in section 2.3.1, enables the construction of a messaging architecture to collect information from assets. In addition, APIs are offered to consume data stored in the repository. More details on the Data Transformation Platform can be found in Deliverable D5.7 "QU4LITY Digital Platforms Open APIs".

### 2.2.2.2 Relation with the Reference Architecture

The Data Transformation Platform is placed at the "IoT Hub/Data Lake" to support the "AI and Big Data" layer (see Figure 1 above) based on its cross-cutting functions associated with the interoperability assurance layer.

### 2.2.2.3 Dependencies

Data Transformation Platform has the following environment and software dependencies:

- WSO2 Integrator >= 6.5.0
- RabbitMQ >= 3.8.17
- Node-RED >= 1.3.5
- Node.js >= 12.22.1



QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

#### 2.2.2.4 Availability

This digital enabler is Open Source software under MIT license and is offered as a containerized solution at GitHub. In the following sub-section, we provide details for its availability.

##### 2.2.2.4.1 GitHub Distribution

The components for this enabler are images available in Docker Hub and the containerized solution is available from GitHub. Links to the relevant repositories are provided below:

- Data Transformation Platform
  - <https://github.com/qu4lity/data-transformation-platform>

#### 2.2.2.5 Installation guidelines

The core infrastructure of the Data Transformation Platform is deployed effortlessly by taking advantage of the facilities offered by Docker. The platform deployment scripts can be found on GitHub<sup>16</sup>.

For a test deployment the following scripts and sample data are offered:

- **README.md**: a text file containing instructions on how to deploy the platform
- **Docker-compose.yml**: a YAML file containing the configurations of the various Docker containers (Docker images, environment variables, networking, data volumes configuration etc.).
- **node-red**: a folder holding Node-RED's Dockerfile and the flows provided as examples that offer an endpoint by means of a REST API, collect the data provided by the API consumer and sends the data through the edge broker. See section 2.2.2.6 for more detail.
- **wso2**: a folder holding the data converters constructed with WSO2 as an example. See section 2.2.2.6 for more detail.

By following the instructions given in the README file, one may both deploy and undeploy the various components of the integrated Digital Enabler. To do so, Docker Compose is being used. Compose is a tool for defining and running multi-container Docker applications. A YAML file is being employed to configure all application's services. Then, with a single command, one may create and start all the services from the aforementioned configuration<sup>17</sup>.

#### 2.2.2.6 Documentation


Two converters are provided along the digital enabler to show the capabilities of the solution.

The first is a converter for WSO2 Integrator built with the Eclipse plugging Integration Studio. In this example we simulate a system that offers an endpoint that consumes XML messages. For that we provide a Node-RED flow endpoint called

<sup>16</sup> <https://github.com/qu4lity/data-transformation-platform>

<sup>17</sup> <https://docs.docker.com/compose/>



	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

The message transformation is performed in this case by the Data Mapper component (see Figure 5). This component enables schema import for input and output and a graphical interface to map attributes in the messages.

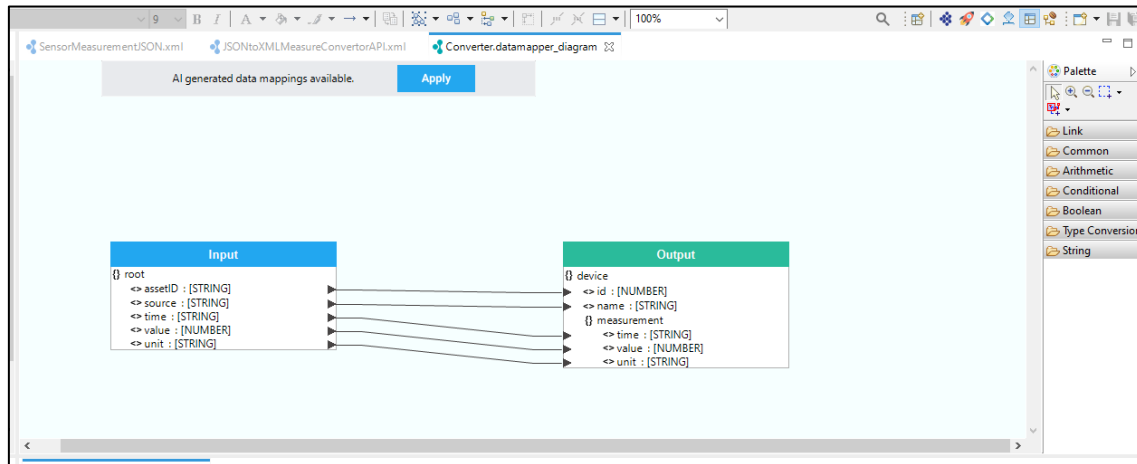


Figure 5 Data Mapper

Additionally, the example offers a Node-RED flow that acts as the API consumer. The flow is called ServiceConsumerJSON.

To run this example the Docker containers provided in GitHub have to be run and the Inject node from the ServiceConsumerJSON activated.

The second converter is built with Node-RED. In this example we simulate a situation where a message with three arrays of 300 elements each is converted in 300 messages each holding the sampleID, the frequency phase and the amplitude of a given sample. Those messages are published in a message broker. The example provides a Node-RED flow that is subscribed to the broker to collect those messages.

The input message is shown in Figure 6. The figure is trunk presenting only some amplitude values. SampleId, frequency and amplitude arrays hold 300 values each. A position in each array determines a sample for the signal represented. Additionally, the message presents information about the asset and the starting timestamp.

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

```
{
  "asset": {
    "assetID": "12345",
    "source": "Signal_sensor_1"
  },
  "measurements": [
    {
      "time": "2021-06-15T02:28:03.743Z",
      "series": {
        "sampleID": [
          "frequency": [
            "amplitude": [
              0.000003099834,
              0.000004028436,
              0.000005538984,
              0.000005540319,
              0.000002965311,
              0.000002916245,
              0.000003928019,
              0.000002902876,
              0.000001947737,
              0.000001835289,
              0.000003632256,
              0.00001812258,
              0.00003949975,
              0.00000586541,
              0.000004762631,
              0.000007493253,
              0.00001087917,
```


Figure 6 Input Message

The output of the converter are 300 messages. One for each sample. Figure 7 presents one of those samples.

```
payload: object
sampleID: 201
frequency: 270
amplitude: 0.000008103253
timestamp: "2021-06-15T02:28:03.743Z"
device: "12345"
source: "Signal_sensor_1"
timestamp: "2021-06-15T02:28:03.743Z"
```

Figure 7 Output Sample Message

The converter is the Node-RED flow called Sampling Example shown in Figure 8. The converter offers an endpoint at <http://node-red:1880/sampling>. The converter After that it divides the arrays and combines the elements in single messages that are published in the qu4lity node (RabbitMQ).

	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

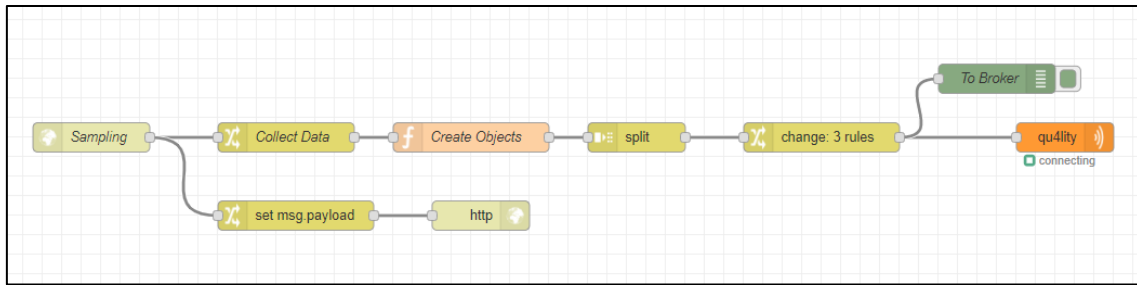


Figure 8 Sampling Converter in Node-RED

In order to test it, an additional tab contains a client that consumes the sampling converter endpoint. Moreover, it contains a flow that consumes and prints AMQP messages.

## 2.2.3 Semi-Supervised Fault Identification

### 2.2.3.1 Description and Usage

The Fault Identification asset is an algorithm for detecting anomalies in time-series. This algorithm is called NullSpace. In D3.6 more information about this algorithm can be found. In quality prediction, when there is no labelled data, changes in the process may show that the quality of the product is not suitable. Since the emerging solutions for zero waste manufacturing are starting to be online, the lack of labelled data is a problem. But the fact that usually, the manufacturing industries are good at manufacturing the parts, usually, there are tons of data that resemble the correct behaviour of the process. Semi-Supervised learning can be used to model a normality and be able to detect when something stands out from this normality.

While supervised learning has labels to train with for all the working scenarios, Semi-Supervised learning, is only trained with one of these labels. Semi-supervised learning is trained using only information from one of the classes, correct operation class in our case. This fact is also known as one-class classification. This enabler makes the semi-supervised process available.


### 2.2.3.2 Relation with the Reference Architecture

As mentioned in D3.6 the Fault Identification enabler is placed at the “AI and Big Data” layer (see Figure 1 above. Moreover, it is integrated in the DataCrop platform, but due to the proprietary nature of the enabler, it is not delivered with the DataCrop platform itself.

### 2.2.3.3 Dependencies

Semi-Supervised Fault identification platform has the following environment and software dependencies:

- python >= 3.6
- Conda >= 4.4

	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

- scipy >= 1.4
- Numpy >= 1.9
- FastAPI >=0.62
- Uvicorn >= 0.12

#### **2.2.3.4 Availability**

The code of this solution is proprietary, so it is not publicly available. Contact Mondragon University for more information.

#### **2.2.3.5 Installation guidelines**

All the dependencies are fulfilled within miniconda docker file so run the following command.

```
docker pull continuumio/miniconda3
```

FastAPI and Uvicorn are not installed within this docker container, so you need to install those dependencies into this docker.

```
Pip install fastapi
Pip install uvicorn[standard]
```

Using this command every dependency is fulfilled. Meaning, that once you get the code, you can run the system. This system enables a web framework built with fastAPI. To run the application run the following command.

```
uvicorn faultID:app --reload
```

With this command you build the application which can make predictions for fault identification.

### **2.2.4 Edge Computing Enabler**

#### **2.2.4.1 Description and Usage**

The edge computing enabler provides the possibility to host applications at the edge of the network (i.e., direct at the machine) and perform (potentially in real time) operations using the data directly from the machine. The advantage of using edge computing at the machine is that you have direct access to the data and can potentially also (in real-time) provide data to the machine back and control the actuators on the machine. The latency of sending data into the cloud is not applicable here.

#### **2.2.4.2 Relation with the Reference Architecture**

This enabler forms the basis of the Edge/Fog part of the QU4LITY Reference Architecture. It has a close connection with all the other layers in the reference architecture, especially with the communication part and the Automation Manufacturing Platforms & Service Operations, as many of the enablers coming from this part can be hosted (partly) on the edge, enabling it directly to be close to the machines and thus to the data. Dependent on the applications hosted on the edge,

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

the data is received from specific sensors on the machines and application specific data is returned either directly to the actuators of the machine or the other applications.

#### **2.2.4.3 Dependencies**

There aren't explicit dependencies on the edge computing enablers. It depends on the interfaces that are available on the edge solutions and the internal software that is running on the edge.

#### **2.2.4.4 Availability**

A free trial of the TIAG edge device, Nerve Blue, can be acquired from the product website<sup>18</sup>.

#### **2.2.4.5 Installation guidelines**

A quick start guide and documentation regarding Nerve Blue can be found on the following nerve website<sup>19</sup>

#### **2.2.4.6 Documentation**

More details about the usage of the edge devices are introduced in WP3 deliverable D3.8 with applications deployed to the edge devices in multiple pilots and experimental facilities.

### **2.2.5 One-click deployment of a disaggregated 5G cloud-native E2E network**

#### **2.2.5.1 Description and Usage**

One-click deployment of a disaggregated 5G cloud-native E2E network is used for bootstrapping K8s automatically on top of Ubuntu 18.04 LTS VMs in order to quickly get a telco K8s-learning playground.

#### **2.2.5.2 Relation with the Reference Architecture**

One-click deployment of a disaggregated 5G cloud-native E2E network is placed at the "Corporate/Plant IT Service Network" layer (see Figure 1 above) based on the network infrastructure it provides in order to facilitate other QU4LITY digital enablers deployments.


#### **2.2.5.3 Dependencies**

One-click deployment of a disaggregated 5G cloud-native E2E network core dependencies are:

- Ansible
- Kubernetes

<sup>18</sup> <https://www.tttech-industrial.com/products/nerve/free-trial/>

<sup>19</sup> <https://docs.nerve.cloud/>

	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

- terraform

#### 2.2.5.4 Availability

One-click deployment of a disaggregated 5G cloud-native E2E network is available on QU4LITY Github<sup>20</sup> as an open-source project.

#### 2.2.5.5 Installation guidelines

##### Repo Structure

- **vmtools**: A bunch of VM management helper scripts used to bootstrap the virtual infra in KVM
- **ansible**: Ansible playbooks to automate the deployment of K8s
- **pac**: Pipelines for deployment automation, specific to each project using this repo (eg. 5glab)
- **terraform**: Preliminary support for deployment in OpenNebula VMs
- **docs**: Miscellaneous documentation regarding networking, load balancer and storage topics

##### Getting Started

###### KVM Host networking setup

The `kvm.yaml` playbook supports setting up the networking infrastructure of a KVM host with the bridging necessary for supporting Intel Multus later in the bootstrap playbook.

Remember to configure `bridge_name`, `bridge_gw` and `bridge_ip` ansible variables within inventory file before provisioning the network setup on the baremetal machine.

As with the bootstrap playbook, setting up inventory is needed prior to running the playbook.

```
ansible-playbook -i inventory kvm.yaml
```

This `kvm.yaml` playbook will install the minimum necessary packages and runs the role `kvmhost` that prepares the pipes on the server (bridge, and libvirt interfaces) in order to launch the cluster. The machine will reboot and then you are ready to initiate the K8s cluster bootstrap setup.

###### Installing K8s


`vmtools/vminstall` is a quick way to generate a minimum KVM image that can act as worker and master. See `vmtools/README.md` to check all details about the tools to manage VMs.

Example:

```
vminstall base
```

<sup>20</sup> <https://github.com/qu4lity/tid-kube5g>



	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

will generate a qcow image named *base*, this image can be later cloned with following commands:

```
vmclone k8s base
vmclone k8sworker1 base
```

These VMs can be later provisioned with the minimal necessary to run K8s: *ansible/bootstrap.yml* is the ansible-playbook that gets you most of what you need to set up a minimal environment to start playing with k8s.

**Important Note:** The qcow image (e.g. *base*) has associated an xml file with the definition associated to the VM when provisioning that image. For kubertelconetes virtual environment we recommend to add two interfaces: one macvtap interface and one interface of type network. This can be done by editing the xml file of the base image. Example: `virsh edit --domain base`

At the network-side of the VM config we mapped one macvtap interface:

```
<interface type='direct' trustGuestRxFilters='yes'>
  <mac address='52:54:00:25:30:3b' />
  <source dev='5gnow' mode='bridge' />
  <model type='virtio' />
  <address type='pci' domain='0x0000' bus='0x01' slot='0x00'
function='0x0' />
</interface>
```

And we include one additional interface for external k8s networks:

```
<interface type='network'>
  <mac address='52:54:00:81:77:68' />
  <source network='5gphysical' />
  <model type='virtio' />
  <address type='pci' domain='0x0000' bus='0x09' slot='0x00'
function='0x0' />
</interface>
```

Prior to run the playbook, the inventory file must be adapted to the needs of the particular deployment. An example is provided at *hosts.example* file, and basically it works as a feature toggle interface, where the machines to be deployed must be defined in the target group, and any feature specific must be enabled per host. Below the target group lies the general variables that control the features for all hosts defined above, so a sane set of defaults is provided.

To bootstrap the kubertelconetes cluster (intel-multus+flannel):

```
ansible-playbook -i inventory bootstrap.yaml
```

#### 2.2.5.6 Documentation

One-click deployment of a disaggregated 5G cloud-native E2E network software additional documentation and examples are available on QU4LITY Github<sup>21</sup>

<sup>21</sup> <https://github.com/qu4lity/tid-kube5g/tree/main/docs>

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

## 2.2.6 QU4LITY Cloud Bridge

### 2.2.6.1 Description and Usage

QU4LITY Cloud Bridge provides a seamless solution to exchange data using the QU4LITY Ontology Model (based on R-MPFQ developed in WP2), enabling a semantic enriched data exchange from on-premises data lakes to QU4LITY Cloud Data Storage using a time-based approach. Furthermore, as part of the realization process of the QU4LITY Cloud Bridge, ENG has been in charge of the transposition of the QU4LITY ontology (based on R-MPFQ model defined by EPFL) into a relational database (for better integration with analytics and visualization components). by delivering one more QU4LITY Digital Enabler, the so-called Q-Ontology Enabler, further described in the following Section 2.1.10.

QU4LITY Cloud Bridge is a node.js application that offers a REST API layer to ease the interfaces with other processing and visualization components pilot taking care of any data decoding/encoding needs (i.e. IEEE754 data encoding).

### 2.2.6.2 Relation with the Reference Architecture

QU4LITY Cloud Bridge provides solutions to implement the Cloud Platform in the Whirlpool pilot (even if the implemented solutions are reusable in other pilots and business cases as well) to ease data storing and accessing, can be mapped towards "Cloud" box (see Figure 1 above) of the Q-RA.

### 2.2.6.3 Dependencies

QU4LITY Cloud Infrastructure relies on the following tools/software:

- Node.js >= 14.17
- Express.js >= 4.17.1
- npm >= 6.14.13
- MariaDB >= 10.5.10
- Nginx >= 1.20.1
- Sequelize >= 6.3.5

### 2.2.6.4 Availability

QU4LITY Cloud Bridge is an Open-Source software and is offered both as source code and a ready-to-be-containerized solution at GitHub. In the following sub-sections, we provide details for this availability.

### 2.2.6.5 GitHub Distribution

QU4LITY Cloud Bridge source code is offered under the GNU Affero General Public License v3.0 and it's available on GitHub repository<sup>22</sup>

<sup>22</sup> <https://github.com/Engineering-Research-and-Development/qu4lity-cloud-bridge>

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

#### 2.2.6.6 Installation guidelines

QU4LITY Cloud Bridge can be deployed effortlessly by taking advantage of the facilities offered by Docker as well as installed locally with the help of node.js. The platform deployment scripts can be found on GitHub.

For a test deployment the following scripts and sample data are offered:

1. README.md: a text file containing instructions on how to install/deploy the platform as well as a list of all the REST API offered by the infrastructure;
2. docker-compose.yml: a YAML file containing the configurations of the various Docker containers (Docker images, environment variables, networking, data volumes configuration etc.) see Table 3 in Appendix I – Digital Enablers' Docker Compose Scripts below.
3. nginx\_conf: a folder containing a sample configuration for nginx
4. mariadb\_conf: folder containing all the .sql import scripts needed to prepopulate the databases with for demonstration purposes.

#### 2.2.6.7 Documentation

An API Documentation for QU4LITY Cloud Bridge is provided, as Postman Collection, on GitHub<sup>23</sup>

### 2.2.7 Q-Ontology Enabler

#### 2.2.7.1 Description and Usage

Q-Ontology Enabler is a set a python scripts which ease the migration to the QU4LITY relational database performing a ETL processes on existing on-premises infrastructures. The relational transposition of R-MPFQ model does not represent a new MES, but a way to organize in the most logical and structured way possible a varied reality although it may become that one day in the future.

#### 2.2.7.2 Relation with the Reference Architecture

Q-Ontology Enabler, being a complementary tool for the QU4LITY Cloud Bridge, can be also mapped towards "Cloud" box (see Figure 1 above) of the Q-RA.

#### 2.2.7.3 Dependencies

Q-Ontology Enabler relies on the following software:

- Python >= 3.9.0

#### 2.2.7.4 Availability

Q-Ontology Enabler is an Open Source collection of python scripts which are offered as migration solution towards R-MPFQ Ontology at GitHub. In the following sub-section, we provide details for this availability.

<sup>23</sup> <https://github.com/Engineering-Research-and-Development/qu4lity-cloud-bridge/tree/master/docs>

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

#### 2.2.7.4.1 GitHub Distribution

Q-Ontology Enabler source code is offered under the GNU Affero General Public License v3.0 and it's available on GitHub repository<sup>24</sup>

#### 2.2.7.5 Installation guidelines

Q-Ontology Enabler doesn't require any particular installation apart from Python interpreter.

#### 2.2.7.6 Documentation

Q-Ontology Enabler scripts need just to be executed under Python interpreter. Each Q-Ontology Enabler requires a data source in .csv format that must reside on the same root folder as the script being executed.

### 2.2.8 VTT OpenVA

#### 2.2.8.1 Description and Usage

VTT OpenVA platform consist of software components that are used as building blocks of visual analytics tools:

- A database that stores the application data in a standard domain independent form
- An extendable analysis and visualization library providing a selection of analysis and visualization methods. The library is customized based on application needs
- Embedded R statistical computing environment
- A web user interface where the user can select variables for analysis and explore the data with the help of visualizations. The visualizations can be in 2D or 3D and interconnected with real object visualizations. The user interface suggests the user the appropriate analysis methods letting them to concentrate on the substance instead of data analysis methods.

VTT OpenVA is independent of the underlying data collection solution. The data can come from several sources, also in real-time. The data to be analysed is loaded from the sources to the database through a uniform data interface.

#### 2.2.8.2 Relation with the Reference Architecture

VTT OpenVA Digital Enabler is placed at the "Simulation and Human-centric Visualization Srv." layer (see Figure 1 above) based on its visual analytics tools functionality.

#### 2.2.8.3 Dependencies

Some of the core dependencies of OpenVA are:

- Apache Tomcat
- PostgreSQL

<sup>24</sup> <https://github.com/qu4lity/q-ontology-enabler>

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

- Spring Framework

#### 2.2.8.4 Availability

VTT OpenVA is an Open-Source software and is offered both as source code at GitHub but as well as a containerized solution at Docker Hub.

- There is a clone repository under QU4LITY GitHub<sup>25</sup>
- There are two Docker images in Docker Hub:
  - pekka-siltanen/openvaplus, that contains VTT OpenVA jar-file and R with the libraries used in the visualizations and
  - pekka-siltanen/postgresopenva, that contains PostgreSQL database with example data.

#### 2.2.8.5 Installation guidelines

Detailed installation and demonstration guidelines are provided under the VTT OpenVA's wiki page on GitHub<sup>26</sup>

#### 2.2.8.6 Documentation

Detailed documentation is provided under the VTT OpenVA's wiki page on GitHub<sup>27</sup>

### 2.2.9 Secure Identity Directory (SID)

#### 2.2.9.1 Description and Usage

Secure Identity Directory (SID) is a decentralized infrastructure for identity management. It answers the problem of how a decentralized application may depend on common, trustworthy information about user identities without introducing any centralization bottleneck. Therefore, SID is centred on a *smart contract* that can be deployed either on the QU4LITY Value Chain Ledger (VCL) or on any Private Ledger (PL). User and machine identities are registered on the ledger in a format that is compatible with the DID standard (see next section) but also contains a rich (and extensible) set of information. This decentralized registry of identity descriptors allows any DApp that supports SID (including SID itself, as will be explained later) to assess the identity of their users in a secure way and, optionally, to enforce access restrictions according to their own policies.

The SID smart contract is installed on QU4LITY's VCL. However, The SID API is *not* exposed directly by the smart contract. Instead, a REST-over-HTTP interface is provided by a "SID API server" that acts as a mediator between the caller and the smart contract. The SID API server must *impersonate* the caller and forward every call to the smart contract, taking full responsibility for authentication. To do that, the SID API server needs full access to the private credentials of the user, which are kept locally in an encrypted vault: the *SID Wallet*. Consequently, the SID API server must

<sup>25</sup> <https://github.com/qu4lity/vttopenva>

<sup>26</sup> <https://github.com/pekka-siltanen/vttopenva/wiki/Docker-demo>

<sup>27</sup> <https://github.com/pekka-siltanen/vttopenva/wiki>

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

be installed and run as a *personal gateway* for a specific user and embed its SID Wallet. The usage of the server as a “regular” shared network service is *not* supported.

To allow users to manage the SID Wallet embedded in their personal API server, the SID DApp includes a simple command-line tool. By issuing commands, the user can insert new identities (which will then need to be registered on the SID ledger by some controller) in its wallet and browse through its contents.

#### **2.2.9.2 Relation with the Reference Architecture**

SID is a Distributed Application (DApp), the smart contract of which can be deployed on either the Value Chain Ledger (VCL) or any Private Ledger (PL).

#### **2.2.9.3 Dependencies**

No dependencies.

#### **2.2.9.4 Availability**

Any QU4LITY partner is allowed to experiment online with the SID DApp, but with the caveat that the VCL is just a “sandbox” environment: no guarantees are provided by the VCL administrator that the system will be up 24/7, that data will be persisted for long periods of time and that security will be properly managed.

A standalone deployment may also be done on any Private Ledger – refer to the installation guidelines below.

#### **2.2.9.5 Installation guidelines**

To use the VCL instance of the SID, both the command-line tool and the API server are needed. In order to receive a pre-built version of the both please contact the ENG team that is responsible for QU4LITY’s task T3.6.

Instead, if you need to install your own Private Ledger system, you must start from a stock distribution of Hyperledger Fabric v2.2<sup>28</sup>. You will then need to deploy the QCH smart contract (“chaincode” in Fabric’s jargon), the source code of which you can find in this public GitHub repository<sup>29</sup>.

#### **2.2.9.6 Documentation**

The secure identity directory documentation is available on GitHub<sup>30</sup>.

<sup>28</sup> <https://hyperledger-fabric.readthedocs.io/en/release-2.2/install.html>

<sup>29</sup> <https://github.com/qu4lity/qu4lity-dapps/tree/master/sid>

<sup>30</sup> <https://github.com/qu4lity/qu4lity-dapps/tree/master/sid>

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

## 2.2.10 Secure Messaging Board (SMB)

### 2.2.10.1 Description and Usage

Secure Messaging Board (SMB) enables the publishing of sensitive content (e.g., software, firmware, algorithms, parameters, data, documents) to subscriber channels, with the added value of strong guarantees on provenance, integrity and confidentiality. For instance, an OEM may safely send customized software updates to any specific smart machine installed at a client site, or to all machines of a certain type. While the actual content is stored as a *binary object* on any cloud facility, an immutable and timestamped record on the ledger provides a “digital seal” that guarantees its authenticity: the record includes a hash value calculated on the content, so that any alteration to the latter can be easily detected. Moreover, the SMB smart contract – in cooperation with SID (see previous chapter) – ensures that the publisher is always correctly identified and cannot be impersonated by a malicious actor.

The SMB smart contract is installed on QU4LITY’s Value Chain Ledger (VCL). This version of the SMB DApp is distributed together with a command-line client tool that offers a simple interactive interface. The client tool provides the basic read / write functions: a POST command for publishing new messages or replacing exiting ones with an updated version, a GET command for retrieving the latest version of given message and a VERSION command that queries for the current version number of a given message.

### 2.2.10.2 Relation with the Reference Architecture

SMB is a Distributed Application (DApp), the smart contract of which can be deployed on either the Value Chain Ledger (VCL) or any Private Ledger (PL).

### 2.2.10.3 Dependencies

SMB depends on the SID smart contract (see previous chapter). The current implementation only supports Google Drive as the cloud-based persistent storage for binary objects, so at least one Google Drive account is also required. Future versions will remove this limitation.

### 2.2.10.4 Availability

Any QU4LITY partner can experiment online with the SMB DApp, but with the caveat that the VCL is just a “sandbox” environment: no guarantees are provided by the VCL administrator that the system will be up 24/7, that data will be persisted for long periods of time and that security will be properly managed.

A standalone deployment may also be done on any Private Ledger – refer to the installation guidelines below.

### 2.2.10.5 Installation guidelines

To use the VCL instance of the SMB smart contract, only the command-line client tool is needed. To receive a pre-built version of the tool, including the `wallet.zip` file

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

that enables access to the VCL, please contact the ENG team that is responsible for QU4LITY's task T3.6.

Instead, if you need to install your own Private Ledger system, you must start from a stock distribution of Hyperledger Fabric v2.2<sup>31</sup>. You will then need to deploy the SMB smart contract ("chaincode" in Fabric's jargon), the source code of which you can find in this public GitHub repository<sup>32</sup>.

#### **2.2.10.6 Documentation**

The official documentation is available on GitHub<sup>33</sup>.

### **2.2.11 Quality Clearing House (QCH)**

#### **2.2.11.1 Description and Usage**

Quality Clearing House (QCH) enables a decentralized workflow that targets a typical Quality Management / ZDM process in a non-hierarchical supply chain scenario. It provides a common "system of record" for a manufacturing ecosystem where actors need to continuously assess the quality of raw material, parts and final products and match the results against contractual standards that may change frequently. Thanks to distributed ledger technology, QCH records are secure and trustworthy: they are timestamped, immutable and non-repudiable. Data storage and business logic are replicated on all the nodes of the system, which are operated equally by all participants, so that no single "owner" of the system exists who may introduce bias in the process.

The QCH smart contract is installed on QU4LITY's Value Chain Ledger (VCL). The QCH API is exposed by a Java library: Java applications are required to embed the library in their code in order to exploit QCH's functionality.

#### **2.2.11.2 Relation with the Reference Architecture**

QCH is a Distributed Application (DApp), the smart contract of which can be deployed on the Value Chain Ledger (VCL).

#### **2.2.11.3 Dependencies**

No dependencies.

#### **2.2.11.4 Availability**

Any QU4LITY partner is allowed to experiment online with the QCH DApp, but with the caveat that the VCL is just a "sandbox" environment: no guarantees are provided by the VCL administrator that the system will be up 24/7, that data will be persisted for long periods of time and that security will be properly managed.

<sup>31</sup> <https://hyperledger-fabric.readthedocs.io/en/release-2.2/install.html>

<sup>32</sup> <https://github.com/qu4lity/qu4lity-dapps/tree/master/smb>

<sup>33</sup> <https://github.com/qu4lity/qu4lity-dapps/tree/master/smb>



QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

A standalone deployment may also be done on any Private Ledger – refer to the installation guidelines below.

### ***Installation guidelines***

To use the VCL instance of the QCH, only the client library is needed (see previous section). In order to receive a pre-built version of the library, including the `wallet.zip` file that enables access to the VCL, please contact the ENG team that is responsible for QU4LITY's task T3.6.

Instead, if you need to install your own Private Ledger system, you must start from a stock distribution of Hyperledger Fabric v2.2<sup>34</sup>. You will then need to deploy the QCH smart contract ("chaincode" in Fabric's jargon), the source code of which you can find in this public GitHub repository<sup>35</sup>.

#### **2.2.11.5 Documentation**

The official documentation is available on GitHub<sup>36</sup>.

## **2.2.12 Security Privacy and Trust Framework**

### **2.2.12.1 Description and Usage**

The Security, Privacy and Trust Framework is the architectural platform of QU4LITY for providing different cybersecurity functionalities such as security of data, privacy, access control, monitoring, etc. This platform can be extended with additional tools in order to provide even more functionalities in order to adapt the system to the needs of the industrial platforms. Therefore, the platform allows for different tools to be used and able to communicate using the communication channels and data repository. This way the data can be correlated and used to obtain more expert information of cyberattacks.

Additionally, being one of the tools about cybersecurity modelling, in order to allow for cybersecurity-as-a-service, this could be extended and integrated with other tools so it could be used for creating secure industrial systems from the design phase, allowing for a more robust monitoring and protection of the data.

### **2.2.12.2 Relation with the Reference Architecture**

The different tools of the SPT are related to different elements of the reference architecture such as data storage and communication and access control systems. This way, the solutions provided by the SPT could be used to protect the communication from external systems to QU4LITY and for monitoring the authentication and authorization of users to different tools or information of QU4LITY.

<sup>34</sup> <https://hyperledger-fabric.readthedocs.io/en/release-2.2/install.html>

<sup>35</sup> <https://github.com/qu4lity/qu4lity-dapps/tree/master/chq>

<sup>36</sup> <https://github.com/qu4lity/qu4lity-dapps/tree/master/chq>

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

### 2.2.12.3 Dependencies

The SPT doesn't have so far any dependency with other parts of QU4LITY as pre-requisite. Tools of the SPT could interact with other components but they work in a different level as they are self-contained tools that would use a common communication channel and data structure for the security and privacy.

### 2.2.12.4 Installation guidelines

The different tools of the SPT that require specific integration and configuration (and are available for access) are described in the following subsections (see 2.2.13 and 2.2.14 below).

## 2.2.13 XL-SIEM

### 2.2.13.1 Description and Usage

The function of the XL-SIEM is to create alarms by correlating every event detected by its agent, as well as the monitoring sensors.


The SIEM have five clearly differentiated parts, one on the client side, and four on the server side:

- A XL-SIEM Agent, which works a SYSLOG server, collecting all relevant information and sending to the server.
- One docker container with a PostgreSQL database. It used to store all the info (events, alarms, and correlated rules) generated in the project.
- One container with the GUI interface. Using its dashboard, the user can visualize and filter the events and alarms and create new correlation rules.
- A message broker based in RabbitMQ used to interact between different components of the SIEM.
- The topology is composed of 5 dockers and make the internal correlation and generate the alarms to be stored in the database and to be showed in the dashboard.

### 2.2.13.2 Relation with the Reference Architecture

The XL-SIEM will be used as a main tool in the SPT, focusing in the area of monitoring and detection of malicious activities in the systems. This way, the information the XL-SIEM produces can be accessed (and analysed) by a cybersecurity expert in the dashboard of the tool. As this is the only tool in this area in the project we did not integrate with others but due to its architecture it can easily be integrated either for data access, correlation, or output of the analysis.

Additionally, the XL-SIEM is deployed in a QU4LITY server, which can then be accessed by the user remotely (in the cloud). This is an additional option to have it deployed in the same system to be monitored, so both provide for a more accessible way of using the tool according to the resources of the stakeholder. This goes in line with the QU4LITY architecture, as the XL-SIEM could be deployed in either way.

	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

Regarding the integration with the system to be protected, the users have to download an agent that will be installed and configured. This agent will then send information to the XL-SIEM, being local or remote, so the integration in the QU4LITY platform is straightforward and simple.

### 2.2.13.3 Availability

The 4 parts of the SIEM server are deployed and working in our host. ATOS will give support and keep it active and working.

Information of the deployment of the agent is provided in terms of script and configuration files in order to help their integration and configuration.

The agent of the XL-SIEM is available in the repository of QU4LITY.

### 2.2.13.4 Installation guidelines

The XL-SIEM installation have mainly two parts; a python script which edit the needed configuration files and a bash script with build all the containers.

To automate the installation, there is also a third script that launch the two scripts mentioned above. A json file with the relevant information needs to be edited before launching the first script. All the scripts are provided in Table 4, Table 5 and Table 6 of Appendix II – Native Installation Scripts below.


### 2.2.13.5 Documentation

The XL-SIEM have a graphical interface. It provides an easy and quick way to visualize info and customize the XL-SIEM:

The main view of the GUI show at resume of all event and alarms generated at first sight, as shown in the next figure:



Figure 9 Welcome view of event and alarms generated

	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

## Add a new Agent



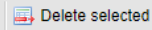

Through the GUI, a new agent can be added. The user just have to submit the IP and the name, and eventually some optional info.

► Dashboards	► SIEM Analysis	Configuration	► Reports
--------------	-----------------	---------------	-----------

XL-SIEM Agents		Displaying 1 to 2 of 2 XL-SIEM Agents		
----------------	--	---------------------------------------	--	--

			
---	---	---	---

IP	Name	Priority	Version	Description
200.10.14.5	xlsiem-server	5	4.1.0	
200.10.14.15	ATOS	5	4.1.0	ATOS

Figure 10 XL-SIEM New agent addition

Welcome admin ► Logout  
atos XL-SIEM

Name *	<input type="text"/>
IP *	<input type="text"/>
Priority	5 ▼
Timezone	GMT+2:00 ▼
Organization	<input type="text"/>
Description	<input type="text"/>

Update Clear Cancel

Values marked with (\*) are mandatory

Figure 11 Entering user data

## Create a new type of event



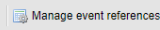
New type of events can be created

► Dashboards	► SIEM Analysis	Configuration	► Reports
--------------	-----------------	---------------	-----------

Data Sources		Displaying 1 to 25 of 387 Data Sources		
--------------	--	--	--	--

		
---	---	---

Data Source ID	Name	Type	Product Type	Description
100000	Antijamming	Detector (1)		Workdsensing Antijamming
90012	NF-Alert	Detector (1)	Management Platform	Netflow Alerts
90011	NIOTX	Detector (1)		Netflow OTX Matcher
90008	RSA-DPM	Detector (1)	Data Protection	RSA DPM
90007	VCenter	Detector (1)	Management Platform	VmWare Virtual Center
90005	SAP	Detector (1)	Other Devices	SAP Software
90003	nessus-info	Detector (1)	Vulnerability Scanner	nessus-ossec
70000	XL-SIEM	Detector (1)	Alarm	XL-SIEM Alerts
51512	arpwatch_new	Detector (1)		Ethernet/FDDI station monitor daemon
30000	TestSensor	Detector (1)	Alarm	A simple sensor for testing purposes
20505	post_correlation_directive	Detector (1)	Alarm	Alienvault post correlation engine for SQL queries
19004	websense	Detector (1)	Data Protection	WebSense: Websense Data Security Suite

Figure 12 New type of events creation



QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

anonymization tool<sup>37</sup>, which comes both as a graphical tool and a software library with an API.

#### **2.2.14.2 Relation with the Reference Architecture**

This enabler is part of the “distributed trustworthiness” layer of the Reference Architecture. As part of the security function of the Reference Architecture it serves as a standalone enabler to assure data privacy among ZDM components and digital platforms by providing data anonymization.

#### **2.2.14.3 Dependencies**

The graphical tool of ARX is a standalone desktop application that can be run on Windows, Linux and MacOS machines. The software library of ARX comes in two versions:

- a) libarx, which contains all dependencies required for using all of ARX’s features and,
- b) libarx-min, which does not include external libraries required. The minimal dependencies for libarx-min are the Colt<sup>38</sup> open source libraries for high performance scientific and technical computing, high performance primitive collections (HHPC)<sup>39</sup> for Java, the Apache commons mathematics library<sup>40</sup> and Java high-performance library for lattices (JHPL)<sup>41</sup>. The full list of dependencies can be found here<sup>42</sup>.

#### **2.2.14.4 Availability**

The ARX data anonymization tool is available both as a cross-platform graphical tool supporting data import, wizards for creating transformation rules, and visualizations of data utility and re-identification risks, and as a software library with an API that delivers data anonymization capabilities to any Java program.

#### **2.2.14.5 Installation guidelines**

The ARX graphical anonymization tool is a platform-independent Java/SWT application. It can be downloaded from its official website<sup>43</sup> as self-contained binary installers as well as executable jar files.

#### **2.2.14.6 Documentation**

The documentation is common with XL-SIEM digital enabler (see section 2.2.13.5 above).

<sup>37</sup> <https://arx.deidentifier.org>

<sup>38</sup> <https://dst.lbl.gov/ACSSoftware/colt>

<sup>39</sup> <http://labs.carrotsearch.com/hppc.html>

<sup>40</sup> <https://commons.apache.org/proper/commons-math>

<sup>41</sup> <https://github.com/prasser/jhpl>

<sup>42</sup> <https://arx.deidentifier.org/development/dependencies>

<sup>43</sup> <https://arx.deidentifier.org/downloads>

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

## 2.3 Integrated Digital Enablers

In this section we present a list of interoperable digital enabler groups that are packaged together to offer a combined solution.

### 2.3.1 Data Transformation Platform + DataCROP

#### 2.3.1.1 Description and Usage

This integrated digital enabler uses the Data Transformation Platform (presented in section 2.2.2) and a data repository (MongoDB) with the objective of providing a system for collecting and storing information coming from the industrial assets of a plant in the form of messages. The platform also offers an API to consume that information from AI applications, monitoring tools or data analysis algorithms. The platform comes from a previous project (Mantis). Improvements have been included to the platform that now includes the following components:

- Data access and ingestion through an Edge Broker. This is part of the Data Transformation Platform presented in section 2.2.2. The Edge Broker is composed of Publish-Subscribe servers that collect messages sent by the CPS connected to the assets in the plant. CPS messages are generated according to the communication and information model proposed by AAS. The solution has been implemented using; RabbitMQ and Advanced Message Queuing Protocol (AMQP).
- Translator/Converters to collect the messages coming from the edge broker and sent them to the repository. The converters have been implemented using Node-RED. This converters have the capability to additionally offer REST endpoints. This component is also part of the Data Transformation Platform presented in section 2.2.2.
- Data Storage systems to store those messages using MongoDB.
- A messaging system based on AAS uses the edge broker (RabbitMQ) to accommodate different types of payload (values, files and events).
- A REST API is offered to consume the information store in the MongoDB repository. The API has been developed using Node-RED (part of the Data Transformation Platform digital enabler).

This integrated digital enabler is used in the Mondragon pilot which enables the construction of a messaging architecture to collect information from assets. In addition, the API allows to consume data stored in the repository. The API is consumed by a visualization solution built with Grafana<sup>44</sup> that monitors process events and parameters in the pilot. More details on this integrated digital enabler will be provided in the deliverable associated to the pilot (to be developed).

#### 2.3.1.2 Dependencies

Data Transformation Platform has the following environment and software dependencies:

<sup>44</sup> <https://grafana.com/>

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

- WSO2 Integrator >= 6.5.0
- RabbitMQ >= 3.8.17
- Node-RED >= 1.3.5
- Node.js >= 12.22.1
- MongoDB >= 4.4.6

### 2.3.1.3 Availability

This integrated digital enabler is an Open-Source software under MIT license and is offered as source code and a containerized solution at GitHub. In the following sub-section, we provide details for its availability.

#### 2.3.1.3.1 GitHub Distribution

The components for this enabler are images available in Docker Hub and the containerized solution is available from GitHub. Code in the form of Node-RED flows is also part of the solution. This code permits message collection and API provision. Links to the relevant repositories are provided below:

- Data Transformation Platform + DataCROP
  - <https://github.com/qu4lity/dtp-crop>

### 2.3.1.4 Installation guidelines

The core infrastructure of the Data Transformation Platform + DataCROP is deployed effortlessly by taking advantage of the facilities offered by Docker. The platform deployment scripts can be found on GitHub<sup>45</sup>.

For a test deployment the following scripts and sample data are offered:


- **README.md**: a text file containing instructions on how to deploy the platform
- **Docker-compose.yml**: a YAML file containing the configurations of the various Docker containers (Docker images, environment variables, networking, data volumes configuration etc.).
- **Node-red**: a folder holding Node-RED's Dockerfile and the flows necessary to collect information from the edge broker and to offer data through the REST API.
- **Mongo**: a folder to assure persistence of the MongoDB collections created with the platform.

By following the instructions given in the README file, one may both deploy and undeploy the various components of the integrated Digital Enabler. To do so, Docker Compose is being used. Compose is a tool for defining and running multi-container Docker applications. A YAML file is being employed to configure all application's services. Then, with a single command, one may create and start all the services from the aforementioned configuration<sup>46</sup>.

<sup>45</sup> <https://github.com/qu4lity/dtp-crop>

<sup>46</sup> <https://docs.docker.com/compose/>



	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

## Starting up

In order to put the project working, you must execute the following command (docker and docker-compose must be already installed) in the folder where docker-compose.yml is:

- `docker-compose up -d`

This will create and start 4 containers:

1. Node-RED => at port 1880
2. RabbitMQ => at port 5672 (management at port 15672)
3. MongoDB => at port 27017
4. MongoDB Express => at port 8081

The first time we run the containers, the username and password for RabbitMQ should be added to node-red nodes connecting to RabbitMQ. In order to do that, we have to access Node-RED (e.g. `http://localhost:1880`) and edit the configuration nodes (amqp-server node and mongodb node):

- amqp-server node: adding the username and password in the security tag.
- mongodb node: adding the username and password directly in the configuration node.

Deploy the changes and the connections to RabbitMQ and Mongo should be working.

The containers are stopped using the following command:

- `docker-compose down`

## Testing

We will send a AMQP message through RabbitMQ, receive it and add the content to a MongoDB collection. Then we will the endpoint to retrieve the entire collection.

1. Open Node-REd in a browser (e.g. `http://localhost:1880`).
2. Go to RabbitMQ tab and press the blue button (inject node).
3. Open the HTTP endpoint in a browser: (e.g. `http://localhost:1880/get/UserCollection`) for UserCollection.
4. A user with admin username should be visible.

## 2.3.2 DataCROP + LDM + Fault Identification

### 2.3.2.1 Description and Usage

This solution facilitates the data collection, data management and usage of Mondragon's Fault Identification algorithm. The mission of the algorithm is, given a collection of sensor measurements as input to detecting anomalies in time-series in JSON format. The anomaly detection is then packaged in the "value" property of the

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

outgoing message of type "Observation" (from the LDM), which is handled by the DataCROP platform, in order to be returned to a Kafka bus. The results (Observation) are annotated based on the LDM and are available to be further aggregated or analysed by the DataCROP platform or be used from any third-party application.

### **2.3.2.2 Integration**

DataCROP platform offers specialized wrappers with standard interfaces in order to facilitate the integration of third-party standalone algorithms. These wrappers handle in a standardized way the required interactions with the Processor Engine and the subscription/publishing to the Message Bus. The subscriptions to the appropriate topics in the DataCROP message bus of the message bus would be used to get information as soon as it becomes available, and the publishing is for "writing" the results back to the appropriate topics as soon as they are produced. For the fault identification algorithm, a Python wrapper is offered.

The wrapper, after receiving an Observation from Kafka, isolates a double carried in its "value" element. Then it executes the Python script with the double as input. After that, it receives the calculation result and repackages it in an Observation. Finally, it publishes the new Observation to Kafka.

If the default configuration is maintained, the two involved Kafka topics are "incoming\_messages" and "output\_messages".

### **2.3.2.3 Availability**

As mentioned above DataCROP (along with the LDM and the Java Wrapper) is an Open-Source software and is offered both as source code at GitHub but as well as a containerized solution at Docker Hub (see section 2.2.1). The Fault Identification algorithm is not open source, so the code/artifact distribution is not available. For the QARMA analytics please contact Mondragon University<sup>47</sup>.

### **2.3.2.4 Installation guidelines**

It follows the same principles as the ones described in section 2.2.1.5 above.

## **2.3.3 DataCROP + LDM + QARMA Analytics**

### **2.3.3.1 Description and Usage**

This solution facilitates the data collection, data management and usage of QARMA machine learning algorithm. The mission of the algorithm is, given a collection of sensor measurements as input to produce a prediction in JSON format. The prediction is then packaged in the "value" property of the outgoing message of type "Observation" (from the LDM), which is handled by the DataCROP platform, in order to be returned to a Kafka bus. The results (Observation) are annotated based on the

<sup>47</sup> [ezugasti@mondragon.edu](mailto:ezugasti@mondragon.edu)

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

LDM and are available to be further aggregated or analyzed by the DataCROP platform or be used from any third-party application.

### 2.3.3.2 Integration

For this integration the QARMA analytics wrapper variation has been implemented. This wrapper supports algorithms which are written in Java (QARMA Algorithm) and subsequently introducing it to the wrapper not as an external executable script but as a Java library included in the project as an Apache Maven dependency.

Since the Wrapper is also written in Java (as we can see in Figure 15 below) this library is automatically be packaged in the Java wrapper JAR file, without the need to place the external algorithm script in a particular directory in the filesystem. The QARMA predictor exported methods available directly from use within the code:

```
// Wrap the result into an observation.
final Observation newValue = new Observation();
newValue.setId(UUID.randomUUID().toString());
newValue.setEdgeGatewayReferenceID(System.getProperty(ProcessorConfig.EDGE_GATEWAY_ID_CONFIG));
newValue.setDataSourceManifestReferenceID(sink);
newValue.setDataKindReferenceID(value.getDataKindReferenceID());
newValue.setCollectionTimestamp(value.getCollectionTimestamp());
newValue.setAcquisitionTimestamp(value.getAcquisitionTimestamp());
newValue.setLocation(value.getLocation());

Prediction prediction = new Prediction(result._predictedValue, result._lower95PercValue, result._upper95PercValue);
newValue.setValue(prediction);
```

Figure 15: The QARMA algorithm used in the wrapper as a library

The implementation of the QARMA wrapper also bears the distinctive characteristic of collecting the input messages into bundles before pushing them into the actual QARMA code as input. As each message corresponds to a collection of timestamped sensor measurements and the algorithm requires a time series for increased accuracy in predicting RUL, the wrapper assumes the additional role of the entity remembering and packaging the "latest x timestamped measurements" into a single structure before "feeding" them to QARMA. In addition, this "x" variable is configurable as an environment variable that can be introduced using the ML dashboard (not unlike the Kafka virtual address and the topic names).

### 2.3.3.3 Availability

As mentioned above DataCROP (along with the LDM and the Java Wrapper) is an Open-Source software and is offered both as source code at GitHub but as well as a containerized solution at Docker Hub (see section 2.2.1). QARMA analytics is not open source so the code/artifact distribution is not available. For the QARMA analytics please contact Intrasoft International<sup>48</sup>.

### 2.3.3.4 Installation guidelines

It follows the same principles as the ones described in section 2.2.1.5 above.

<sup>48</sup> [ioannis.soldatos@intrasoft-intl.com](mailto:ioannis.soldatos@intrasoft-intl.com)

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

## 2.3.4 DataCROP + LDM + Secure Messaging Board (SMB)

### 2.3.4.1 Description and Usage

This solution offers all the functionalities of the DataCROP component (collection, routing, pre-processing and annotation of collected data) but is enhanced with decentralized data reliability offered by the Secure Messaging Board (SMB) component. DataCROP is enhanced following the principles below:

- Secure Messaging Board by supporting DataCROP Processor Engine provides the capability for distributing processor manifest objects (e.g., analytics configuration data) across multiple gateways. Note that a processor manifest defines how data streams are to be processed by an individual Processor Engine, using a combination of predefined data processing elements and workflow instructions. Using the SMB and Distributed Ledger as the distribution channel, DataCROP ensure a truly decentralized but also reliable system, as processor manifests are "signed, sealed and timestamped" so that no forgery or tampering is possible.
- DataCROP can also utilize a result publishing service offered by SMB that makes it possible for processor instances (e.g. analytics), wherever deployed, to share their results on the Distributed Ledger infrastructure, thus contributing to a common data set representing the combined results across the entire distributed system. The virtues of such as workflow guarantees on provenance, integrity and confidentiality of the results.

### 2.3.4.2 Integration

For the integration of the two enablers the SMB replaces the DataCROP Model and Registry repository by supporting specialised interfaces for the Data Source Manifests (DSM), Processor Manifests (PM), Data Source Definitions (DSD) and Processor Definitions (PD). Moreover, for the result persistence SMB offers also support for the DataCROP Observations. The above-mentioned data models are described in more details in deliverable D3.13 section 5.3.

### 2.3.4.3 Availability

As mentioned above DataCROP (along with the LDM) and SMB are Open-Source software and are both offered at GitHub (see sections 2.2.1 and 2.2.10 above). DataCROP is also offered as a containerized solution at Docker Hub (see section 2.2.1).

### 2.3.4.4 Installation guidelines

It follows the same principles as the ones described in sections 2.2.1.5 and 2.2.10.5 above respectively.

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

## 2.3.5 QU4LITY Ontology + Visual Component Simulation

### 2.2.6.1 Description and Usage

The QU4LITY Ontology, as introduced in Section (2.1.1) is applied to support the development of application ontologies for different pilots. One of the applications is the Airbus pilot, where an application ontology is developed and integrated with the Visual Component simulation system to create a trade space framework for aircraft production system design. It focuses on the research and development phase of the assembly line for a new model of aircraft. During this early phase, industrial architects need to evaluate different industrial scenarios and to perform trade-off to optimize the future industrial architecture using different performance parameters. More specifically, the application scenario focuses on the fuselage orbital junction process to be designed for one assembly station of the Final Assembly Line (FAL) for the new aircraft model. There are two options to execute the orbital junction process, i.e. a manual process and an automated process using a flex-track robotic mechanism.

The trade-off is expected to be performed between the manual process and the automated flex track process. The main differences between them are the external and internal drilling operations. For the manual process, both drilling operations are performed by operators; for the automated option, the external drilling operations are performed by the Flex Track robot while the internal drilling operations are performed by operators.

According to the application scenario, the overall functional architecture for this pilot is defined as shown in Figure 16. Figure 16, it contains several function blocks including Requirement Management block, Architecture Definition block, Visualization block, System Integration block and Verification block. The ontology is the core of the System Integration block which integrates all relevant data and information from other blocks.

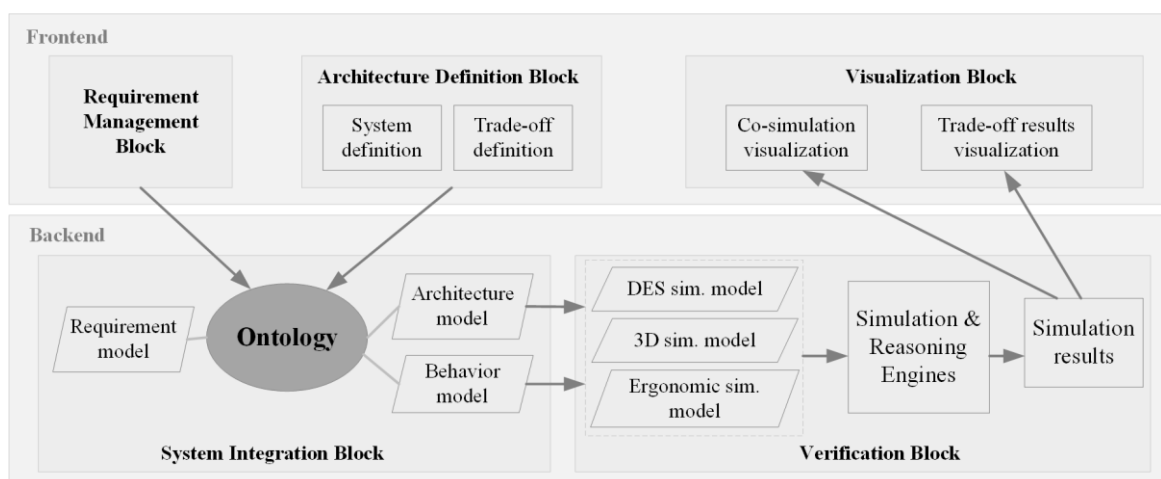



Figure 16 Overview of the functional architecture of trade space framework for aircraft production system design

As shown in Figure 17, the ontology serves as the information and knowledge integration hub providing input for the discrete event simulation (DES) and 3D

	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

simulation in the verification block. The ontology is exported as OWL file and a customized OWL parser is developed to parse the ontology and extract necessary information for the simulations. The simulation function is supported by Visual Components 3D factory simulation suite that consists of a set of innovative tools which set the standard for modern simulation. The simulation suite gives machine builders, system integrators, and manufacturers around the world a simple, quick, and highly cost-effective way to build and simulate their total process solutions. The simulation results are further processed and visualized by the visualization function block to support decision making.

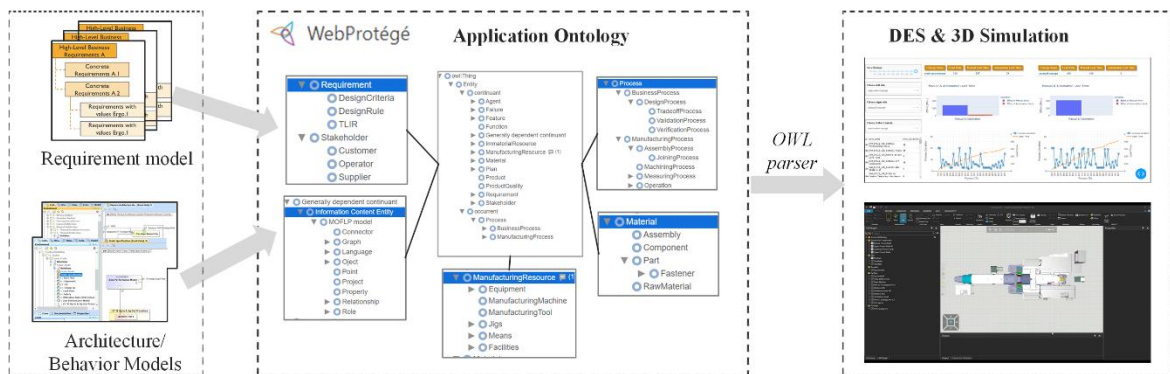


Figure 17 The roles of application ontology and 3D simulation in the trade space framework

### 2.3.5.1 Availability

The entire trade space is still under development and is not yet ready for sharing. The current application ontology is open available on WebProtégé<sup>49</sup>, which is under frequent update. The Visual Components 3D factory simulation suite is provided by partner Visual Components<sup>50</sup>. More details about the software are available on the website provided.

### 2.3.5.2 Installation guidelines

The current version of this ontology is available, read-only, on Webprotégé. For editing and redesign, it can be exported in different format such as RDF, XML, Turtle, OWL etc. The integration and parser for the simulation software is still under development, which will be shared by the end of the Airbus pilot.

## 2.3.6 Anomaly Detection for Assembly Process / Shift-In Movement

### 2.3.6.1 Description and Usage

Fraunhofer-IPA develops a ML based system for anomaly detection. The system can be tailored to different assembly processes in order to identify process anomalies as early as possible and to enable corrective actions. The system requires data from the production line: One key requirement is that process data is available from the

<sup>49</sup> <https://webprotege.stanford.edu/#projects/8b324962-cac1-43ff-8e31-428dd4ae654f/sharing>

<sup>50</sup> <https://www.visualcomponents.com/>

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

assembly process itself. This may contain time series from encoders (integrated in axis from linear actuators and robots), fore-torque-sensors, force-way-sensors and/or other relevant sensors. The system also utilizes data from EOL test station (or from other subsequent test assessments).

Anomaly detection takes place on an edge device that is located at the production site. The software on the edge device performs the ML based inference (i.e., the evaluation of the current/previous cycle), in combination with other auxiliary tasks like data pre-processing, data fusion or database storage. It also performs training of the ML models when the ML model complexity is low.

In case of ML models with high complexity, the training of the ML models takes place in the cloud, i.e., the training is performed by cloud services. The cloud services are available at clouds operated by Fraunhofer IPA (e.g., Virtual Fort Knox).

#### 2.3.6.2 Availability

The software running on the edge device will be available as a docker container. This facilitates the integration into different environments. By now, it is however not decided yet how the adaption to a specific productions process will be implemented.


## 2.4 QU4LITY Open-Source Digital Enablers

Several of the Digital enablers described above are offered as Open-Source Software, thru QU4LITY GitHub<sup>51</sup> organization (see Figure 18 below), following the code management methodology described in this deliverable. As mentioned in section 2.1.3, most of the digital enablers had pre-existing codebase which was mirrored under the QU4LITY GitHub organization. The Digital enablers that are provided thru the QU4LITY GitLab are also listed below:

1. One-click deployment of a disaggregated 5G cloud-native E2E network from TID
  - Available at: <https://github.com/qu4lity/tid-kube5g>
2. QU4LITY Cloud Infrastructure from ENG
  - Available at: <https://github.com/qu4lity/qu4lity-cloud-bridge>
3. Q-Ontology Enabler from ENG
  - Available at: <https://github.com/qu4lity/q-ontology-enabler>
4. OpenVA from VTT
  - Available at: <https://github.com/qu4lity/vttopenva>
5. DataCROP DDA Platform from INTRA
  - Available at: <https://github.com/qu4lity/data-crop>
6. QU4LITY DApps from ENG which includes the
  - a. Secure Identity Directory (SID) available at: <https://github.com/qu4lity/qu4lity-dapps/tree/master/sid>
  - b. Secure Messaging Board (SMB) available at: <https://github.com/qu4lity/qu4lity-dapps/tree/master/smb>

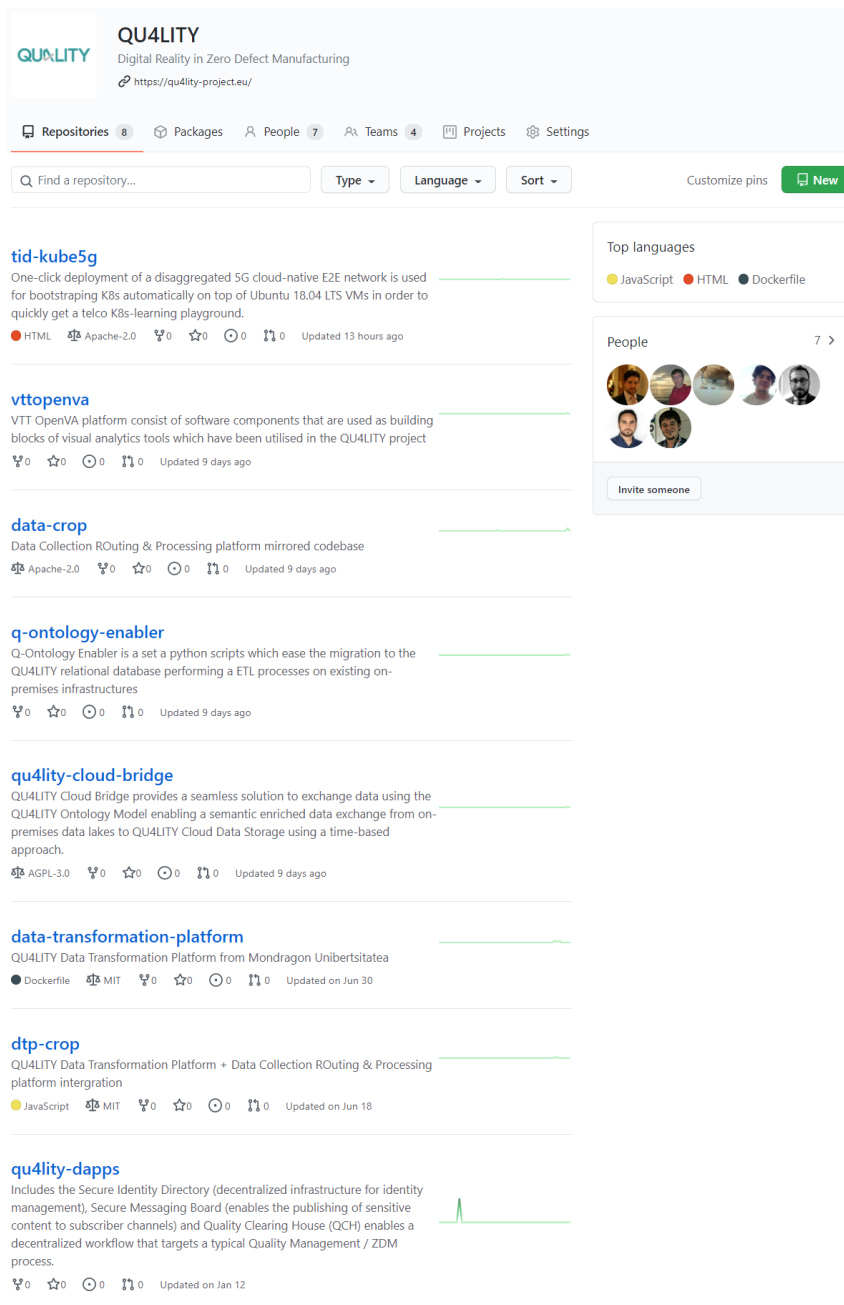
<sup>51</sup> <https://github.com/qu4lity>



	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

- c. Quality Clearing House (QCH) available at:  
<https://github.com/qu4lity/qu4lity-dapps/tree/master/chq>
7. Data Transformation Platform from MGEP.
- Available at: <https://github.com/qu4lity/data-transformation-platform>
8. Data Transformation Platform + DataCROP from MGEP + INTRA.
- Available at: <https://github.com/qu4lity/dtp-crop>

Figure 18 below illustrates the GitHub QU4LITY organization welcome page with the available digital enabler repositories.



**QU4LITY**  
Digital Reality in Zero Defect Manufacturing  
<https://qu4lity-project.eu/>

Repositories 8 Packages People 7 Teams 4 Projects Settings

Find a repository... Type Language Sort Customize pins New

**tid-kube5g**  
One-click deployment of a disaggregated 5G cloud-native E2E network is used for bootstrapping K8s automatically on top of Ubuntu 18.04 LTS VMs in order to quickly get a telco K8s-learning playground.  
HTML Apache-2.0 Updated 13 hours ago

**vttopenva**  
VTT OpenVA platform consist of software components that are used as building blocks of visual analytics tools which have been utilised in the QU4LITY project  
Updated 9 days ago

**data-crop**  
Data Collection ROUTing & Processing platform mirrored codebase  
Apache-2.0 Updated 9 days ago

**q-ontology-enabler**  
Q-Ontology Enabler is a set a python scripts which ease the migration to the QU4LITY relational database performing a ETL processes on existing on-premises infrastructures  
Updated 9 days ago

**qu4lity-cloud-bridge**  
QU4LITY Cloud Bridge provides a seamless solution to exchange data using the QU4LITY Ontology Model enabling a semantic enriched data exchange from on-premises data lakes to QU4LITY Cloud Data Storage using a time-based approach.  
AGPL-3.0 Updated 9 days ago

**data-transformation-platform**  
QU4LITY Data Transformation Platform from Mondragon Unibertsitatea  
Dockerfile MIT Updated on Jun 30

**dtp-crop**  
QU4LITY Data Transformation Platform + Data Collection ROUTing & Processing platform intergration  
JavaScript MIT Updated on Jun 18

**qu4lity-dapps**  
Includes the Secure Identity Directory (decentralized infrastructure for identity management), Secure Messaging Board (enables the publishing of sensitive content to subscriber channels) and Quality Clearing House (QCH) enables a decentralized workflow that targets a typical Quality Management / ZDM process.  
Updated on Jan 12

Top languages  
JavaScript HTML Dockerfile

People 7 >  
Invite someone

Figure 18 QU4LITY GitHub organization.



QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

### 3 QU4LITY Platform Interoperability

According to the definition of IEEE, “interoperability” means the ability for two (or more) systems or components to exchange information and to use the information that has been exchanged. Interoperability plays a key role in ZDM-related scenarios. In broader context, interoperability describes the ability of a system or software application to exchange or make use of data and is an essential requirement for all hard and software elements that participate in the exchange of information in the given framework. Due to a cross-linking of multiple systems in various domains in I4.0, interoperability now extends and comprises a much broader framework. Therefore, regarding common RA components, interoperability combines all three layers, integration layer, communication layer, and information layer, as introduced in the RAMI4.0 architecture.

Modern industrial products and systems are highly complex which usually involve multiple standards, such as ETSI, IETF, IEEE, ITU-T and ISO etc., as well as requirements set by Industry forums. QU4LITY interoperability specifications are applied to direct the work to be accomplished by a QU4LITY Pilot in terms of interoperability. The specifications define essential technical and operational standards that must be met by technical systems across QU4LITY interoperability scenarios. This should be done to meet the key requirements and ensure interoperability in respect of components and interfaces.

#### 3.1 QU4LITY Common Standards used

In the deliverable D2.8, QU4LITY Task 2.4 has thoroughly reviewed the interoperability requirements of QU4LITY pilots. The requirements of every pilot are investigated mapping to different degrees of interoperability: Unstructured Data Exchange, Structured Data Exchange, Seamless Sharing of Data, and Seamless Sharing of Information. According to the QU4LITY RA, the general interoperability requirements are analysed based on the RA components defined in D2.12, including Collaboration, Business and Operation Services; Engineering and Planning Services; Data-driven Modelling and Learning Services; Digital Twin and Planning Services; Simulation and Human-centric Visualization Services; IoT Automation Services; Control Services; Assets & Smart Products; HPC; Cloud; Value Chain Ledger; Data Lake / Big Data Analytics Infrastructure; IoT Hub; Private Ledger; Edge/Fog; 5G Multi-Access Edge Computing.

Based on the requirements of each pilot, interoperability standards, protocols and related frameworks are collected and classified. These protocols and standards are grouped into the following layers to provide some level of organization: Communication protocol; Semantic; Network functionality; Physical functionality; and Multi-layer framework.

- **Communication protocol:** OPC-UA, Modbus, GigE Vision, USB3.0, Profibus, CameraLink, MQTT, QIF, HTTP/REST, SSH/FTP, PPMP, MTconnect, CoAP, AMQP, DDS, XMPP.

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

- **Semantic:** AutomationML, SSN Ontology, IoTDB, SensorML, LsDL, RAML, SENML, MPFQ.
- **Network functionality:** TCP; UDP; Routing – RPL (RFC6550), CARP; IPv6/IPv4; addressing, Multicast, QoS, Security; IEEE 802.1X EAP T/TLS.
- **Physical functionality:** 5G; IEEE 802.15.4; ISA 100.11a; Bluetooth; WirelessHART; LPWAN; 3G/LTE; ANT+; IEEE 802.2 Ethernet; RFID/NFC; IEEE 802.22 Wi-Fi; DigiMesh.

**Multi-layer framework:** AllJoyn; IoTivity; ZigBee; ROS; Z-Wave; KAFKA; Thread.

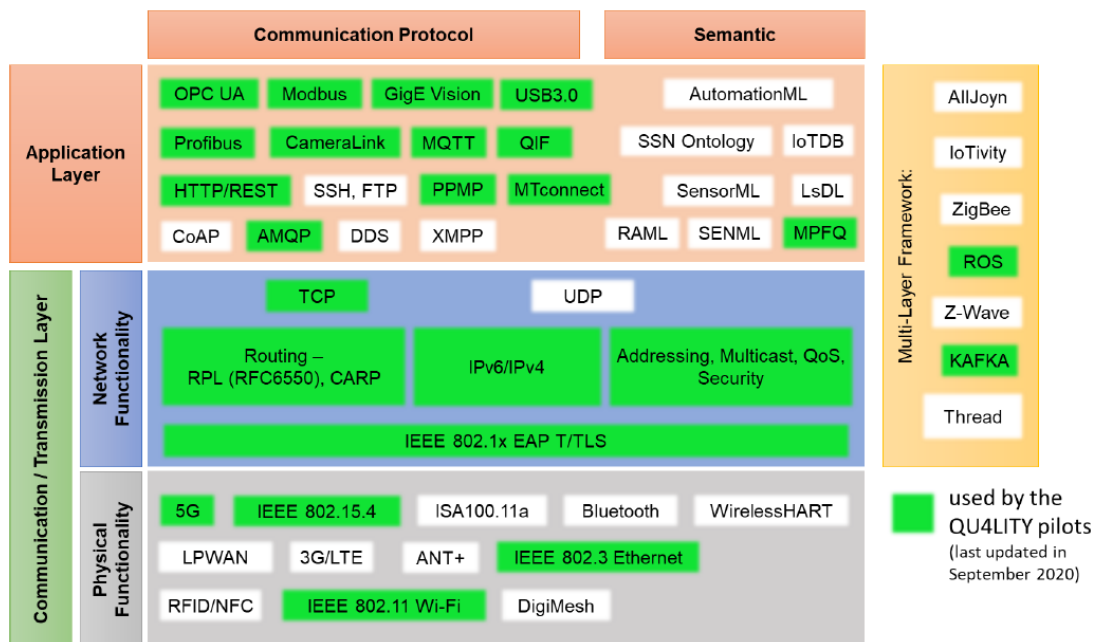



Figure 19 Overview of interoperability standards, protocols and frameworks for ZDM

According to this classification, the standards used by the QU4LITY pilots (green) are shown in Figure 19, together with other standards and protocols that extend the groups. The detailed standards are introduced in D2.8 Chapter 4.1.

A survey was conducted by T2.4 to investigate the application of relevant protocols, standards concerning interoperability used by the QU4LITY pilots. Part of the results are shown in Figure 20 below. In can be seen, that MQTT and OPC-UA are most frequently used, whereas some pilots use proprietary communication protocols and APIs. It is also obvious, that Edge/Fog hardware tend to have very broad interoperability needs ranging from GigE Vision, over Profinet to MQTT and Wi-Fi. Overall, different standards and protocols are needed for different applications and their specific requirements.

	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

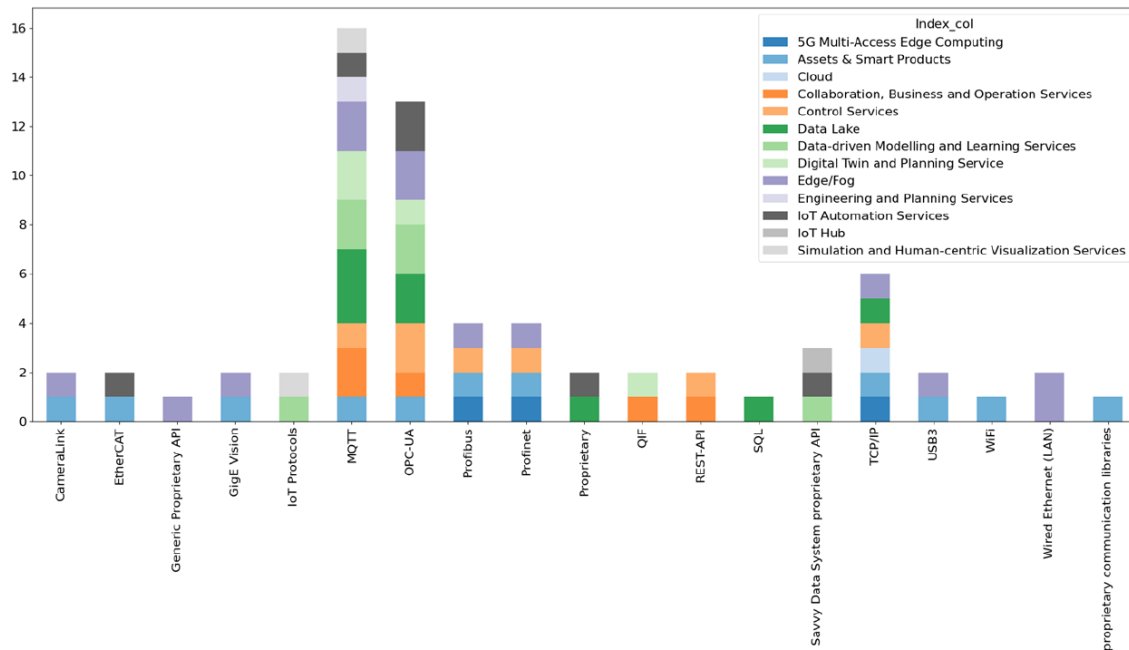


Figure 20 Standards, protocols and frameworks for ZDM currently used by the pilots

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

## 4 QU4LITY Semantic Interoperability

Semantic interoperability refers to the ability of information systems to exchange data unambiguously with shared meaning, which enables machine computable logic, inference, knowledge discovery, and data federation between information systems. Semantic interoperability is concerned with the packaging of data (syntax) as well as with the simultaneous transmission of the meaning with the data (semantics). This is accomplished by adding data about the data, called metadata, linking each data element to a controlled, shared vocabulary. The meaning of the data is transmitted with the data itself, in an information package that is independent of any information system. It is this shared vocabulary, and its associated links to an ontology, which provides the foundation and capability of machine interpretation, inferences, and logic.


To achieve semantic interoperability across different ZDM equipment and processes, a range of data models and vocabularies are. The specification of these data models should be based on existing standards for representing plants, production processes and quality processes information. The specifications of various openly accessible digital models, such as MPFQ-model (Material, Production Process, Product Functions/Features, Product Quality), will provide reference mechanisms (e.g., APIs and tools for Create-Read-Update-Delete (CRUD) operations with bindings in different languages and formats) for implementing similar models for QU4LITY project.

### 4.1 QU4LITY Semantic Data Models and Ontologies

#### 4.1.1 QU4LITY Semantic Data Models

To integrate all relevant elements that affect product and process quality, a RMPFQ-model is developed in Task T2.5 which focuses on "QU4LITY Digital Models and Vocabularies". This model is based on the previously developed MPFQ model which has been introduced in the previous version of this report. The previous MPFQ model only covers the processes of assembly manufacturing covering Material, Process, Function and Quality. In QU4LITY project, pilots focusing on both machining and assembly processes are included. Therefore, the RMPFQ model is developed by adding a Resource element. The definition of each element is listed as follows:

- Manufacturing Resource, according to ISO 15531, represents the devices, tools and means, at the disposal of the enterprise to produce goods and services, but except raw material and final product components,
- Material represents everything that is needed to produce a certain product or product component, which may include raw materials, pre-products, consumables, operating supplies, product components and assemblies,
- Manufacturing Processes are defined as processing and transforming materials into the final goods by using machines, tools and human labour. This process is defined within the plant engineering,
- Product Functions / Features represent the distinguished characteristics of a product item, which may include functionalities like specific tasks, actions or processes that the product is able to perform; and/or other features like performance,

	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

- Product Quality (Q) is defined as, according to DIN EN ISO 9000, the degree of conformance of final product functions and features to designed requirements.

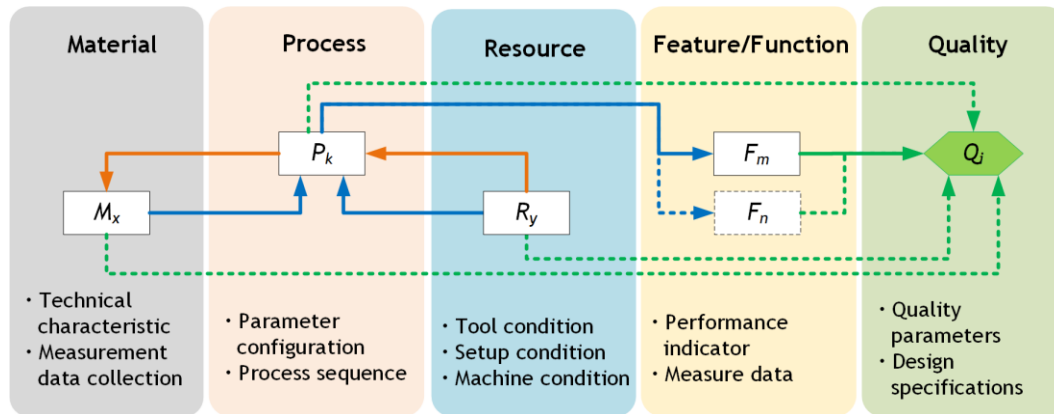


Figure 21 RMPFQ-model elements and their interrelations

Figure 21 shows the elements of the proposed RMPFQ-model and their interrelations, as well as some data related to these elements. First, a given workpiece (M) is machined by machining resources (R), e.g. a given setup (fixturing and associated tooling) and a cutting tool (R), through a planned machining process (P), composing the RPM interactions (marked with orange lines). Second, the machining process (P) uses input material (M) and resources (R) to produce one or more features (F), composing the RPMF interactions (marked with blue lines). Moreover, all the RMPF-elements may also have straightforward impact on the quality (Q) of the machined workpiece (marked with green lines). There also exist relations among different resources, i.e. machine, setup, and cutting tool.

This model covers the most critical factors that affect the quality of a product or process. Each of the elements can be further decomposed into lower-level components in practical applications. More details are introduced in D2.10.

#### 4.1.2 QU4LITY Ontologies

The structure of the QU4LITY Ontologies is as shown in Figure 22. It includes a series of ontologies corresponding to different levels. The core of these ontologies is the QU4LITY Domain Ontology which is based on the IOF-CORE Ontology as introduced in previous sections. Under that, two subdomain ontologies, i.e. process-oriented subdomain ontology and machine-oriented subdomain ontology will be developed corresponding to the two types of the QU4LITY pilot. Finally, several application ontologies based on certain pilots will be developed to demonstrate how the ontologies are applied in real use cases.

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

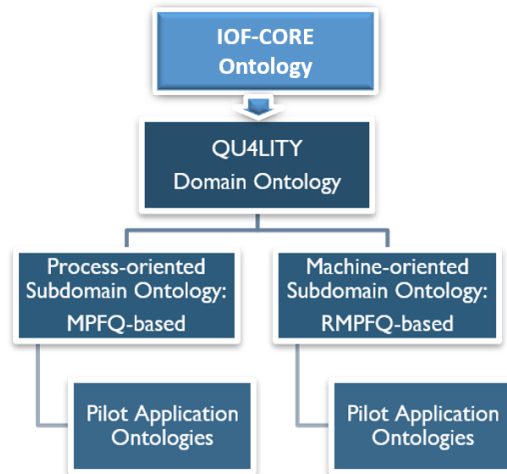


Figure 22 Structure of the QU4LITY Ontologies

The general structure of the QU4LITY Domain ontology follows the IOF-CORE ontology with more subclasses being added. The vocabularies of the ontology are organized according to the RMPFQ model as shown in Figure 23.

The lowest ontological level is the application-level ontology, which aims to represent specific application cases with highly specialized classes and individuals such as a device from a specific manufacturer, a work station, a production line etc. The application ontology will often use or reference domain ontologies to construct ontological classes and relationships between classes. For QU4LITY pilots, application ontologies can be developed follow the QU4LITY domain ontology introduced above. In D2.10, several application ontologies are introduced with detailed application background and development approach.

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

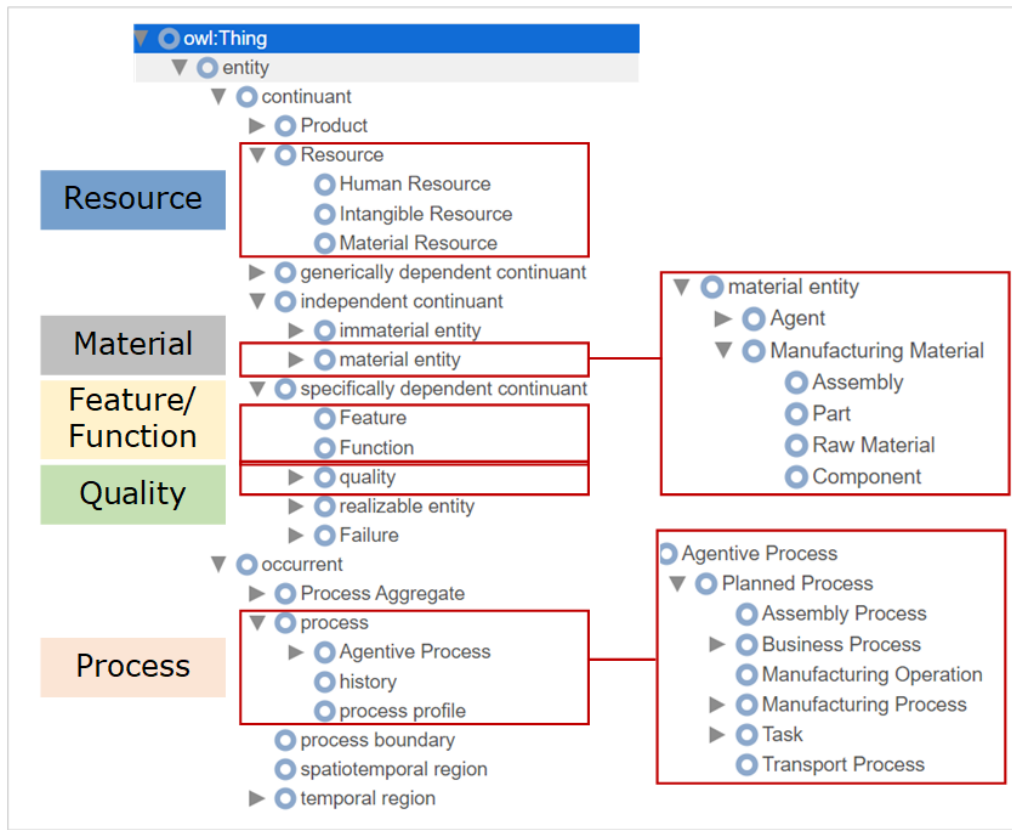



Figure 23 Structure and main classes of QU4LITY domain ontology

#### 4.1.3 Application of QU4LITY Data Models and Ontologies

The main target of QU4LITY data models and ontologies is to enhance interoperability between different sources of data and information. Depending on the application scenario, customized application ontologies can be developed under the structure of the QU4LITY domain ontology as introduced above. The RMPFQ data model can be used to identify and organize relevant factors that can affect the quality of a produce or a process. More details can be found in D2.10 which introduces detailed applications cases.

Another application of the QU4LITY data models and ontologies is to support semantic-driven digital twin models. Conventional digital twin models are lack of autonomous capabilities which are critical for realizing the AQ vision. Domain ontologies enable to capture and summarize intuitive information in a complex system using standardized languages. Augmented semantic capabilities can be added to digital twins by integrating semantic modelling technologies, thus to identify the dynamics of virtual model evolution and enhancing the decision-making capabilities. An exemplary semantic-driven digital twin model for machining processes have been developed based on the QU4LITY data models and ontologies. The general structure of this model is presented in Figure 24, which consists of five main components, including physical manufacturing system, virtual models, data management and services. The data management component is the core of the proposed semantic-

	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

driven DT model as it connects all the other components. The data collected from the elements of the proposed RMPFQ-model are also integrated in this component.

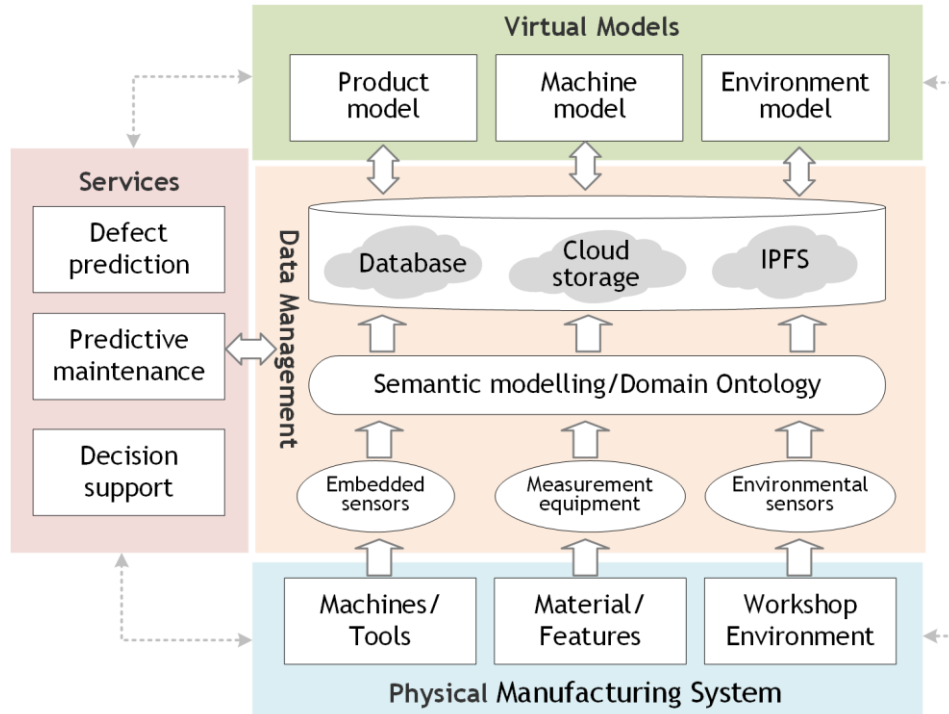


Figure 24 Semantic-driven digital twin model for machining processes

The data models and ontologies are applied mainly in three pilots including the Airbus production system trade space framework and GF machining abnormal detection system process quality management.

The application ontology of Airbus pilot aims to integrate information about system requirements and behaviour models, and then provide support for simulation. As the first step, both ontology and behaviour model (SysML model) will take process diagram as input to assure the alignment. The ontology will provide input for simulation (in JSON or XML). Once the knowledge base is created, in future phases, the ontology will be updated automatically according to the behaviour models.

The GF pilot application ontology follows the machine-oriented subdomain ontology. It aims to formalize the domain knowledge of the machining process to provide support for defect detection, predictive maintenance, and process optimization etc. The domain experts including the machining engineers, machine manufacturers, process planning experts etc., work together to define the machining process, operation parameters, possible defects and data collection approaches etc.

More details about the applications are introduced in WP2 deliverable D2.10 and some on-going works will be presented in WP7 pilot achievements by the end of the project.



QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

#### 4.1.4 Semantic Models and Vocabularies implementation

##### 4.1.4.1 Description and Usage

Semantic Models and Vocabularies aims to provide a range of data models and vocabularies to drive the flow and exchange of digital data across different ZDM equipment and processes. It is one of the key enablers to enhance the interoperability between different components of the reference architecture. The digital models are summarized through the RMPFQ model; and the vocabularies are specified in a series of ontologies including the QU4LITY domain ontology and multiple pilot application ontologies.

##### 4.1.4.2 Relation with the Reference Architecture

This enabler is the foundation of the “Digital Models and Vocabularies” layer of the Reference Architecture (see Figure 1 above). It has close connections with the Interoperability Assurance Layer. It takes input from Interoperability Assurance Layer about user stories and stakeholder’s requirements for assuring interoperability among components. In return, it provides semantic models to enable the interoperability among ZDM components and digital platforms. Besides, it also provides data models and semantic models to support data-driven services and digital twin modelling components.

##### 4.1.4.3 Dependencies

The QU4LITY domain ontology and relevant application ontologies are constructed based on the IOF-Core ontology which refers to the Basic Formal Ontology (BFO).

##### 4.1.4.4 Availability

The current version of this ontology is available, read-only, on Webprotégé<sup>52</sup>. For editing and redesign, it can be exported in different format such as RDF, XML, Turtle, OWL etc.

The IOF-Core ontology is developed by the IOF CORE Working Group<sup>53</sup>. It is still under active development. A repository of the latest version of the Core ontology is shared on Github<sup>54</sup> which is open available.

##### 4.1.4.5 Documentation

More details about the usage of the enabler are introduced in WP2 deliverable D2.10 with application examples based on multiple pilots.

<sup>52</sup> <https://webprotege.stanford.edu/#projects/8b81796d-6c0c-4cfb-875a-df68a444c1af/edit/Classes>

<sup>53</sup> <https://www.industrialontologies.org/top-down-wg/>

<sup>54</sup> <https://github.com/NCOR-US/IOF-BFO/tree/IOF-Core-2020>

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

## 4.2 Lightweight Digital Models for ZDM (FAR-EDGE/PROPHECY data model)

### 4.3 Lightweight Digital Models

The Lightweight Digital Models (LDM) offers a simplified consolidation and management of digital information across distributed applications. The LDM were developed in the scope of background projects of the partners (i.e. H2020 FAR-EDGE and H2020 PROPHECY).

#### 4.3.1 Description and Usage

The LDM are used to support the representation of factory data, metadata of different manufacturing data sources and analytics. The following core entities are used:

For the factory Data and Metadata:

- **Data Source Definition (DSD):** Defines the properties of a data source on the shop floor, such as a data stream from a sensor or an automation device.
- **Data Interface Specification (DI):** The DI is associated with a data source and provides the information need to connect to it and access its data, including details like network protocol, port, the network address and more.
- **Data Kind (DK):** Specifies the semantics of the data source. The DK can be used to define virtually any type of data in an open and extensible way.
- **Data Source Manifest (DSM):** Specifies a specific instance of a data source in-line with its DSD, DI and DK specifications. Multiple manifests (i.e. DSMs) are therefore used to represent the data sources that are available in the factory in the scope of the predictive maintenance platform.
- **Observation:** Models and represents the actual dataset that stems from an instance of a data source that is represented through a DSM. Hence, it references a DSM, which drives the specification of the types of the attributes of the Observation in-line with the DK that facilitates the discoverability of the data. An Observation is associated with a timestamp and keeps track of the location of the data source in case it is associated with mobile (rather than a stationary) edge node. An Observation has a location attribute (virtual or physical), which identifies the placement of the data source. The value type of observation is a complex object which is described with the DK entity that an Observation references. Hence, an observation can depict multiple raw measurements coming from a machine or a single value (i.e. the number of cycles/m of a rotor) or even an Analytics Processor result (i.e. the calculated RUL of a machine).
- **Edge Gateway:** Models an edge gateway of an edge computing deployment of the predictive maintenance platform. In the scope of deployment of the platform, data sources are associated with an edge gateway. This usually implies not only a logical association but a physical association as well, i.e. an edge gateway is deployed at a station and manages data sources in close physical proximity to the station.

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

For the factory Data Analytics and Metadata:

- **Analytics Processor Definition (APD):** This specifies a processing function to be applied on one or more data sources. Three processing functions are defined, including functions that pre-process that data of a data source (i.e. Pre-Processors), functions that store the outcomes of the processing (i.e. Store Processors) and functions that analyse the data from the data sources (i.e. Analytics Processors). These three types of processors can be combined in various configurations over the data sources in order to define different analytics workflows.
- **Analytics Processor Manifest (APM):** This represents an instance of a processors that is defined through the APD. The instance specifies the type of processors and its actual logic through linking to a programming function. In the case of DataCROP, the latter is a class implemented in the Java language.
- **Analytics orchestrator Manifest (AM):** An AM represents an entire analytics workflow. It defines a combination of analytics processor instances (i.e. of APMs) that implements a distributed data analytics task. The latter is likely to span multiple edge gateways and to operate over their data sources.

LDM have been used in several pilots and open calls (e.g. RIASTONE, THYSSENKRUPP, IDEAL) as a mean to facilitate the QARMA4Industry algorithm for estimating RUL values. More details about LDM data model can be found in section 5.3 of the first version of this deliverable namely D3.13.

#### 4.3.2 Relation with the Reference Architecture

The LDM is placed at the “Digital Models and Vocabularies” layer (see Figure 1 above) as they may be used as the basis for automated configuration, simulation and field abstraction.

#### 4.3.3 Dependencies

As a component the LDM has the following environment and software dependencies:

- MongoDB >= 3.6.4
- Node.js >= 10.1.0
- npm >= 5.6.0

#### 4.3.4 Availability

LDM is part of the DataCROP platform and since it is an Open-Source software is offered both as source code at GitHub<sup>55</sup> but as well as a containerized solution at Docker Hub<sup>56</sup>. In the following two sub-sections, we provide details for this availability.

<sup>55</sup> <https://github.com/qu4lity/data-crop>

<sup>56</sup> <https://hub.docker.com/u/faredge>

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

#### 4.3.4.1.1 GitHub Distribution

LDM source code is offered under the Apache License 2.0 and its components are available from GitHub<sup>57</sup>. LDM is consisted of the following components:

- Model Repository
- Digital Models

#### 4.3.4.1.2 Docker Distribution

DataCROP dockerized components are offered thru Docker Hub and various deployment options are also available by offering the equivalent YAML files for Docker Compose and Docker Swarm.

DataCROP DDA Docker Hub availability is provided below:

- Model Repository
  - <https://hub.docker.com/repository/docker/faredge/model-repository>

### 4.3.5 Installation guidelines

It follows the same principles as the ones described in section 2.2.1.5 above.

<sup>57</sup> <https://github.com/qu4lity/data-crop>

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

## 5 Conclusions

This deliverable has dealt with two important issues for the QU4LITY digital infrastructure and enablers, namely:

- The integration of different digital enablers into turnkey solutions, in-line with the reference architecture of the project (QUALITY-RA).
- The interoperability across diverse components of these turnkey solutions including both syntactic and semantic interoperability.

These issues have been addressed in the deliverable in-line with the QUALITY-RA and based on the following principles:

- **Adherence to industry best practices**, notably best practices for the modular packaging and distribution of the QU4LITY components, using popular platforms for stack management and source code management such as Docker and Github. Likewise, known approaches for syntactic and semantic interoperability have been followed.
- **Development of custom value-added infrastructures**, for edge computing deployments and blockchain applications. These two special purpose infrastructures have been introduced to address specific requirements of ZDM use cases, such as the need for deploying solutions close to the field (i.e. edge computing solutions) and the need for sharing and validating ZDM related data across the supply chain (i.e. blockchain solutions).
- **Reuse of partners' technologies**: Whenever possible, background technologies and platforms of the partners have been reused. This is for example the case with the interoperability solutions of the project, where a middleware platform of one of the partners has been used for syntactic interoperability, as well as with the semantic interoperability solutions where digital models specified in other projects and extended in QU4LITY has been used.
- **Packaging of existing digital enablers**: In-line with the presented approach to integration, the partners have worked on the packaging of existing digital enablers

The integration and interoperability solutions that are presented in this deliverable are general and applicable to the various QU4LITY pilots and use cases. They are also appropriate for supporting enablers and use cases that aligned to the data-driven autonomous quality concept. This final version of the deliverable was destined to present the actual implementation of the integrated enablers and their integration/use in the QU4LITY pilot systems.

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

## 6 References

- 
- [1] Docker, Inc., "Docker Documentation", available at: <https://docs.docker.com/>, last accessed: March 2020

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

## List of figures

Figure 1 QU4LITY Reference Architecture (Q-RA) (D2.12) .....	9
Figure 2 Valid XML message for endpoint.....	18
Figure 3 JSON messages format for endpoint client.....	18
Figure 4 WSO2 Converter API .....	18
Figure 5 Data Mapper.....	19
Figure 6 Input Message .....	20
Figure 7 Output Sample Message.....	20
Figure 8 Sampling Converter in Node-RED .....	21
Figure 9 Welcome view of event and alarms generated .....	35
Figure 10 XL-SIEM New agent addition .....	36
Figure 11 Entering user data .....	36
Figure 12 New type of events creation .....	36
Figure 13 XL-SIEM events.....	37
Figure 14 XL-SIEM alarms .....	37
Figure 15: The QARMA algorithm used in the wrapper as a library .....	43
Figure 16 Overview of the functional architecture of trade space framework for aircraft production system design .....	45
Figure 17 The roles of application ontology and 3D simulation in the trade space framework.....	46
Figure 18 QU4LITY GitHub organization. ....	48
Figure 19 Overview of interoperability standards, protocols and frameworks for ZDM .....	50
Figure 20 Standards, protocols and frameworks for ZDM currently used by the pilots .....	51
Figure 21 RMPFQ-model elements and their interrelations .....	53
Figure 22 Structure of the QU4LITY Ontologies.....	54
Figure 23 Structure and main classes of QU4LITY domain ontology .....	55
Figure 24 Semantic-driven digital twin model for machining processes .....	56

<b>QU4LITY</b>	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

## List of tables


Table 1 List of the QU4LITY digital enablers .....	11
Table 2 DataCROP Docker Compose script .....	70
Table 3 QU4LITY Cloud Infrastructure Docker Compose script .....	71
Table 4 XL-SIEM Initialization.....	72
Table 5 XL-SIEM Json configuration .....	74
Table 6 XL-SIEM Docker launch .....	74



QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

## List of Abbreviations

AM	Analytics orchestrator Manifest
AMQP	Advanced Message Queuing Protocol
AP	Analytics Processor
APD	Analytics Processor Definition
APM	Analytics Processor Manifest
AQ	Autonomous Quality
CPS	Cyber Physical Systems
CRUD	Create-Read-Update-Delete
DApp	Distributed Application
DataCROP	Data Collection Routing & Processing
DDA	Distributed Data Analytics
DES	Discrete Event Simulation
DK	Data Kind
DLT	Distributed Ledger Technology
DSD	Data Source Definition
DSM	Data Source Manifest
DT	Digital Twin
E2E	End to End
EAE	Edge Analytics Engine
EI	Enterprise Integrator
ESB	Enterprise Service Bus
HPC	High Performance Computing
IOF	Industrial Ontologies Foundry
IoT	Internet of Things
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
LDM	Lightweight Digital Models
ML	Machine Learning
OS	Open Source
PL	Private Ledgers
QA	Quality Assessment
QCH	Quality Clearing House
QCH	Quality Clearing House
RA	Reference Architecture
RAMI4.0	Reference Architecture Model Industrie 4.0
RDF	Resource Description Framework
REST	Representational State Transfer
RUL	Remaining Useful Life
SID	Secure Identity Directory
SIEM	Security Information and Event Management
SIEM	Security Information and Event Management
SMB	Secure Messaging Board
SPT	Security Privacy and Trust
VCL	Value Chain Ledger
VM	Virtual Machine
WSO2	Web Services Oxygenated 2
XML	eXtensible Markup Language
YAML	Ain't Markup Language
ZDM	Zero Defect Manufacturing

	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

## Appendix I – Digital Enablers’ Docker Compose Scripts

### DataCROP

```
# docker-compose -f test.yml -p test up

version: '3.3'


services:

  analytics-dashboard:
    image: faredge/analytics-dashboard:1.0.1
    ports:
      - 8000:8000
    environment:
      NAME: 'FAR-EDGE Analytics Dashboard'
      NODE_ENV: development
      OPEN_API_FOR_ANALYTICS_BASE_URL: http://localhost:4444/api
      MODEL_REPOSITORY_BASE_URL: http://localhost:8888/api
    depends_on:
      - model-repository
      - open-api-for-analytics
    restart: on-failure:5

  cloud-storage:
    image: mongo:3.6.4
    ports:
      - 27017:27017
    volumes:
      - cloud-data:/data/db
      - ./data/cloud:/docker-entrypoint-initdb.d:ro
    restart: on-failure:5

  configuration:
    image: confluentinc/cp-zookeeper:5.0.0
    ports:
      - 2181:2181
    environment:
      ZOOKEEPER_CLIENT_PORT: 2181
      ZOOKEEPER_SERVER_ID: 42
      ZOOKEEPER_TICK_TIME: 2000
    restart: on-failure:5

  data-router-and-preprocessor:
    image: faredge/data-router-and-preprocessor:1.0.3
    ports:
      - 7777:7777
```


	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

```

environment:
  NAME: 'FAR-EDGE Data Router and Preprocessor'
  NODE_ENV: development
  LOG_LEVEL: debug
  API_BASE_URL: http://localhost:7777/api
  MAX_REQUEST_BODY_SIZE: 100
  MONGODB_HOST: edge-storage
  MONGODB_NAME: device-registry
  MONGODB_POOL_SIZE: 20
  MONGODB_PORT: 27017
  KAFKA_BROKER_HOST: streams
  KAFKA_BROKER_PORT: 9092
  MODEL_REPOSITORY_BASE_URL: http://model-repository:8888/api
  EDGE_GATEWAY_MAC_ADDRESS: E5-E5-B9-5C-45-10
  EDGE_GATEWAY_ID: befa2ec0-2e9b-4e97-ad0a-c9fe91c1b55c
depends_on:
  - edge-storage
  - messages
  - model-repository
  - streams
restart: on-failure:5

edge-analytics-engine:
  image: faredge/edge-analytics-engine:1.0.3
  ports:
    - 9999:9999
  environment:
    NAME: 'FAR-EDGE Edge Analytics Engine'
    NODE_ENV: development
    LOG_LEVEL: debug
    API_BASE_URL: http://localhost:9999/api
    MAX_REQUEST_BODY_SIZE: 100
    MONGODB_HOST: edge-storage
    MONGODB_PORT: 27017
    MONGODB_NAME: analytics-storage
    MONGODB_POOL_SIZE: 20
    MODEL_REPOSITORY_BASE_URL: http://model-repository:8888/api
    DATA_ROUTER_AND_PREPROCESSOR_BASE_URL: http://data-router-and-
preprocessor:7777/api
    EDGE_GATEWAY_ID: befa2ec0-2e9b-4e97-ad0a-c9fe91c1b55c
  volumes:
    - ./processors:/processors:ro
  depends_on:
    - data-router-and-preprocessor
    - edge-storage
    - model-repository
  restart: on-failure:5

```

	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

```

edge-storage:
  image: mongo:3.6.4
  ports:
    - 27018:27017
  volumes:
    - edge-data:/data/db
    - ./data/edge:/docker-entrypoint-initdb.d:ro
  restart: on-failure:5

```

```

message-echoer:
  image: aksakalli/mqtt-client:latest
  depends_on:
    - messages
  command: sub -h messages -t "#" -v
  restart: on-failure:5

```

```

messages:
  image: eclipse-mosquitto:latest
  ports:
    - 1883:1883
    - 9001:9001
  volumes:
    - mosquitto-config:/mosquitto/config
    - mosquitto-data:/mosquitto/data
    - mosquitto-log:/mosquitto/log
  restart: on-failure:5

```

```


model-repository:
  image: faredge/model-repository:1.0.3
  ports:
    - 8888:8888
  environment:
    NAME: 'FAR-EDGE Model Repository'
    NODE_ENV: development
    LOG_LEVEL: debug
    API_BASE_URL: http://localhost:8888/api
    MAX_REQUEST_BODY_SIZE: 100
    MONGODB_HOST: cloud-storage
    MONGODB_PORT: 27017
    MONGODB_NAME: model-storage
    MONGODB_POOL_SIZE: 20
  depends_on:
    - cloud-storage
  restart: on-failure:5

```

```

open-api-for-analytics:
  image: faredge/open-api-for-analytics:1.0.3
  ports:

```

	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU


```

- 4444:4444
environment:
  NAME: 'FAR-EDGE Open API for Analytics'
  NODE_ENV: development
  LOG_LEVEL: debug
  API_BASE_URL: http://localhost:4444/api
  MAX_REQUEST_BODY_SIZE: 100
  MONGODB_HOST: cloud-storage
  MONGODB_PORT: 27017
  MONGODB_NAME: analytics-storage
  MONGODB_POOL_SIZE: 20
  KAFKA_BROKER_HOST: streams
  KAFKA_BROKER_PORT: 9092
  MODEL_REPOSITORY_BASE_URL: http://model-repository:8888/api
depends_on:
  - cloud-storage
  - model-repository
  - streams
restart: on-failure:5

random-data-publisher:
  image: faredge/mqtt-random-data-publisher:1.0.3
  environment:
    NAME: 'MQTT Random Data Publisher'
    NODE_ENV: development
    LOG_LEVEL: debug
    MQTT_BROKER_URL: mqtt://messages:1883
    MQTT_TOPIC: bob/temperature
    MIN_VALUE: 35
    MAX_VALUE: 42
    VALUE_INTERVAL: 10
  depends_on:
    - messages
  restart: on-failure:5

streams:
  image: confluentinc/cp-enterprise-kafka:5.0.0
  ports:
    - 9092:9092
    - 29092:29092
  environment:
    KAFKA_BROKER_ID: 23
    KAFKA_ZOOKEEPER_CONNECT: 'configuration:2181'
    KAFKA_LISTENER_SECURITY_PROTOCOL_MAP:
PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT
    KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT
    KAFKA_ADVERTISED_LISTENERS:
PLAINTEXT://streams:9092,PLAINTEXT_HOST://localhost:29092

```

	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

```

KAFKA_BROKER_RACK: r1
KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
KAFKA_DELETE_TOPIC_ENABLE: 'true'
KAFKA_AUTO_CREATE_TOPICS_ENABLE: 'true'
KAFKA_JMX_PORT: 9991
depends_on:
  - configuration
restart: on-failure:5

volumes:
  cloud-data:
  edge-data:
  mosquitto-config:
  mosquitto-data:
  mosquitto-log:

```

Table 2 DataCROP Docker Compose script

## Cloud Infrastructure


```

version: "3"

services:
  nginx:
    hostname: nginx
    image: nginx:latest
    networks:
      - hostnet
    ports:
      - "8080:80"
    depends_on:
      - qu4lity_cloud_bridge
    volumes:
      - ./nginx_conf/.htpasswd:/etc/nginx/.htpasswd
      - ./nginx_conf/nginx.conf:/etc/nginx/nginx.conf

  qu4lity_cloud_bridge:
    hostname: qu4lity_cloud_bridge
    image: node:14
    networks:
      - hostnet
    expose:
      - "9000"
    user: "node"
    working_dir: /home/node/app
    depends_on:
      - mpfq_mariadb
    environment:
      - NODE_ENV=production

```

	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

```

- PORT=9000
- MPFQ_MARIADB_HOST=mpfq_mariadb
- MPFQ_MARIADB_PORT=3306
- MPFQ_MARIADB_DB=whr_mpfq_relational
- MPFQ_MARIADB_USER=root
- MPFQ_MARIADB_PASSWORD=r00t
volumes:
- ./node_conf/qu4lity-cloud-bridge:/home/node/app
command: "npm start"


mpfq_mariadb:
  hostname: mpfq_mariadb
  image: mariadb:latest
  networks:
    - hostnet
  ports:
    - "3306:3306"
  volumes:
    - container-volume:/var/lib/mysql
    - ./mariadb_conf:/docker-entrypoint-initdb.d/
  environment:
    - MYSQL_ROOT_PASSWORD=r00t

volumes:
  container-volume:

networks:
  hostnet:

```

Table 3 QU4LITY Cloud Infrastructure Docker Compose script

	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

## Appendix II – Native Installation Scripts

### XL-SIEM

Step A:


```
echo "Copy json" >> $USER_PATH/init.log
cp /mnt/*.json $USER_PATH/scenariolayout.json
echo "Loading data " >> $USER_PATH/init.log
python3 init_citef.py
sleep 1200
echo "Launching Dockers" >> $USER_PATH/init.log
/home/cw_atos/reset.sh
```

Table 4 XL-SIEM Initialization

Step B:

```
from pathlib import Path
import json
import logging
web = "/home/cw_atos/dashboard/docker/web/context/config.conf"
topology = "/home/cw_atos/topology/docker/storm/context/conf/XL-SIEM.conf"
agent = "/home/cw_atos/dashboard/docker/mariadb/sql/data/04-agents.sql"
SCENARIO_CONFIGURATION = "/home/cw_atos/scenariolayout.json"
RABBITMQ_VM_TYPE = "MESSAGE_BROKER"
ADR_VM_TYPE = "ADR"
AGENT_VM_TYPE = "MONITORING_AGENT"
logger = logging.getLogger(__name__)
LOG_FORMAT = '%(asctime)s %(levelname)s: %(message)s - %(name)s - %(funcName)s - %(lineno)d'
PARSER_LOG_FILE = "/home/cw_atos/parser_configurator.log"
logging.basicConfig(format=LOG_FORMAT, filename=PARSER_LOG_FILE,
filemode='w+', level=logging.INFO)
logger.warning('Loading Scenario Data ')
scenario_json_file = Path(SCENARIO_CONFIGURATION)
scenario_file = open(SCENARIO_CONFIGURATION, "r")
data_scenario = json.load(scenario_file)
def initialdata(type):
    machines = (vm for vm in data_scenario['virtualMachines'] if
vm['configuration'] and ('type' in vm['configuration'])
and vm['configuration']["type"] == type)
    ip = None
    for vm in machines:
        logger.warning('Looking for the right machine')
        for nic in vm['nics']:
            if nic['isManagement']:
```



	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

```

        ip = nic['ip']
        logger.warning('The IP of the ' + str(type) + ' is ' +
str(ip) )
    return ip
ipadr = initialdata(ADR_VM_TYPE)
ipmb = initialdata(RABBITMQ_VM_TYPE)
ipagent = initialdata(AGENT_VM_TYPE)
logger.warning('Changing Configuration of the topology configuration ')
with open(topology, "r") as f:
    lines = f.readlines()
    f.close()
#print(lines)
logger.warning('Changing the database an MB ip ')
ddb = 'databaseIP = ' + ipadr + '\n'
lines.insert(87, ddb)
old_line = lines[88]
lines.remove(old_line)
mb = 'serverRabbitMQ = ' + ipmb + '\n'
lines.insert(15, mb)
old_line = lines[16]
lines.remove(old_line)
with open(topology, "w") as f:
    lines = "".join(lines)
    f.write(lines)
    f.close
logger.warning('Changing Configuration of the web ')
with open(web, "r") as f:
    web_lines = f.readlines()
    f.close()
ip_topology = 'ip=' + ipadr + '\n'
web_lines.insert(2, ip_topology)
old_line = web_lines[3]
web_lines.remove(old_line)
ip_ddb = 'host=' + ipadr + '\n'
web_lines.insert(27, ip_ddb)
old_line = web_lines[28]
web_lines.remove(old_line)
with open(web, "w") as f:
    web_lines = "".join(web_lines)
    f.write(web_lines)
    f.close

logger.warning('Adding the Cyber-Agent ')

with open(agent, "r") as f:
    agent_lines = f.readlines()
    f.close()
insert_agent = agent_lines[3]

```

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

```

a = insert_agent.split('\\"')
a[3] = ipagent
insert_newagent = '\\\".join(a)
agent_lines.remove(insert_agent)
agent_lines.insert(3, insert_newagent)
with open(agent, "w") as f:
    agent_lines = "".join(agent_lines)
    f.write(agent_lines)
    f.close

```

Table 5 XL-SIEM Json configuration

#### Step C:

```

echo "Launching Dockers" >> $USER_PATH/init.log
cd /home/cw_atos/dashboard/docker/mariadb
./install.sh -s=cyber -p=3306
/home/cw_atos/dashboard/docker/web/install.sh -s=atos -p=8080
cd /home/cw_atos/topology/docker/storm
docker-compose -f storm.yml up -d

```

Table 6 XL-SIEM Docker launch

QU4LITY	Project	QU4LITY - Digital Reality in Zero Defect Manufacturing		
	Title	Library of Integrated, Interoperable Digital Enablers (Final Version)	Date	30/07/2021
	Del. Code	D3.14	Diss. Level	PU

## Partners:

