DIGITAL MANUFACTURING PLATFORMS FOR CONNECTED SMART FACTORIES

D3.13 Library of Integrated, Interoperable Digital (V1)

Deliverable Id :	**D3.13**
Deliverable Name :	**Library of Integrated, Interoperable Digital Enablers (Version 1)**
Status :	**Final**
Dissemination Level :	**PU**
Due date of deliverable :	**31/03/2020**
Actual submission date :	**14/04/2020**
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, ENG, ATLAS, ATOS, TNO, FHG**

Abstract: This deliverable presents the project's approach to integrating differnet 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.

Co-funded by the Horizon 2020 Programme of the European Union

Contents

HISTORY 6
1. Introduction 8
1.1 Scope and Purpose of the Deliverable 8
1.2 Methodology 9
1.3 Relation to Other Deliverables 10
1.4 Deliverable Structure 11
2. Packaging & Integration of QU4LITY Digital Enablers 12
2.1 QU4LITY Enablers Integration Concept and Elements of the Integration Solutions 12
2.1.1 Overview 12
2.1.2 Software Packaging 12
2.1.3 Container Tools 12
2.1.3.1 Kubernetes 12
2.1.3.2 Docker Compose 12
2.1.3.3 Docker Swarm 13
2.1.4 Management & Monitoring Tools 13
2.1.4.1 Rancher 13
2.1.4.2 Sumo Logic 13
2.1.4.3 Portainer 13
2.1.4.4 Datadog 14
2.1.4.5 Prometheus 14
2.1.5 Code Management 14
2.1.5.1 GitHub 14
2.1.5.2 GitLab 14
2.1.6 Repository Management 15
2.1.6.1 Nexus Repository Manager 15
2.1.6.2 Docker Hub 15
2.1.6.3 JFrog 15
2.2 QU4LITY Integration Requirements 15
2.3 QU4LITY Integration Infrastructure 19
2.3.1 Software Packaging with Docker images 19
2.3.2 Container Tool with Docker Compose. 20
2.3.3 Management/Monitoring with Portainer. 20

2.3.4 Code Management with GitHub. 21
2.3.5 Repository Management with Docker Hub. 22
2.4 QU4LITY Deployment Infrastructure for Edge Computing 22
2.4.1 Concept of Edge Computing 22
2.4.2 Virtualization (Virtual Machines) 24
2.4.3 Deployment of Dockerized workloads 25
2.4.4 Management of workloads 27
2.4.5 Use of the Edge Computing Deployment Infrastructure in QU4LITY ...28
2.5 QU4LITY Deployment Infrastructure for DLT Services 29
2.5.1 Concept of DLT Services in QU4LITY 29
2.5.2 QU4LITY Common DLT Infrastructure 30
2.5.3 Packaging and Distribution of QU4LITY DLT Services 31
2.5.4 Integrating QU4LITY DLT Services 32
2.5.5 Example Use Case 33
2.6 QU4LITY Integration Concept 35
2.6.1 Mapping of Digital Enablers to QU4LITY-RA 35
2.6.2 Integration Examples 37
3. Library of Packaged Enablers 39
3.1 Overview 39
3.2 FAR-EDGE IoT Analytics Platform 40
3.2.1 Description 40
3.2.2 Availability 41
3.2.2.1 FAR-EDGE GitHub Distribution 41
3.2.2.2 FAR-EDGE Docker Distribution 42
3.3 VTT OpenVA Platform 42
3.3.1 Description 42
3.3.2 Availability 43
3.4 RUL Prediction Solution 43
3.4.1 Description 43
3.4.2 Availability 44
3.5 Decision Support System (DSS) and Strategies for ZDM) 44
3.5.1 Description 44
3.5.2 Availability 45
3.6 Security Privacy and Trust (SPT) Enabler (ATOS) 45
3.6.1 Description 45

3.6.2 Availability 45
3.7 MGEP Converter Enabler 46
3.7.1 Description 46
3.7.2 Availability 46
3.8 Improved Failure Classification Enabler 46
3.8.1 Description 46
3.8.2 Availability 47
3.9 DLT Service: Quality Clearing House 47
3.9.1 Description 47
3.9.2 Availability 47
3.10 DLT Service: Decentralized Analytics Engine Configuration 48
3.10.1 Description 48
3.10.2 Availability 48
3.11 DLT Service: Secure Analytics Results Publishing 48
3.11.1 Description 48
3.11.2 Availability 48
4. QU4LITY Platform Interoperability Solutions 49
4.1 QU4LITY Platform Interoperability Approach 49
4.1.1 Interoperability in Industry 4.0 Types of Interoperability 49
4.1.2 Types of Interoperability 49
4.1.3 Interoperability in the QU4LITY-RA 50
4.2 QU4LITY Middleware for Platform Interoperability 51
4.2.1 Background and Rationale 51
4.2.2 Platform Description 51
4.2.3 Guide for Using the Platform in QU4LITY Use Cases 54
5. QU4LITY Semantic Interoperability Solutions 55
5.1 QU4LITY Semantic Interoperability Approach 55
5.2 QU4LITY Semantic Data Models and Ontologies 55
5.2.1 Background and State of the Art 55
5.2.2 Use of Ontologies and Digital Twins in QU4LITY 57
5.2.3 Guide for Using the Semantic Models in QU4LITY Use Cases 59
5.3 Lightweight Digital Models for ZDM 60
5.3.1 Background 60
5.3.1 Lightweight Digital Models for ZDM QU4LITY Extensions 63
5.3.1.1 Data Kind 63

5.3.1.2 Observation 63
5.3.1.3 AdditionalInformation 64
6. Conclusion and Future Outlook 65
7. References 66
List of figures 68
List of tables 69
List of Abbreviations 70
Partners: 72

HISTORY

Version	Date	Modification reason	Modified by
0.1	12/02/2020	Initial Table of Contents to be shared & discussed with WP leader and contributors, following the latest telco (11/02/2020) for this task	INTRASOFT
0.11	17/02/2020	Updated TOC based on partners' feedback	INTRASOFT
0.12	18/02/2020	Initial inputs from EPFL (on digital models) and Mon (on platform interoperability)	EPFL, MON
0.13	18/02/2020	Introduction & scope of the deliverable	INTRASOFT
0.15	20/02/2020	Revisions following inputs from TTT	TTT, INTRASOFT
0.16	09/03/2020	Inputs on Containers and Virtualization at the Edge	TTT
0.17	10/03/2020	Inputs on Stack Management and Code Management Tools	INTRASOFT
0.18	11/03/2020	Description of the Middleware Platform and its Packaging	MON
0.19	12/03/2020	Various Edits and Fine-Tuning	INTRASOFT
0.20	13/03/2020	Integration of Interoperability Inputs	MON, INTRASOFT
0.21	18/03/2020	Description of VTT's OpenVA Digital Enabler	VTT
0.22	18/03/2020	Updates to the deliverable structure following teleconference on 18/03	INTRASOFT
0.23	20/03/2020	Integration of inputs from ATLAS and MON in Section 3	INTRASOFT, ATLAS, MON
0.24	24/03/2020	Updates from TTT and EPFL on the use of edge nodes and semantic models	INTRASOFT, TTT, EPFL
0.25	26/03/2020	Edits on the Alignment to the QU4LITY-RA in Section 2	INTRASOFT
0.26	26/03/2020	Semantic Interoperability Section 4 Updates	INTRASOFT
0.27	27/03/2020	Inputs on DLT Enablers (Section 2 & 3)	ENG
0.28	27/03/2020	Addition of TNO and ATOS Enablers inputs	INTRASOFT, TNO, ATOS
0.29	27/03/2020	Updates on Containerization and Packaging Tools (Section 2)	INTRASOFT
0.30	30/03/2020	Overview of enablers and updates in Section 4	INTRASOFT

0.31	31/03/2020	Various edits, re-formatting of the document and initial quality control	INTRASOFT
0.32	31/03/2020	Preparation of version of quality control by QU4LITY partners	INTRASOFT
0.40	02/04/2020	Updates and fine-tuning in Section 4	INTRASOFT
0.50	09/04/2020	Review of the document and contribution to table 5	FHG-ILT
0.55	09/04/2020	Review of the document by the Technical Manager of the project	ENG
1.0	10/04/2020	Implementation of Revisions following quality control and reviews; Preparation of version for delivery to the Project Coordinator & Submission	INTRASOFT

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 in order 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:

- The QU4LITY Integration Concept, including mechanisms, infrastructures and tools for integrating different enablers in turnkey solutions.
- 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 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 Methodology

The methodology of the deliverable is divided in two parts:

1)Methods for selecting the integration and packaging infrastructure of the project. Given that the integration solution has an impact on all providers/vendors of digital solutions, a relevant survey has been distributed to the QU4LITY partners. The goal of the survey was to identify the needs and the constraints of the various components' vendors/providers in terms of the packaging and integration of their components. It was created with an outlook of using state of the art techniques for stack management and source code management, including popular tools with industrial relevance such as Docker and Github. Based on the outcomes of the survey, decisions about the infrastructure that was selected and established were taken. The overall approach will be validated based on the actual integration of various enablers in end-to-end solutions. Several examples of such integration efforts will be detailed as part of the next version of the present document/deliverable i.e. deliverable D3.14. Overall the methodology is illustrated in Figure 1 and comprises the following phases:

- **Phase 1: Needs Identification & Requirements Analysis**: Starting from a set of integration and packaging practices that are widely used in the industry, this Phase identified the requirements of the providers of the QU4LITY digital enablers. It focused on requirements associated with the integration and packaging of QU4LITY components. In this direction, a relevant survey was administered to the project partners and its results were analysed to elicit relevant requirements. The design of the survey assumed that QU4LITY will dispose with the identified best practices with minor deviations from them.
- **Phase 2: Infrastructure Selection, Design and Implementation**: Following the identification of requirements, this Phase prescribed, designed and implementing the project's infrastructure for integration and packaging of digital components/enablers. This infrastructure is described in Chapter 2 of this deliverable.
- **Phase 3: Enablers Integration – End-to-End Solutions**: This phase specified the digital enablers that will follow the project's integration guidelines. It also created some initial integration instances. However, more examples are to be provided as part of D3.14.

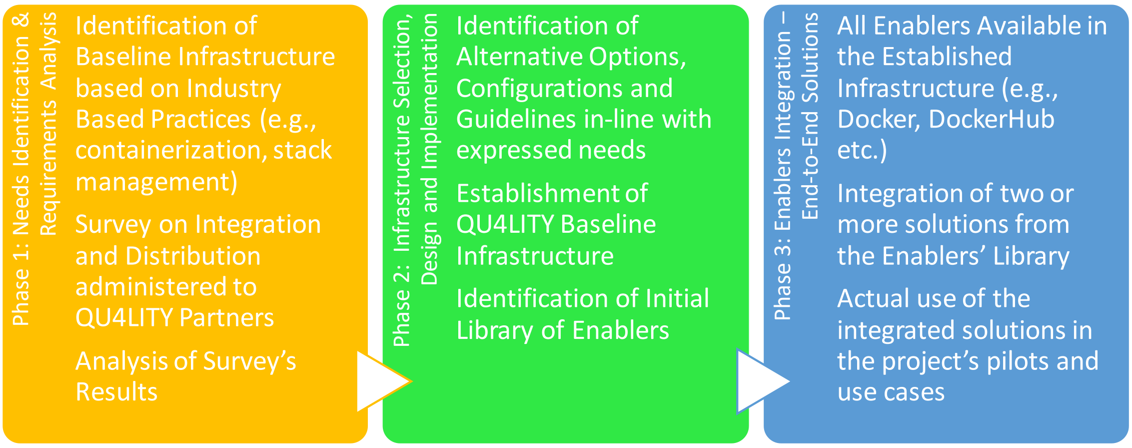

Figure 1: Three Phase Methodology for Establishing and Validating the Project' s Infrastructure for Components Packaging and Integration

2)Methods for implementing interoperability solutions. Given the different facets of interoperability (e.g., data interoperability, semantic interoperability, syntactic interoperability) different interoperability goals were identified along with proper solutions to achieve them. Specifically a solutions for the interoperability of industrial platforms and technologies (including digital automation platforms) is introduced to address the syntactic interoperability challenge, while a set of (semantic) digital models are proposed as a solution to the semantic interoperability of different CPS (Cyber Physical Systems) systems. In the scope of deliverable D3.14 (i.e. the next version/release of the present deliverable) concrete semantic interoperability use cases that will leverage the digital models of the project will be presented. Likewise, the middleware solution for platforms interoperability will be prototyped and used in practical syntactic interoperability use cases.

1.3 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 this deliverable. As such, the deliverable is relevant to the following WP3 deliverables that have been already delivered:

- D3.1 Connectivity Technologies for Autonomous Quality (Version 1).
- D3.3 HPC and Cloud Resources for ZDM (Version 1).
- D3.5 BigData and Analytics Infrastructure (Version 1).
- D3.7 Fog Nodes and Edge Gateways for ZDM deployments (Version 1).
- D3.9 QU4LITY SPT Framework (Version 1).
- D3.11 Permissioned Blockchain for ZDM (Version 1).

Note also that the second and final versions of the above deliverables will leverage the infrastructures of this deliverables and will produce relevant integrated digital enablers. Hence, the present deliverable is linked to D3.2, D3.4, D3.6, D3.8, D3.10

and D3.12 as well. The latter set of deliverables will consider the present one in integrating and packaging their enablers.

Also, the deliverable leverages inputs and results from:

- D2.11 Reference Architecture and Blueprints (Version 1), which provides the overall ZDM system integration concept of the project. The results of the present deliverable are compatible with D2.11 and support the integration of solutions in-line with the RA of the project.
- D2.9 QU4LITY Digital Models and Vocabularies (Version 1), 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. The present deliverable (D3.13) builds on these digital models in order to establish the project's semantic interoperability approach.

1.4 Deliverable Structure

The deliverable is structured as follows:

- Section 2 following this introductory section presents the main integration requirements for the QU4LITY digital enablers and the integration infrastructure established.
- Section 3 provides the library of digital enablers of the project, which follow or will follow the integration and packaging guidelines of this document.
- Section 4 focuses on the description of the project's platform interoperability solution.
- Section 5 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 6 is the concluding section of the deliverable. It also provides an outlook for the final release of the deliverable (i.e. for D3.14).

2. Packaging & Integration of QU4LITY Digital Enablers

2.1 QU4LITY Enablers Integration Concept and Elements of the Integration Solutions

2.1.1 Overview

In this section we will provide an overview of popular tools and platforms which facilitated the management, packaging, development and distribution of software components. In the next session we will see that these findings have been included in an online questionnaire in order to identify the preferred ones from the technical QU4LITY partners.

2.1.2 Software Packaging

For the QU4LITY software packaging category we have considered only one candidate which is the Docker[1] images. Docker images is currently the dominant technology/methodology and is considered a de facto. A Docker image is a file, comprised of multiple layers, used to execute code in a Docker container. An image is essentially built from the instructions for a complete and executable version of an application, which relies on the host OS kernel. In the following sections (wherever relevant i.e. stack management, monitoring tools etch.) we are only considering tools that are compliant with the Docker Platform.

2.1.3 Container Tools

These tools facilitate the Docker container deployment and management. The most popular among them are listed below:

2.1.3.1 Kubernetes

Kubernetes is an open source orchestration system for Docker containers. It handles scheduling onto nodes in a compute cluster and actively manages workloads to ensure that their state matches the users declared intentions.

Kubernetes[2] presents the following advantages for developers: (i) It is a leading docker container management solution; (ii) It is a simple and powerful solution; (iii) It is an open source solution, yet backed by a giant vendor like Google; (iv) It has the right abstractions; (v) It provides services scalability and (vi) It offers a replication controller.

2.1.3.2 Docker Compose

With Compose, you define a multi-container application in a single file, then spin your application up in a single command which does everything that needs to be done to get it running.

[1] https://docs.docker.com/
[2] https://stackshare.io/stackups/docker-compose-vs-docker-swarm-vs-kubernetes

The popularity of Docker Compose[3] stems from its: (i) Multi-container descriptor; (ii) Fast development environment setup; (iii) Ability to easily link containers; (iv) Offering of a simple yaml configuration; (v) Easy setup; (vi) Support for Yml or yaml format; (vii) It support for Standard Docker API and (ix) its open source.

2.1.3.3 Docker Swarm

Swarm serves the standard Docker API, so any tool which already communicates with a Docker daemon can use Swarm to transparently scale to multiple hosts: Dokku, Compose, Krane, Deis, DockerUI, Shipyard, Drone, Jenkins... and, of course, the Docker client itself.

Highlights on why developers choose Docker Swarm[4]: (i) Docker friendly; (ii) Easy to setup; (iii) Standard Docker API; (iv) Easy to use; (v) Native; (vi) Free; (vii) Clustering made easy; (viii) Simple usage; (ix) Integral part of docker.

2.1.4 Management & Monitoring Tools

2.1.4.1 Rancher

Rancher[5] is an open-source multi-cluster orchestration platform, lets operations teams deploy, manage and secure enterprise Kubernetes. Rancher is a complete software stack for teams adopting containers. It addresses the operational and security challenges of managing multiple Kubernetes clusters, while providing DevOps teams with integrated tools for running containerized workloads.

2.1.4.2 Sumo Logic

Sumo Logic[6] is a cloud-native, log review tool that provides advanced analysis, visualization, and alerting options. The metrics monitoring solution provides real-time security and operational information, and allows you to diagnose and troubleshoot all application and infrastructure problems. Machine learning analytics also means the quick discovery and future prediction of threats and anomalies before they can become an issue and affect end-users.[7]

Cost: Free Up to 500MB/day

Professional: Logs & Metrics: $90/month, 1GB/day

Enterprise: Logs & Metrics: $150/month, 1GB/day

2.1.4.3 Portainer

Portainer[8] is an open-source lightweight management user interface for Docker environments. Portainer works on top of the Docker API and provides a detailed

---

[3] https://stackshare.io/stackups/docker-compose-vs-docker-swarm-vs-kubernetes
[4] https://stackshare.io/stackups/docker-compose-vs-docker-swarm-vs-kubernetes
[5] https://rancher.com/
[6] https://www.sumologic.com/
[7] https://dzone.com/articles/50-useful-docker-tools
[8] https://portainer.io/

overview of Docker. Capabilities include the ability to manage containers, images, networks, and volumes.[9]

Cost: Free

2.1.4.4 Datadog

Datadog[10] is a SaaS-based data analytics platform for large-scale cloud environments that generates and collects metrics/data events from servers, databases, and applications. The full-stack monitoring service provides support for Docker, Kubernetes, and Mesos.[11]

Cost: Free up to 5 hosts

Pro: $15/host/month

Enterprise: $23/host/month

2.1.4.5 Prometheus

Prometheus[12], which is developed by SoundCloud, is an open source system-monitoring and alerting toolkit. It incorporates many aspects of monitoring such as metric generation and collection, results visualization, and alerting capabilities for when anomalies occur. Prometheus excels at recording numeric time series and complements both machine-centric monitoring as well as highly dynamic service-oriented architectures.[13]

Cost: Free

2.1.5 Code Management

2.1.5.1 GitHub

GitHub is a web-based DevOps lifecycle tool that provides a Git-repository manager providing wiki, issue-tracking and CI/CD pipeline features. Public or Private[14] schemes available and offered as a service.

2.1.5.2 GitLab

GitLab is a web-based DevOps lifecycle tool that provides a Git-repository manager providing wiki, issue-tracking and CI/CD pipeline features. Public or Private[15] schemes available and offered as a service.

[9] https://dzone.com/articles/50-useful-docker-tools
[10] https://www.datadoghq.com/
[11] https://dzone.com/articles/50-useful-docker-tools
[12] https://prometheus.io/
[13] https://dzone.com/articles/50-useful-docker-tools
[14] https://github.com/pricing
[15] https://about.gitlab.com/handbook/ceo/pricing/

2.1.6 Repository Management

2.1.6.1 Nexus Repository Manager

Nexus Repository Manager OSS[16] repository manager to administrate artefacts (mainly Docker images). Public or Private[17] schemes are offered as a service.

2.1.6.2 Docker Hub

Docker Hub[18]: With Docker Hub you can create, manage, and deliver your teams' container applications. Public or Private[19] and offered as a service.

2.1.6.3 JFrog

JFrog[20] is a commercial binary repository manager software designed to store the binary output of the build process for use in distribution and deployment. Artifactory provides support for various package formats such as Maven, Conan, Debian, NPM, Helm, Ruby, Python, and Docker.

2.2 QU4LITY Integration Requirements

In this section we provide the online survey[21] results which identified the QU4LITY technology provider's preferences. The survey, along with following up discussions in tele-conferences led to the selection of the integration tools and platforms to be used for the QU4LITY integration and packaging solution. Specifically, based on the survey the most preferred options regarding tools were:

- Software Packaging: Docker images.
- Container Tool: Docker Compose.
- Management & Monitoring Tool: Portainer.
- Code Management: GitHub.
- Repository Management: Docker Hub.

The survey's questions and answers/results follow:

1. Which is your preferred Container Tool?

The preferences are depicted in Figure 2 and Table 1.

[16] https://help.sonatype.com/repomanager3
[17] https://www.sonatype.com/product-pricing
[18] https://hub.docker.com/
[19] https://hub.docker.com/pricing
[20] https://jfrog.com/
[21] https://www.surveymonkey.com/r/653YBZH

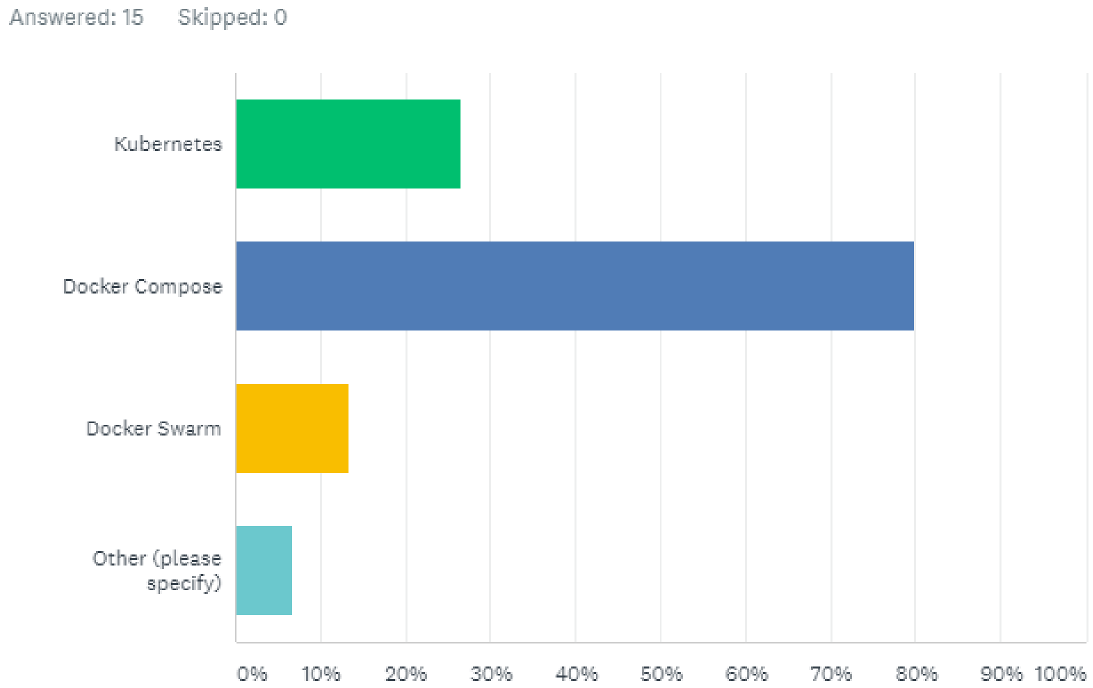

Figure 2: Stack Management environment Preferences

ANSWER CHOICES	RESPONSES(%&Count)	
Kubernetes	26.67%	4
Docker Compose	80.00%	12
Docker Swarm	13.33%	2
Not applicable	6.67%	1

Table 1: Stack Management environment Preferences

2. What is your preferred Docker Management & Monitoring Tool?

The preferences are depicted in Figure 3 and Table 2.

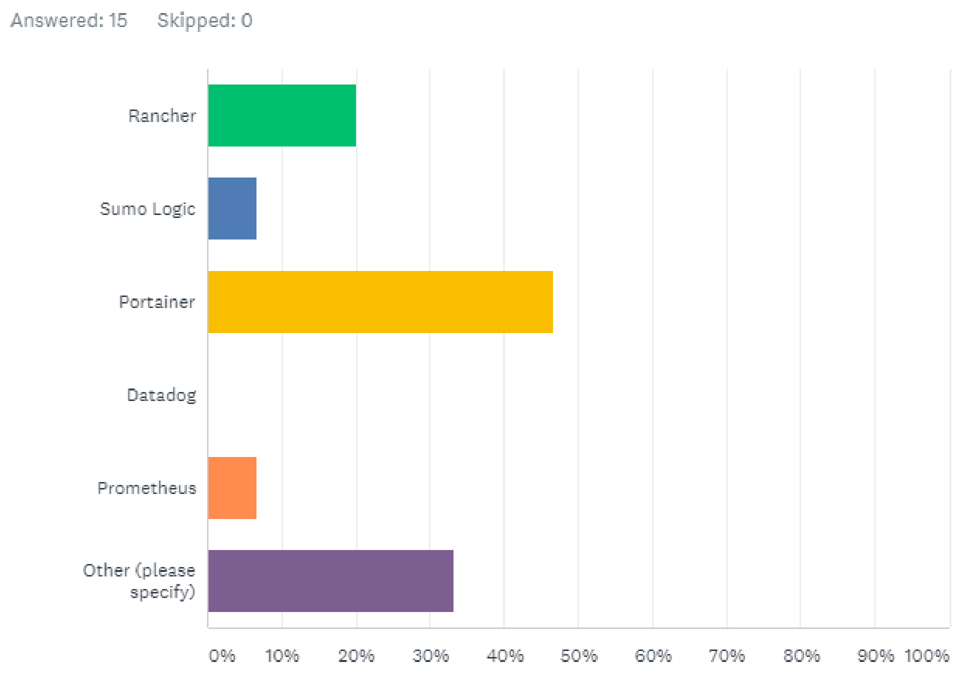

Figure 3: Docker Stack Management/Monitoring Tool Preferences

ANSWER CHOICES	RESPONSES(%&Count)	
Rancher	20.00%	3
Sumo Logic	6.67%	1
Portainer	46.67%	7
Datadog	0.00%	0
Prometheus	6.67%	1
Other: Docker API(1) Docker Command Line(1) No Preference (3)	6.67% 6.67% 20.00%	1 1 3

Table 2: Docker Stack Management/Monitoring Tool Preferences

3. What is your preferred Code Management tool

The preferences are depicted in Figure 4 and Table 3.

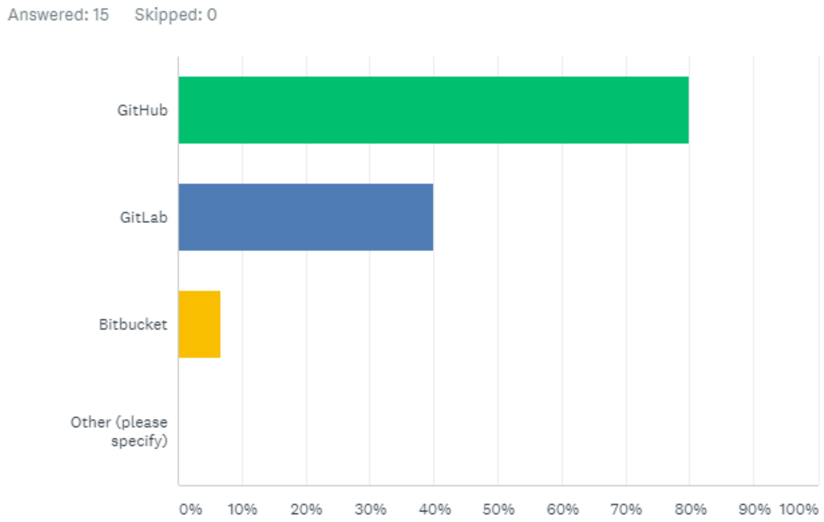

Figure 4: Code Management tool Preferences

ANSWER CHOICES	RESPONSES(%&Count)	
GitHub	80.00%	12
GitLab	40.00%	6
Bitbucket	6.67%	1

Table 3: Code Management tool Preferences

4. Which is your preferred Repository Management service?

The preferences are depicted in Figure 5 and Table 4.

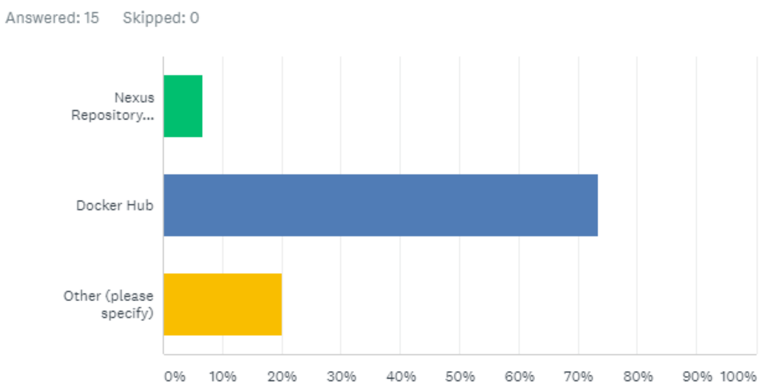

Figure 5: Repository Management service Preferences

ANSWER CHOICES	RESPONSES(%&Count)	
Nexus Repository Manager OSS	6.67%	1
Docker Hub	73.33%	11
Other: JFrog (1)	6.67%	1
GitLab(1)	6.67%	1
No Preference (1)	6.67%	1

Table 4: Repository Management service

2.3 QU4LITY Integration Infrastructure

In this section we provide the tools and platforms which will be used in QU4LITY, based on the technical partners preferences above, in order to provide the packaging and integration of the offered combinations of digital enablers in the scope of ZDM scenarios.

2.3.1 Software Packaging with Docker images.

As mentioned above Docker containers will be used to package and distribute the different Digital Enablers offered in WP3. 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 offers shipping, testing, and deploying methodologies easily and quickly, where time between writing code and running it in production can be significantly reduced.

Docker provides the ability to package and run an application in a loosely isolated environment called a container. The isolation and security allow you to run many containers simultaneously on a given host. Containers are lightweight because they don't need the extra load of a hypervisor but run directly within the host machine's kernel. This means you can run more containers on a given hardware combination than if you were using virtual machines. You can even run Docker containers within host machines that are actual virtual machines [Docker].

There are many tutorials in order to containerize an application or a system and offer it thru a repository management service (see section 0) which span from beginners to more advanced ones depending on the technologies used. An intermediate one which doesn't focus in a specific technology and provides the relevant aspects that are necessary to establish a well-defined contract between Dev and Ops teams can be found in [Souza18], which provides an article on how to "dockerize" any application. The article provides a 10-steps checklist which includes the:

- Choice of a base Image.
- Installation of the necessary packages.
- Addition of custom files.

- Definition of users that will run your container.
- Definition of the exposed ports.
- Definition of the entry point.
- Definition of the configuration method.
- Externalization of the data.
- Logs handling.
- Logs rotation and other append only files.

2.3.2 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.

Information on how to edit a docker-compose.yml file can be found at Docker Docs [Docker] and more specifically at the Get started with Docker Compose[22]

2.3.3 Management/Monitoring with Portainer.

As mentioned above the preferred option for container management and monitoring is the usage of the Community Edition (CE) of Portainer. Portainer CE[23] is a lightweight management toolset that allows you to easily build, manage and maintain Docker environments. Portainer offers a GUI (Graphical User Interface) which alleviates the complexity of using CLI (Command Line Input) commands. Portainer CE minimizes the learning curve of using and managing docker environments. Portainer offers the following features:

- UI that covers all of essential docker CLI actions
- Enhanced functions, not available from the command line
- Expert configuration built into the software
  - Including pre-validation checks for complex deployments
- Management of access control and LDAP authentication
- Aggregation view of swarm clusters
- Log viewer
- Remote console with process performance viewer

Directions on how the technology providers can install Portainer environment in a local Docker instance can be found at the Portainer's Deployment[24] documentation.

[22] https://docs.docker.com/compose/gettingstarted/
[23] https://www.portainer.io/products-services/portainer-community-edition/
[24] https://portainer.readthedocs.io/en/stable/deployment.html

General documentation along with user and configuration guides can be found in Portainer's Documentation[25]

2.3.4 Code Management with GitHub.

The Open Source (OS) Digital Enablers source code can be managed in a Git repository and more specifically the GitHub[26] since it was the preferred option. A QU4LITY organization has been set up at GitHub (**https://github.com/qu4lity**) where every OS Digital Enabler which is developed within the context of QU4LITY project can have its own repository.

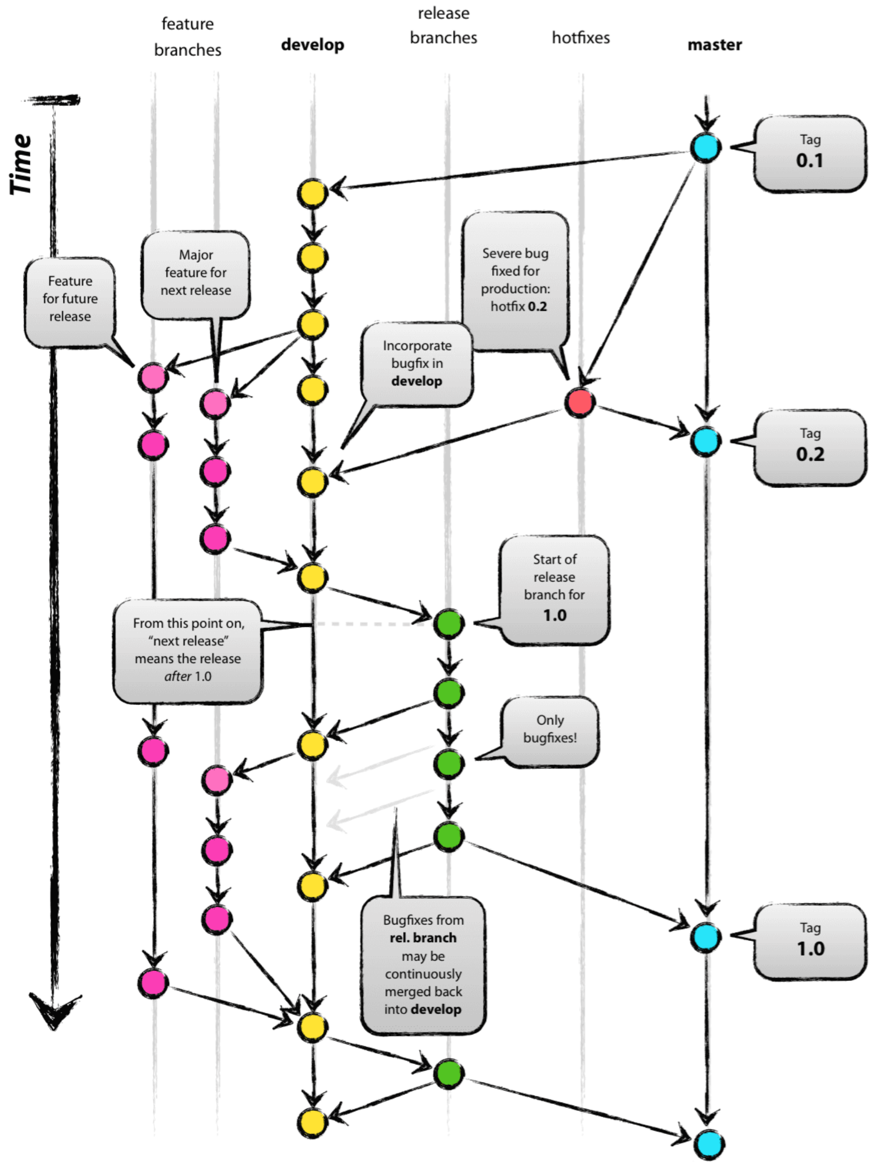

Figure 6 A Complete Git branching model[27]

[25] https://portainer.readthedocs.io/en/stable/#
[26] https://github.com/
[27] https://nvie.com/posts/a-successful-git-branching-model/

The Digital Enablers owner that wish to create dedicated repositories for their Assets may contact Mr. Nikos Kefalakis (Nikos.KEFALAKIS@intrasoft-intl.com) 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.

Guidelines on how to use GitHub can be found in the GitHub Guides[28] and more specifically for a beginner the Hello World guide[29] could be used.

QU4LITY could use the branching model (or part of it) proposed by Mr. Vincent Driessen[30] "A successful Git branching model"[31] and a complete version of which is shown in Figure 6.

In cases where existing components (available in other GitHub branches) are used, repository mirroring mechanisms will be employed to ensure access to the components from the project's GitHub.

2.3.5 Repository Management with Docker Hub.

The Open Source (OS) Digital Enablers containers can be hosted in Docker Hub[32] since it was the preferred option. A QU4LITY organization has been set up at Docker Hub (**https://hub.docker.com/orgs/qu4lity**) where every OS Digital Enabler which is developed within the context of QU4LITY project can have its own repository.

The Digital Enablers owner that wish to create dedicated repositories for their Assets may contact Mr. Nikos Kefalakis (Nikos.KEFALAKIS@intrasoft-intl.com) with their Docker Hub 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) or Authors(level Read-Write) depending on the team organization.

2.4 QU4LITY Deployment Infrastructure for Edge Computing

2.4.1 Concept of Edge Computing

In traditional IoT architectures, data are collected from geographically dispersed sensors and transported to a central repository (e.g. Cloud) where it is combined and processed collectively. Increasing efficiency, scalability and performance in everyday tasks, integration of cloud computing with Internet of Thigs enables enterprises to make better business decision faster and respond to changing market conditions in

[28] https://guides.github.com/
[29] https://guides.github.com/activities/hello-world/
[30] http://nvie.com/about/
[31] http://nvie.com/posts/a-successful-git-branching-model/
[32] https://hub.docker.com/

real time. Although cloud computing has enabled the processing of huge amount of data, it is not an ideal solution for many applications. Massive amount of data needs to be transferred from the edge of the network into the cloud and backwards, thereby straining the bandwidth and latency of the communication. To tackle these limitations, edge computing has been introduced that will exist parallel to cloud computing.

Edge computing is viewed as a decentralized computing infrastructure, in which computing resources and application services can be distributed along the communication path from the data source to the cloud. Computational requirements can be fulfilled "at the edge", where the data is collected or where the certain actions are being performed (as depicted in Figure 7). Low latency and faster real-time analysis of data coming from machines located at the edge can be of huge advantages for reacting faster to errors in the production process, thereby being able to improve the quality of the end product.

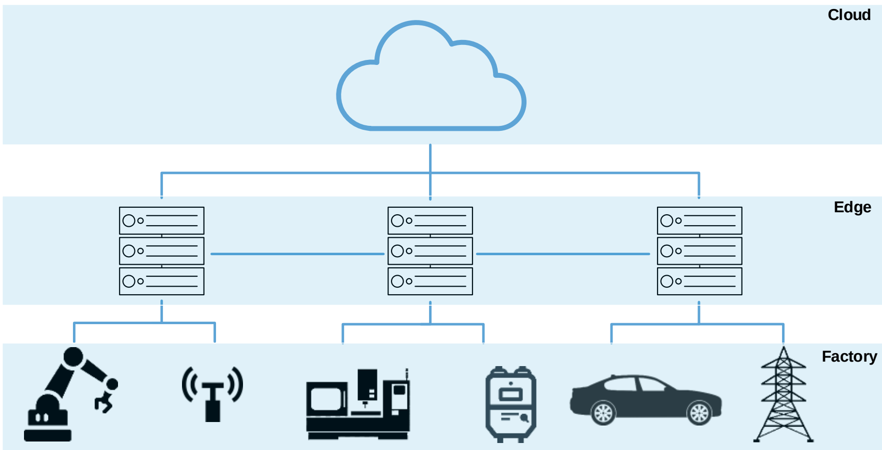

Figure 7: The Edge Computing Paradigm Illustrated

Edge computing offers, compared to traditional cloud computing many benefits, which are described below:

- **Improved performance:** The edge is not only a location for data collection and transmission to the cloud. The edge also is preforming processes, analysis of data and acts on the collected data within millisecond and is therefore crucial for optimizing industrial data, especially when targeting real-time (autonomous) quality control.
- **Compliance, data privacy and data security:** Edge computing allows companies to perform data collection and operations on these data on side using a public/private cloud by using local computing resources. These local computing resources can be in the specific area, region or even on-site at the actual premises, thereby offering the required local security boundaries.
- **Reduction of operational costs:** By placing computational processes at the edge of the network, a continuum from the device to the cloud is created that handles the massive amount of data that is generated. By processing the data closer to where it is generated, it decreases the response time to events (by

removing the round trip to the cloud for analysis) and it avoids costly bandwidth (as there is no need for transmission of gigabytes of data). Additionally, data is analysed locally, reducing the costs of protection of sensitive production data.

2.4.2 Virtualization (Virtual Machines)

Virtualization is a well-known concept in the general-purpose computing industry for more than a decade. It introduces an abstraction layer, such that applications (and complete operating systems) execute on virtual machines (VM) (or Virtual Containers), instead of their direct execution on the underlying hardware.

There are plenty of benefits of virtualization in general-purpose computing and many translate easily to benefits also in embedded systems. For example, (i) virtualization supports multiple operating systems. Rich operating systems like Linux can thus coexist with real-time operating systems or Windows operating systems on the same hardware platform. Additionally, (ii) services that used to run on dedicated hardware can be consolidated into a single hardware platform. This transition from federated to integrated architectures has been leveraged by other industries like the avionics industry for many years. Finally, (iii) virtualization can also be a means to maintain security properties. A security breach in one VM remains isolated and does not affect applications that are hosted in other VMs. Figure 8 depicts a schematic overview of how a system with virtualization is structured compared to a classical system without the concept of virtualization. Depending on the specification of the device (e.g. number of cores, processing powers), the number of virtual machines can vary.

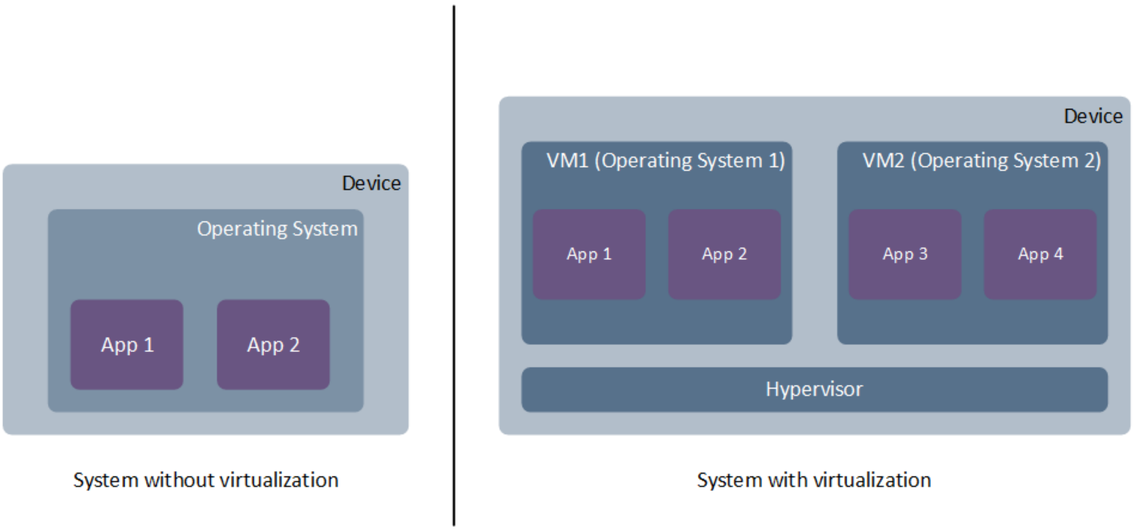

Figure 8: Classical System Architecture vs. Virtualization

Two types of virtualization are commonly identified. Type 1 Hypervisors (also called native of bare-metal hypervisors) implement a software layer that is directly executed on top of the underlying hardware. Type 2 Hypervisors (also called hosted hypervisors) form a software layer that is positioned on top of an operating system. A type 2 hypervisor has the same functionalities as the type 1, but the positioning within the edge architecture is different. The right side of Figure 8 depicts a Type 1 hypervisor.

Virtualization techniques allow a single hardware to run multiple VMs in parallel. For example, it is possible to run Windows (and even multiple instances of it) in parallel with rich operating systems like Embedded Linux. The hypervisor, as the core element of virtualization, guarantees bounded interference or even interference-freedom from one virtual machine on the respective others. Provided the right system architecture is in place, it can be argued that safety-related applications can be even co-located with other, non-critical or lesser critical applications on the same hardware and furthermore connected via a e.g. deterministic Ethernet to even other controlling hardware.

2.4.3 Deployment of Dockerized workloads

The deployment of workloads is dependent on the management system that is being applied within your environment, so it is very difficult to provide a generalized description of how deployment is being handled. In this section, the deployment of workloads is described as it is being done with the Nerve Blue solution from partner TTTech Industrial Automation AG (TIAG).

A task or service running within an edge device, which is not part of the edge device self is defined as a workload. Typically, a workload is installed by a user in order to achieve a certain task. These workloads will execute a specific task at the edge device, which can vary from being a firewall, an analytics application, an actuation of a part of a machine, visualization, etc. Workloads can be created as virtual machines or containers, that are executed in isolation from one another. Workload consolidation reduces costs and increases operational efficiency by removing hardware from the factory floor and integrating management functions like remote update, monitoring and centralized logging (see Section 2.4.4). Within the QU4LITY project, many of the tasks being hosted at an edge device will perform analytics based on data coming from the manufacturing machine to improve the quality of the production process. If an autonomous quality process is required, a feedback loop needs to be implemented on the edge device that will also perform an actuation at the machine, based on the analytics performed. As this needs to be done in real-time (during a running manufacturing process), it is important that the analytics is performed as close as possible to the machine, thereby being able to react to changes as fast as possible.

As mentioned in the previous subsection, edge devices support virtualization, thus providing virtual machines with different operating systems. To ease the deployment of workloads to an edge device, workloads are dockerized.

Figure 9 provides an overview of a management system (in this case a graphical depiction of the Nerve Blue Management System). It supports a workload repository, where users (e.g. OEMs, machine builders) can provide updates to workloads that are currently running on machines and can be uploaded to the machines. The Management System should also provide a graphical user interface, providing the factory or machine operator an overview of all the connected machines and the workloads deployed to these different machines. The management of workloads with the aid of a user interface will be discussed in more detail in section 2.4.4. Finally, to deploy the workloads to the connected edge devices, different communication protocols need to be supported, as not all devices and machines support the same protocol.

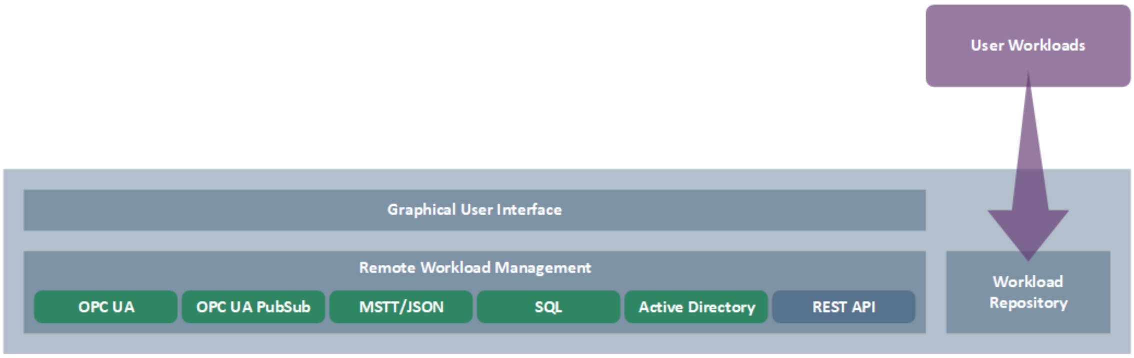

Figure 9: Management System

For the Nerve Blue there are three types of workloads that can be deployed, namely CODESYS workloads, virtual machines workloads and Docker workloads (see Figure 10). These workloads are developed offline by software developers and made available by e.g. machine builders or system integrators. By uploading them to a repository (as a dockerized solution), the machine or line operator can push the newly provided workloads to one or multiple edge devices simultaneously.

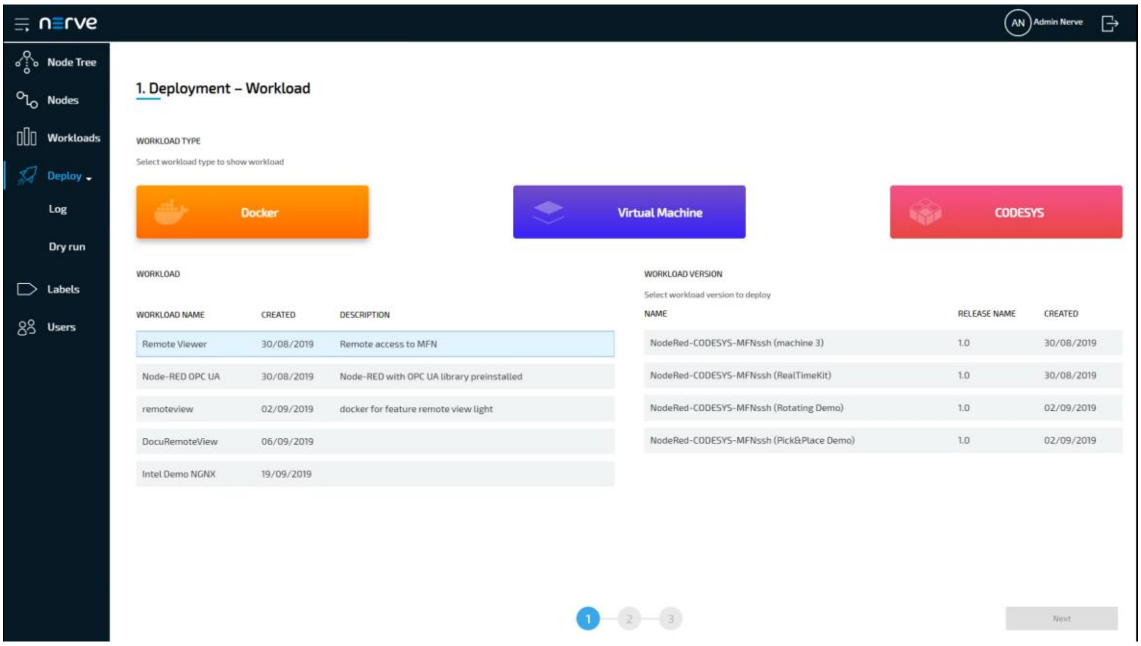

Figure 10: TTTech Nerve Blue – Deployment of three different types of workloads

2.4.4 Management of workloads

The management system is a web-based service that permits the management of edge nodes that are registered to the specific management system. It can be used to:

- Monitor edge devices
- Deploy and control workloads on an edge device
- Manage workloads

The management system can either be hosted locally or in the cloud. The management system provides access to all edge devices located around the work, enabling software configuration and remote diagnostics. Additionally, it functions as a central repository for workloads, which can be remotely deployed as Docker containers, VMs or CODESYS® applications to one more device that are registered and connected to the management device, as is graphically represented in Figure 11.

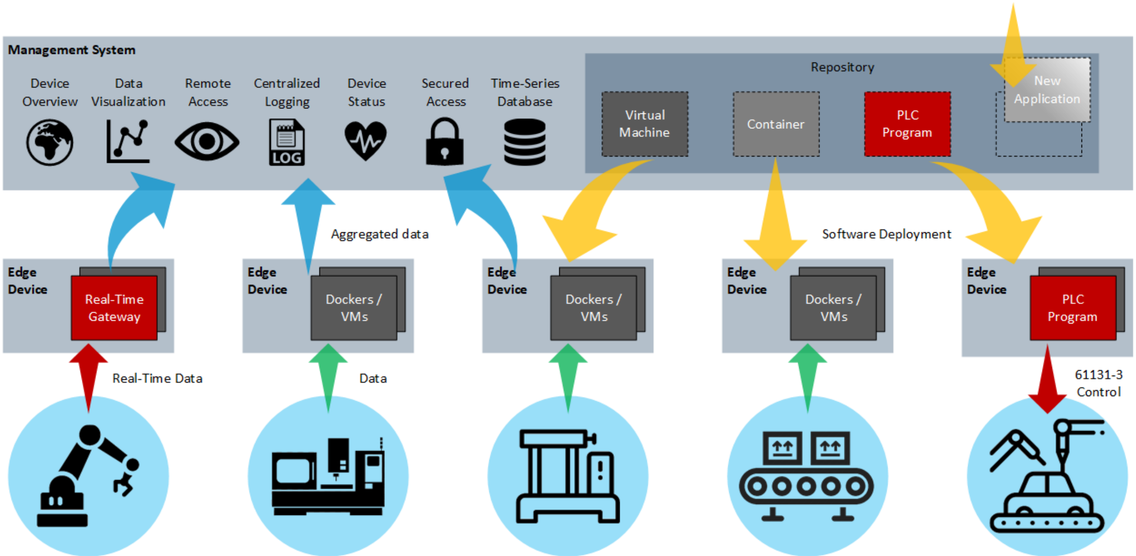

Figure 11: Management of workloads

The management system enables users to deploy workloads on connected edge devices, but also to manage these workloads. This can be interpreted as providing updates or solutions to errors. The management system will offer the possibility to either deploy a workload to a single connected device, or to multiple devices at the same time. Finally, the management system provides information from the connected devices back to the human operator, dependent on the available functionalities within the management system. This can be graphical representation of the connected machines (e.g. processing time, online time, etc.) and graphical representation of the connected edge devices. Examples of this are represented in Figure 12, where screenshots of the Nerve Blue management system are depicted.

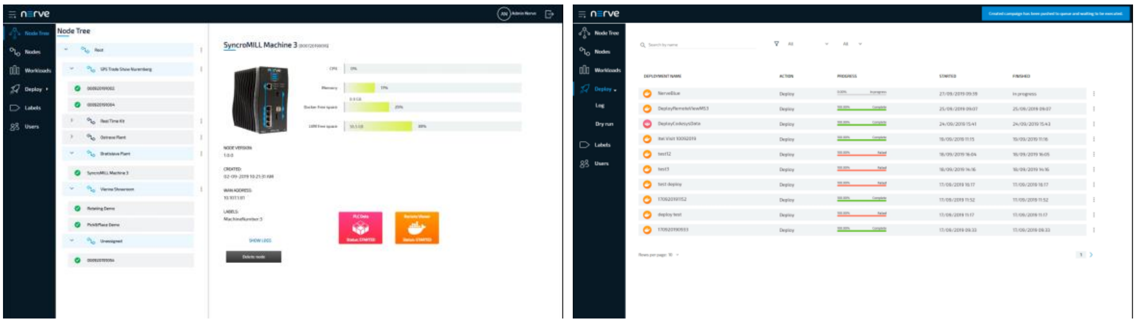

Figure 12: Nerve Blue – Management of edge devices and workloads

2.4.5 Use of the Edge Computing Deployment Infrastructure in QU4LITY

Within the QU4LITY project and the task related to edge/fog computing (i.e. Task 3.4), an overview of the edge technologies developed is provided. Most of the partners within this task are also involved in an explicit industrial pilot, where the developed or extended edge/fog technologies will be integrated, tested and validated. Additionally, they cooperate with other partners inside the consortium that will deploy their quality applications on one of the specific devices.

TTTech, as leader of Task 3.4 and provider of the Nerve Blue solution, is involved in the Philips pilot (Philips OneBlade shaving unit production line). Philips is currently developing new production lines with increased capacity to meet the increased market demand. Philips is targeting the following three metrics within the project: Time to Market, Production costs and Product/component quality. Nerve Blue will be deployed in the production line and will start collecting production and machine data from specific sections within the line, where the edge device is connected. These data will be made available for the other partners that are modifying the raw data and evaluating the processed data. Within the first approach, these quality applications will provide feedback to the human operator of the machine or production line, informing where errors occur or where potential optimization can take place. The final goal is that the applications identify solutions and perform autonomous quality actions, thereby optimizing the production process without any human interference or downtime of the production.

Pilot partners TNO, Fraunhofer and Sintef will develop individual quality processing applications, varying from data exchange and autonomous data management, image processing and shift-in torque chain quality control. These applications will be created as Docker applications and will be made available through the Nerve Management System, from which they can be deployed to any of the registered edge devices. The deployment of the applications can be triggered by the person responsible for the operation of the plant. The dockerized applications are pushed into the Nerve Management System by the application developers or the machine builders that provide either new or updated functionalities. Within the Nerve Management System, the uploaded docker applications are stored as images on the cloud and can be deployed from there to the selected edge device.

After deployment. the applications are hosted on the designated edge devices, either in a separate virtual machine or directly on the host OS within the edge device.

The Savvy Smart Box from Danobat infrastructure will be deployed, tested and validated at the prototypes' laboratory of DANOBAT, where it will be monitoring an LG machine. The quality applications will be either hosted on the Savvy Box itself, or on the connected cloud (which could be a local cloud or an external one). The Savvy Box addresses both sensing and acting capabilities implementing control functions.

The edge/fog solution from Unparallel, which is called FOOTPRINT, is currently under development and no explicit pilot or experimental facility has been identified yet where it will be deployed and validated.

2.5 QU4LITY Deployment Infrastructure for DLT Services

In this section we discuss a special class of QU4LITY digital enablers: DLT Services. The project task that is responsible for the design and development of DLT Services is T3.6 ("Blockchains for Secure Decentralized State Management in ZDM") and its the results will be released as part of the D3.12 demonstrator, due by M27 (March 2021). However, the packaging and distribution of DLT Services require a unique approach, due to the nature of the software artefacts at play. These problems are our focus in this context.

2.5.1 Concept of DLT Services in QU4LITY

Deliverable D3.11 "Permissioned Blockchain for ZDM", released in January 2020, provides a preliminary, high-level overview of Distributed Ledger Technologies and of their use in QU4LITY to enable secure state sharing and synchronization of distributed industrial processes involved in AQ/ZDM. For instance, the report describes how Blockchain *smart contracts* can improve agreements management between manufacturers, customers and other stakeholders in a supply chain. Many other potential use cases exist, though. From the perspective of AQ/ZDM applications, there are two important categories of functionality that smart contracts can enable:

- **Decentralization of control**: processes with multiple autonomous actors can be run safely without the supervision of an authority. Smart contracts achieve this objective by replicating their business logic and persistent memory on multiple nodes. Nodes must reach a collective agreement on each single change of state (a "transaction" proposed by a client) before it is accepted by the system as a whole. All communication between nodes go through *peer-to-peer* network protocols, so that no central service is needed to enforce rules or even coordinate the process.

- **Trustworthy tracking of events**: despite the lack of a central repository, everything written on a Blockchain by a smart contract cannon be altered or removed without all participants (nodes and clients) noticing, thanks to the cryptographic techniques used to *seal* both individual transactions and their sequential order. Adding digital identities and signatures to the picture results into a system of record that enforces *nonrepudiation*[33].

In deliverable D3.11, a DLT Service is described as a smart contract plus the client software used by applications to connect and interact with it. So, DLT Services are software components composed by two parts – the service and its client – that are executed in different environments. Although their development is done in task T3.6, in the context of task T3.7 they are the interoperable *digital enablers* that must be packaged, distributed and integrated for pilot sites and validation facilities to use. A prerequisite for this is the availability of a DLT Infrastructure (see §2.5.2).

It is worth noting that, in most cases, the logic and the data model of a smart contract are application-specific. There is a very limited number of AQ/ZDM use cases that may be effectively supported by a totally generic DLT Service. This is the reason why only a few DLT Services are in the QU4LITY library of packaged enablers (see Section 3). These are also the digital enablers that will be validated in experimental facilities (WP6). Custom-designed DLT Services will probably also be developed on demand for the use in large scale pilots (WP7).

2.5.2 QU4LITY Common DLT Infrastructure

The "DLT (or Blockchain) Infrastructure", often mentioned in QU4LITY's DoA and deliverables, is the runtime environment that hosts the smart contract side of DLT Services. In the original QU4LITY Reference Architecture, such infrastructure was referred to as the Value Chain Ledger (VCL) and Private Ledgers (PL): the former is a single common facility used by multiple organizations that belong to the same business ecosystem, while the latter term refers to any private instance deployed internally to a specific organization, with no information shared with the outside world. In D3.11, it was discussed how Hyperledger Fabric (HLF), a mature and performant open source Blockchain system licensed by the Linux Foundation, could be used in both VCL and PL roles. In the specific context of the QU4LITY project, HLF's capability of managing multiple private *channels*[34] on a single physical instance creates the possibility to run all DLT Services used by the QU4LITY pilots on one central facility – a model that is often named "Blockchain-as-a-Service" (BaaS), because all the nodes of the Blockchain system are hosted on the Cloud and operated by one single service provider. With this architecture we don't exploit at all the decentralization concept that is at the heart of the DLT paradigm, but on the other hand we gain a significant advantage in terms of accessibility, as pilot sites do not need to set up their own HLF *peer node(s)* in order to run their experimentation. As the BaaS model does not have any negative impact on validation (decentralization is

[33] The term "nonrepudiation" refers to the capability of ensuring that the author of a communication cannot deny its authenticity and provenance.
[34] A channel, in HLF's jargon, is the equivalent of a tenant in a multi-tenant system: an isolated logical partition of a larger system where only a subset of users has access.

not actually achieved, but is effectively simulated), in QU4LITY we decided to adopt it to implement the project's common DLT Infrastructure.

The task responsible for the definition, creation and operation of QU4LITY's common DLT Infrastructure is T3.6. Concretely, the DLT Infrastructure will be provided by Engineering Group, as part of their ongoing **Hyperlab** initiative (see Figure 13). Hyperlab will give access over the public Internet to all those QU4LITY partners that are willing to integrate, for testing purpose within the project, their AQ/ZDM applications with the DLT Services published in the library (see Section 3) or even with their internally-developed smart contracts. Every QU4LITY use case will be allocated – on request – a dedicated private channel on Hyperlab's HLF instance, so that each experimentation can proceed in total privacy and without any conflict with the others. This does not prevent those partners who wish to operate autonomously their own private network to do so, by deploying the standard HLF platform available on the Hyperledger community site[35]: all DLT Services in the catalog are fully compatible with HLF v2.0.

Figure 13 -Engineering's Hyperlab leaflet

2.5.3 Packaging and Distribution of QU4LITY DLT Services

Smart contracts are different from most kinds of software in that they are deployed on their hosting environment by means of a Blockchain transaction. On the other hand, the usefulness of client software is best when provided as a code module that developers can embed into their own application – although more complex models based on local services that expose a REST API are also a viable option. Without going into detail over this, the bottom line is that DLT Services are not suitable for distribution under the form of Docker images, as done for most digital enablers in

[35] See https://hyperledger-fabric.readthedocs.io/en/latest/build_network.html

the QU4LITY library. The hosting environment itself – i.e. the HLF nodes of the Blockchain infrastructure – is indeed packaged as several Docker images, and its deployment is automated by scripts. This clearly means that whoever is willing to set up their private DLT Infrastructure – as discussed in the previous section – will be able to do so in a similar way as a "regular" digital enabler in the QU4LITY library. However, the DLT Infrastructure is not considered as a digital enabler and so it's not included in the QU4LITY library.

Practically speaking, each DLT Service in the library consists in the following items:

- **The smart code implementation**, which is an HLF *chaincode* program[36] written in the Java language, using HLF's Java SDK. The chaincode exposes an access-controlled API that is meant to be invoked by client software[37]. Such API is *not* documented, because applications will never need to invoke it directly: instead, they will use the client library that is provided as part of the "package" (see next point). On request from any legitimate QU4LITY user, a dedicated instance of the smart contract will be deployed by Hyperlab admins on a private HLF channel.
- **The client implementation**, which is a Java library packaged as a single JAR file. The library exposes an in-process API to applications, which is a higher-level abstraction of the corresponding smart contract API. Applications will invoke the former (local) as a proxy of the latter (remote). All protocol-specific details, including those related to security, are hidden from the application. The JAR file also contains the full API documentation in JavaDoc format, also available online for convenience.
- **The general documentation of the DLT Service**. This is not the same as the API documentation, because it explains the use case workflow and goes into details of the data model – i.e. the structure of data that applications exchange with the smart contract.

2.5.4 Integrating QU4LITY DLT Services

Integration of a user application with a DLT Service is done by merging the client library, provided as part of the package, into the code of the application. It is worth noting that the choice of the Java language for the development of the client library (see previous section) is a constraint: although it is theoretically possible to call Java libraries from "foreign" environments[38], the most straightforward way to do it is that both the caller and the called run within the same JVM. The minimum supported version of the Java environment is v1.8.

As explained above, every DLT Service comes with generic documentation and API technical specifications. In the documentation, developers will find the specification of the workflow and of the data structures used in it. Such data structures are

[36] See https://hyperledger-fabric.readthedocs.io/en/release-2.0/chaincode.html
[37] Access control is not managed by the chaincode, but by the HLF platform on which policy are configured on a per-endpoint basis.
[38] For an example, see https://www.graalvm.org/

implemented in the client library as Java classes, so that application coders don't need to deal with text-based data representations like XML or JSON.

For the application to work, it must be connected to the smart contract side of the DLT Service. To do this, two things are required: the correct configuration of network parameters and valid credentials for authentication / authorization on the DLT Infrastructure. The former are pointers to a specific chaincode instance deployed on a given channel of a given HLF network; the latter is a digital certificate, released by Hyperlab's *certificate authority*, that identifies a user and, optionally, assigns some attributes (e.g. organization and role). Both are provided by Hyperlab admins on request by the developer / user, packaged into a single ZIP file that must be copied to the local filesystem of the device hosting the application. Clearly, credentials are "personal" for every client, so the registration and configuration must be repeated for each of them, even if they are different instances of the same application.

Once everything is set up, using DLT Services through its local client is not much different than making in-process calls. One thing that application developers need to know is that these calls, despite being apparently local, are triggering a transaction on a distributed system. Although smart contract transactions are asynchronous, the client will simulate a synchronous call by blocking the caller until the remote transaction is confirmed. This means increased latency: the time after which an API method call *returns* will be typically some orders of magnitude longer than an equivalent local call. Mileage may vary, so *developers are encouraged to test first that the latency they experience is well-tolerated by their AQ/ZDM process*.

2.5.5 Example Use Case

For this example, we focus on the Quality Clearing House (QCH), which is part of the QU4LITY library of DLT Services (see Section 3). QCH enables a decentralized workflow for quality management in supply chain scenarios. The supply chain processes supported by QCH follow a simple pattern, the workflow of which is described below. To exemplify the pattern and for the sake of simplicity, we have identified distinct "actors" playing the three roles embodied in the system (Quality Master, Provider, Quality Assessor); however, in real-world supply chain processes it is likely that multiple organizations will play the provider role, and/or that one single organization will play the remaining ones.

Records

All records have their own unique identifier, which is used internally for cross-reference, and are owned by the party that creates them.

- Quality Assessment Data Model (QADM): a structured digital document that defines the standard of quality that applies to a given material, part or product, as stipulated by a commercial agreement (which is out of this scope). The standard is expressed in terms of a list of measurements, each consisting of a qualitative definition[39] and a quantitative range. One QADM document may exist for the entire duration of a contract, or new versions may be created

[39] The vocabulary used to identify measurements must be in common between all parties involved: the meaning of each measurement declaration, which includes not only the "what" but also the "how" and possibly the "when", must be unambiguous for everyone. To this goal, a formal ontology may be defined.

that override previous ones in order to follow along the evolution of quality requirements.

- Shipping Unit Manifest (SUM): a digital record that identifies a shipped batch of materials, parts or products as subject to a given quality agreement. It consists of a pointer to an existing QADM and of a list of IDs, each associated to a physical item in the batch.
- Quality Assessment (QA): a digital record that reports the quality measurements taken on a received batch of materials, parts or products, along the guidelines of their agreed standard. It consists of pointers to an existing QADM and SUM, plus the actual values of all the measurements taken. Depending on the quality agreement in place, measurements may be reported per-batch (average values) or per-item.

Roles

- Quality Master: typically, it's the manufacturing company that manages the supply chain. It creates the QADM document(s).
- Provider: it's always a provider of the supply chain (seller of goods). It creates SUM records.
- Quality Assessor: it is responsible of measuring the quality parameters on physical items with respect to the standard. It may be the same entity as the Quality Master or a different one – i.e., a third party in charge of independent assessment. In the latter case, it should be trusted by all the involved parties. The Quality Assessor creates QA reports.

Actors in the example

- Factory A: plays the role of Quality Master.
- Factory B: plays the role of Producer.
- Company C: plays the role of Quality Assessor.

Example workflow

1. When a commercial agreement is first defined, Factory A defines the quality standard and creates a new QADM, which is published on the QCH. Factory A also sets up and configures its quality assessment process and tools in collaboration with Company C, which provides the metrology equipment that is deployed on Factory A's premises.
2. Factory B prepares a batch of goods under the aforementioned agreement. The physical items in the batch are tagged with individual IDs. When the batch is shipped, Factory B publishes a new SUM record on the QHC that points to the reference QADM and lists all the IDs contained in the shipment.
3. Factory A receives the shipment. As the individual items herein contained are unloaded, they are sent to a quality assessment facility where the equipment provided by Company C is in use. The metrology tool read the tag, identifies the item and execute the appropriate measurements.
4. When the shipment has been entirely processed, a QA report is generated by the metrology tool and published on the QCH on behalf of Company C[40].
5. The process can now iterate any number of times, starting from point #2.

[40] The tool digitally signs the QA record with a private key that matches a public key that is known to belong to Company C.

6. When payment to Factory B is due, Factory A will apply any penalties and discounts defined in the agreement for missed quality targets documented in the QCH.

2.6 QU4LITY Integration Concept

2.6.1 Mapping of Digital Enablers to QU4LITY-RA

Deliverable D2.11 has introduced the reference architecture of the project (QU4LITY-RA), which is also illustrated in Figure 14. 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.

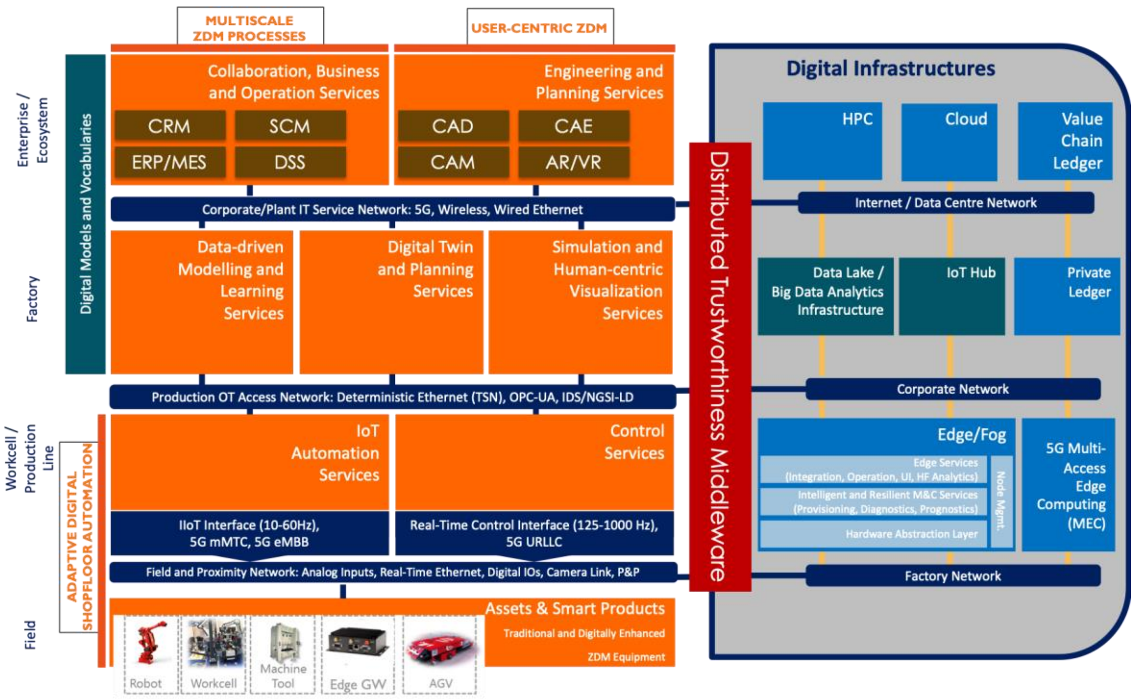

Figure 14: Snapshot of the QU4LITY Reference Architecture

Based on the outcomes of the above-listed surveys, QU4LITY will enable YAML based solutions for the configuration files of the different modules and their combination in-line with the QU4LITY-RA. YAML will be combined with tools like Docker Compose, to enable the definition and execution of multi-container containerized (i.e. Docker-based) applications. Specifically, YAML files will be used to configure the different modules and services that will comprise an industrial use case in ZDM. Accordingly, the proposed infrastructure will enable the creation and execution of all the services based on the developed configuration. Overall, the QU4LITY integration infrastructure will enable the combination/integration of different components in turnkey ZDM solutions.

The following table lists the digital enablers of the project, which are developed in other tasks of WP3 and described in other deliverables of the same workpackage. The table outlines the mapping of these enablers to the QU4LITY-RA building blocks, as well as whether they can be dockerized (i.e. whether they provide docker support).

Component Name/Description	Owner	License Type	Mapping to QU4LITY-RA	Docker Support
FAR-EDGE Distributed Data Analytics (DDA)	INTRASOFT	Open Source	Data-driven Modelling and Learning Services	Yes
Training of deep neural network models on HPC cluster	JSI	Open Source	Data Lake / Big Data Analytics Infrastructure	No
Long Short-Term Memory (LSTM) approach for RUL estimation	ATLAS	Open Source	Data-driven modelling and learning services	Yes
Decision Support System (DSS) and Strategies for ZDM	ATLAS	Open Source	Data-driven modelling and learning services	Yes
Quantitative Association Rule Mining (QARMA)	INTRASOFT	Proprietary	Data-driven modelling and learning services	Yes
Additive Simulator for processing time estimation	TTS	Proprietary	Digital Twin	No
Analytics quality trend cockpit	TTS	Proprietary	Data-driven modelling and learning services	No

Component Name/Description	Owner	License Type	Mapping to QU4LITY-RA	Docker Support
XL-SIEM	ATOS	Proprietary	IDS- Event & Logs management	Yes
QU4LITY Blockchain Infrastructure	ENG	Open Source	Value Chain Ledger & Private Ledger	Yes
QU4LITY Cloud Infrastructure	ENG	Open Source	Cloud	Yes
Converters for Interoperability	MGEP/MON	Open Source	Distributed Middleware	Yes
Open VA	VTT	Open Source	Data Lake / Big Data Analytics Infrastructure & Data-driven modelling and learning services	Yes
Image analyzer using CNN approach for (AM) surface inspection	FHG-ILT	TBD	Data-driven modelling and learning services	Yes

Table 5: List of QU4LITY Digital Enablers, Their Owners and Their Mapping to QU4LITY-RA

2.6.2 Integration Examples

Following the project's integration principles and the use of the project's containerization infrastructures, different modules from the above list will be integrated in use cases. Some sample combinations follow:

- The FAR-EDGE DDA enabler for data collection can be integrated with INTRASOFT's QARMA data mining component to enable data driven identification of defected parts and to estimate RUL. Security components like XL-SIEM can be also included in the same bundle. All components provide docker containerization support and hence their integration can be realized based on the YAML/Docker Compose approach.
- Any data mining algorithm from the list (e.g., LSTM approach for RUL, QARMA for RUL, ZDM) can be bundled with the distributed ledger / blockchain enabler to facilitate sharing of information about quality issues (e.g., presence or absence of defects, RUL information to be considered in the supply chain) with other supply chain stakeholders. As these components are all Dockerized, their integration will be based on the YAML/Docker Compose approach.

- The Digital Twin enabler (i.e. Additive Simulator for processing time estimation) will be deployed and used based on the edge computing infrastructure of the project.

Overall, the proposed integration infrastructure provides flexibility for integrating different modules into digital manufacturing solutions based on the QUALITY-RA.

3. Library of Packaged Enablers

3.1 Overview

Following paragraphs illustrate a set of QU4LITY digital enablers that will be packaged and distributed in-line with the integration concepts of the previous section. The enablers are summarized in the following table. Note that the presented list is non-exhaustive, as additional QU4LITY components are like to packaged and integrated in QU4LITY solutions as part of the second version of this deliverable.

Enabler	Description	IP Owner / Partner
FAR-EDGE DDA Platform	Platform for Data Collection and Analytics in Industrial Environments	INTRASOFT
Open VA	Platform for Data Management and Visualization	VTT
RUL Prediction Solution	Deep Learning Systems for Remaining Useful Life Prediction	ATLANTIS
DSS and ZDM Strategies	Manages zero-defect processes, by filtering out false alarms originated from predictive analytics and deciding mitigation actions to cope with defects	ATLANTIS
Security Privacy and Trust Enabler	Provides functionalities that protect QU4LITY systems against invalid accesses to internal systems and suspicious behaviours	ATOS
MGEP Converter Enabler	Middleware data for data acquisition and translation in various format, that boost interoperability in QU4LITY (see Section 4)	MON
Improved Failure Classification Enabler	Enables data-driven product quality, including identification and classification of various defects. Leverages historic data from various sources and systems in the shopfloor	TUDO
DLT Service: Quality Clearing House	Decentralized Workflow Management for Quality Management and ZDM scenarios in the supply chain	ENG
DLT Service: Decentralized Analytics Engine Configuration	Secure distributed processing of data analytics directives, represented via appropriate manifests in an edge computing environment	ENG

Enabler	Description	IP Owner / Partner
Secure Analytics Results Publishing	Enables distributed edge computing nodes to publish and share their analytics results in a decentralized fashion, towards creating a common analytics dataset	ENG

Table 6: QU4LITY Digital Enablers to be Packaged and Distributed in-line with the QU4LITY Integration Principles and Infrastructures

3.2 FAR-EDGE IoT Analytics Platform

3.2.1 Description

The Distributed Data Analytics (DDA) platform developed in the FAR-EDGE project is an IoT platform consisted from two core components which are the Edge Analytics Engine (EAE) and the Distributed Analytics Engine (DAE). The Edge Analytics Engine, as implied by the name, resides within the Edge Gateways of the FAR-EDGE infrastructure and is responsible in processing low level data streams collected from one Edge Gateway. The Distributed Analytics Engine resides within the Cloud tier and is responsible in processing data coming from the different EAEs. It is capable to provide more complex and consolidated analytics from the whole infrastructure (multiple Edge Gateways). They both follow the same principles and provide common functionality but have different scopes. One equally important asset of the DDA is the Analytics Processor (AP). APs are the heart of the Analytics system where APs instances are combined from the EAE and DAE in order to provide a complex Analytics solution. In order to provide an integrated DDA solution within the FAR-EDGE system various other components are involved. A high level description of these components/modules is provided in the following list:

- **The Data Routing and Pre-processing** (DR&PP) component (including device registry service) role is to forward data generated by Field sources while hiding technical complexities to both sides of the channel. Additionally, to the field layer data collection DR&PP component provides appropriate mechanisms in order to register Data Sources.
- **Cloud Tier Data Storage** stores data stemming from the DAE/EAE and provides opportunities for additional data analytics functionalities. Moreover, it provides a result storage repository.
- **Gateway Tier Data Storage** stores data stemming from the EAE and provides opportunities for additional data analytics functionalities. Moreover, it provides a result storage repository.
- **Model Repository** for Analytics supports the sharing of common digital models which are used from the FAR-EDGE components.
- **Distributed Analytics Engine** is a runtime environment hosted at the cloud. It is the programmable and configurable environment that executes data analytics logic over data coming from multiple Edge Gateways in order to produce complex and consolidated analytics. The DAE is also configurable,

while comprising multiple analytics instances that are driven by multiple smart contracts.

- **Edge Analytics Engine** is a runtime environment hosted in an EG. It is the programmable and configurable environment that executes data analytics logic locally in order to meeting stringent performance requirements, mainly in terms of latency. The EAE is in charge of data analytics within a single EG. The EAE is also configurable, while comprising multiple analytics instances that are driven by multiple smart contracts.
- **Analytics Processor** implements the data processing functionalities that are necessary to implement an edge analytics task. Analytics Processor can consume multiple data streams and produces one.
- **Open API for Analytics** implements all the public cloud service endpoints belonging to the FAR-EDGE Analytics Domain. These endpoints allow external systems to interact with the distributed data analysis logic deployed on the Cloud and Gateway Tier (Distributed and Edge Analytics Engine).

3.2.2 Availability

FAR-EDGE DDA is an Open Source software and is offered both as source code at GitHub but as well as a containerized solution at Docker Hub. In the following two sub-sections, we provide details for this availability.

3.2.2.1 FAR-EDGE GitHub Distribution

FAR-EDGE DDA source code is offered under the Apache License 2.0 and its components are available from GitHub. Links to the relevant repositories are provided below:

- Open API for Analytics
    - https://github.com/far-edge/open-api-for-analytics
- Edge Analytics Engine:
    - https://github.com/far-edge/edge-analytics-engine
- Model Repository:
    - https://github.com/far-edge/model-repository
- Digital Models
    - https://github.com/far-edge/DigitalModels
- Analytics Processors
    - https://github.com/far-edge/analytics-processors
- Analytics Dashboard
    - https://github.com/far-edge/analytics-dashboard
- MQTT Data Publishers:
    - File: https://github.com/far-edge/mqtt-file-data-publisher
    - Random Data: https://github.com/far-edge/mqtt-random-data-publisher

3.2.2.2 FAR-EDGE Docker Distribution

FAR-EDGE DDA 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.

FAR-EDGE DDA 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

3.3 VTT OpenVA Platform

3.3.1 Description

As detailed in deliverable D3.5, 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 and Python statistical computing environments.
- 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 analyzed is loaded from the sources to the database through a uniform data interface.

3.3.2 Availability

VTT OpenVA source code will be published in GitHub. A demonstration of using OpenVA in Qu4lity project setting will be offered in Docker Hub. The demonstrator will consist of two Docker images, one containing database of anonymized demonstration data and the other one containing analysis environment. To ensure easy installation of the demonstrator, Docker Compose will be used. Compose file will be published in the GitHub, allowing one step installation of the demonstrator.

3.4 RUL Prediction Solution

3.4.1 Description

ATLANTIS enabler with the name, Long Short-Term Memory (LSTM) approach for RUL Estimation, will be used for prediction of defects based on asset's deterioration rate. The ***Remaining Useful Life (RUL)*** analysis can provide valuable information regarding the deterioration rate of assets, as the former is defined as the length from the current time to the end of the useful life. Accurate RUL estimation plays a critical role in the improvement of the quality of the produced product and in the Zero-Defect Manufacturing process in general. The RUL analysis can be either ***model-driven*** or ***data-driven***. In the context of the Qu4lity project, ATLANTIS Engineering will deploy solutions from both the algorithmic families.

ATLANTIS Engineering will use the Long Short-Term Memory (LSTM) approach for RUL estimation, which belongs to Data-driven RUL calculation. LSTM approach can make full use of the sensor sequence information and expose hidden patterns within sensor data with multiple operating conditions, fault and degradation models.

Quality RA Mapping:

- Model-driven RUL Calculation
- Data-driven RUL Calculation

Used Technologies

- Docker
- InfluxDB
- Angular
- C# .NET Core ASP.NET
- SciPy
- Tensorflow

Docker Compose is a tool for defining and running multi-container Docker applications. With ***Docker Compose*** a ***YAML file*** will be used to configure our application's services. Then, it will be possible with a single command to create and start all the services from our configuration.

Docker Images will be collected from ***Docker Hub*** and will be complemented with volumes and parameters configured in the ***Docker Compose*** files.

Docker Hub will be used as ***Repository Management Service.***

GitHub is our preferred Code Management Tool.

3.4.2 Availability

An initial version is currently available, improvements will be made.

3.5 Decision Support System (DSS) and Strategies for ZDM)

3.5.1 Description

The main functionality of ***Decision Support System (DSS)*** is to manage zero-defect processes, by filtering out false alarms originated from predictive analytics and deciding the mitigation actions to cope with defects. The ***DSS*** is combined with and triggers the activation of semantically defined ***ZDM Strategies*** to control the propagation of defects and manage their occurrence in multi-stage production.

Quality RA Mapping:

- Prediction of defects based on asset's deterioration (through data analytics & machine learning)

Used Technologies

- Docker
- SQL Server
- C# .NET Core ASP.NET
- React
- Reactive Extensions for .NET (Rx)

All the previously mentioned technologies will be packaged, with all their dependencies, using ***Docker Compose***.

Docker Compose is a tool for defining and running multi-container Docker applications. With ***Docker Compose*** a ***YAML file*** will be used to configure our application's services. Then, it will be possible with a single command to create and start all the services from our configuration.

Docker Images will be collected from ***Docker Hub*** and will be complemented with volumes and parameters configured in the ***Docker Compose*** files.

Docker Hub will be used as ***Repository Management Service.***

GitHub is our preferred Code Management Tool.

3.5.2 Availability

It is not currently available. However, the intention is to proceed with the ***dockerisation*** of ***DSS***. It should be noted that internal reorganisation of the component has to be implemented to achieve that.

Data Models Used

- Proprietary

3.6 Security Privacy and Trust (SPT) Enabler (ATOS)

3.6.1 Description

This enabler will cover the security aspects as described D3.5. Components enrolled create a security layer providing authentication, authorization and monitoring functionalities to the ZDM ecosystem and the digital enablers. Therefore, it will provide the necessary protection to the infrastructure to proof against invalid accesses to internal systems and suspicious behaviours.

- The framework will act as the gateway, authorizing external systems to log with different roles in the internal infrastructure.
- Policy management deployed on both access control and events monitoring.
- Monitoring component will manage alarms and provide a risk assessment of the infrastructure, based on processing logs.
- Display capacities offer a viewing environment which listing the perform of task, showing relevant information graphs and diagrams.

This security enablers middleware will enhance communication access from side to side connecting as much components were needed.

3.6.2 Availability

These components are not available yet. The intention is to server the enablers dockerized on containers. Regarding Security Incident Event Management as many other security solutions, is a complex enabler difficult to set up. Before deploying the tool, there is the need of acquiring a deep understanding of how the infrastructure to be protected is connected, which assets are the more relevant for the organization, how they are connected, which types of attacks we are planning to detect, etc. With this information, we can plan the deployment of the different components (XL-SIEM agents) in the concrete subnets, the required sensors to be deployed, and the consequent rules for correlation. Usually, a consultancy activity should be done before deploying and managing a system like this, since, as many other enterprise solutions it requires a detailed planning. It's just the opposite to plug & play and maybe not be available except for the pilots.

3.7 MGEP Converter Enabler

3.7.1 Description

This converter enabler is a middleware platform that allows data acquisition and translation in multiple formats. It is composed of converters that enable data and protocol transformation. They also enable data storing in multiple repositories. The platform can work under different architectures (Request/response, Event Driven, Service Oriented Architecture, etc.). Converters can be customized by developers to their interoperability needs. This middleware platform is composed of three servers (see Section 4 for more details on the solution).

3.7.2 Availability

The enabler will be offered/composed using Docker Technology. A Docker Compose file in YAML (YAML Ain't Markup Language) format will hold the service architecture of the middleware platform. Servers will be included as images in that Docker Compose file. Images will be collected from Docker Hub[41] and will be complemented with volumes and parameters configured in the Docker Compose files or in Docker files. The different versions originated in the project will be stored in GitLab[42] to assure compatibility and lifecycle manager for development and operations. Converters and flows constructed using the tools will also be also stored in the GitLab repository for version compatibility.

3.8 Improved Failure Classification Enabler

3.8.1 Description

This enabler is a module that allows data-based prognosis of product quality. Therefore, it can use historical as well as online data from systems on shop floor (e.g. MES (Manufacturing Execution Systems)). Furthermore, it processes raw sensorial data to classify whether a product is defect or not. Moreover, the possibility whether a tested product is defect or not will be an output. The prognoses will be done by Machine Learning Algorithms, which need to provide these outputs at runtime, as soon as they receive new sensorial input.

However, the method is principally independent of the underlying data collection solution. For example, the data can be provided by several resources in real-time

[41] https://hub.docker.com/
[42] https://about.gitlab.com/

and offline. The data used for the prognosis are loaded from these sources by an interface.

3.8.2 Availability

The enabler will be developed in a Siemens specific environment. A description of the general functionality could be provided as well as a python pseudo code. This pseudo code could be provided e.g. on GitHub.

3.9 DLT Service: Quality Clearing House

3.9.1 Description

Quality Clearing House (QCH) enables a decentralized workflow for quality management in supply chain scenarios. Deployed on the Value Chain Ledger, 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 Blockchain technology, QCH records are secure and trustworthy: they are timestamped, immutable and non-reputable. 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[43].

The QCH digital enabler is going to be developed specifically for the QU4LITY project.

3.9.2 Availability

The packaging and distribution of DLT Services has been described in detail in §2.5.3, while generic instructions for their integration have been provided in §2.5.4. Source code and documentation for both the smart contract and the client library will be available from a public GitHub repository, licensed under the Apache v2.0 license[44]. The binary distribution of the client library and of the bindings to the DLT Infrastructure will be provided to requesting QU4LITY partners by the administrators of the Hyperlab facility (see §2.5.4).

[43] This holds true for real-world deployments. Actually, as explained in §2.5.2, in QU4LITY we choose to simplify the DLT Infrastructure and run all nodes on a single Cloud facility.
[44] This will allow anyone – not only QU4LITY partners – to deploy the DLT Service on their own HLF network.

3.10 DLT Service: Decentralized Analytics Engine Configuration

3.10.1 Description

Decentralized Analytics Engine Configuration (DAEC) enables secure and tracked distribution of data processing directives in a distributed data analytics system based on Edge Computing. Directives are issued by administrators as an *analytics manifest* that defines how data streams are to be processed by individual edge nodes, using a combination of predefined data processing elements and workflow instructions. Using the DLT Infrastructure as the distribution channel ensures a truly decentralized but also reliable system, even in scenarios where the network is open to multiple stakeholders: analytics manifests are signed, sealed and timestamped, so that no forgery or tampering is possible.

The DAEC digital enabler was originally developed in the FAR-EDGE project, under the name "AEC".

3.10.2 Availability

The same availability conditions described for the QCH digital enabler (see the previous paragraph on the QCH availability) apply here.

3.11 DLT Service: Secure Analytics Results Publishing

3.11.1 Description

Secure Analytics Results Publishing (SARP) enables the edge nodes of a distributed data analytics system based on Edge Computing to share their "local" results on the DLT Infrastructure, thus contributing to a common data set representing the combined results across the entire network. Individual analytics nodes will publish new data by pushing an incrementally updated data set when necessary. A subscription mechanism allows all nodes to be notified if and when some other node changes the global data set, they are interested in. The virtues of this management system lie in immutability and non-repudiation of published data sets.

The SARP digital enabler was originally developed in the FAR-EDGE project, under the name "ARP".

3.11.2 Availability

The same availability conditions described for the QCH digital enabler (described in earlier subsection about the OCH) apply here.

4. QU4LITY Platform Interoperability Solutions

4.1 QU4LITY Platform Interoperability Approach

4.1.1 Interoperability in Industry 4.0 Types of Interoperability

Interoperability is defined as "*the ability of two or more systems or components to exchange data and use information*" [VanderVeer08]. Computing systems are distributed and have a dynamic nature. In addition, these systems rely on heterogeneous technologies and use different information representations. These factors hamper the data exchange and data processing between different agents (machines, controllers, sensors …). Therefore, to achieve full interoperability, the exchanged information not only must have a common syntactic base, but also a common structure and common semantics [Ouksel99].

Interoperability is an important topic for the next generation of Industry4.0 systems [Nilsson18]. These systems are characterized by distribution and tend to comprise numerous smart devices that acquire, produce and exchange data in diverse formats and based on different semantics. This is the reason why interoperability is addressed in most reference architectures and reference architectural models for Industry 4.0 and the Industrial Internet of Things, such as the Reference Architecture of the Industrial Internet Consortium (IIC) and the Reference Architecture Model Industrie 4.0 (RAMI4.0). A detailed discussion about how these reference architectures address the issue of interoperability in industrial systems has been included in deliverable D2.11 of the project.

4.1.2 Types of Interoperability

There are four types/levels of interoperability [VanderVeer08]:

- **Technical interoperability**: enables machine-to machine communications through communication protocols and the hardware/software infrastructure required for those protocols to operate.
- **Syntactical interoperability**: provides a common syntax and encoding to the exchanged data, through data representation languages such as eXtensible Markup Language (XML) or HyperText Markup Language (HTML).
- **Semantic interoperability**: guarantees that there is a common meaning and understanding of the exchanged data. Specifically, semantic interoperability ensures that IT systems can exchange data in unambiguous ways, through sharing the meaning of data. It is also a foundation for enabling effective machine computable logic (e.g., machine learning algorithms), while boosting inferencing, knowledge extraction and knowledge discovery. Likewise, semantic interoperability facilitates the federation of data and services between information systems, which is particularly useful in scenarios involving interconnected cyber-physical systems and
- **Organisational interoperability**: Enables the data exchanged between organizations that rely on different infrastructures and heterogeneous

information systems. This level of interoperability requires a successful technical, syntactical and semantic interoperability.

The lower layers of the communication stack provide technical interoperability. The technological stack at this level is extend and conforms the physical and link layers necessary to enable the Network layer and its main protocol Internet Protocol.

For nearly a decade, industrial automation solutions (notably solutions involving digital components) have attempted to address interoperability concepts using a variety of solutions and approaches. As a prominent example, Web Services and Service-Oriented Architecture (SOA) have been extensively used to address technical and syntactic interoperability in industrial systems, through converting products, processes, and other resources into digital services. These approaches have been effective in bridging the gap of the different platforms and their data formats, yet they do not enable a shared understanding of the meaning of data.

Industry 4.0 systems and solutions must ultimately incorporate semantic interoperability as well, in order to enable their components to exchange data and information with unambiguous meaning. Ontologies and semantic web solutions are used in various industries to solve similar problems. Recently, effective semantic systems have been introduced and validated in the scope of the Internet of Things (IoT) (e.g., [Gyrard15], [Datta18]), which could be potentially applicable to Industry4.0 as well (i.e. Industrial IoT systems). Furthermore, in the IoT space there are also solutions for semantic interoperability across IoT platforms (e.g., [Ganzha17], [Lanza18]). Nevertheless, Industry4.0 introduce harsher challenges given that they comprise highly customizable production processes and complex automation systems that can be hardly modelled based on a single standard or format. Specifically, it is typical for digital automation systems to comprise or adhere to a wide various of standards [Lee15], [Nilsson18]. In this context, QU4LITY starts from known approaches for syntactic and semantic interoperability and advances them in versatility, functionality and ease of use. The project’s approach to interoperability is illustrate in following paragraphs.

4.1.3 Interoperability in the QU4LITY-RA

Systems compliant to QU4LITY-RA are distributed cyber-physical systems that can be deployed in a cloud/edge infrastructure. Hence, the integration and packaging infrastructure specified in Section 2 can ensure the communication between the various components. However, to ensure their interoperability there is a need for sharing digital models across the different components. Hence, the QU4LITY-RA specification includes a set of Digital Models and Vocabularies, which are destined to be shard across the different components, to facilitate the exchange information across the various components of the QU4LITY-RA, in ways that ensure a common interpretation of information. The QU4LITY-RA considers two levels of data interoperability are considered:

- **Syntactic interoperability** aims at exchanging information in a common data format with a common protocol to structure the data; and

- **Semantic interoperability** aims at interpreting the meaning of the symbols in the messages correctly. These interoperability components provide a flexible method of composing services so that the system behaviour can be adapted at run-time to enable advanced ZDM processes.

As outlined in the previous paragraph, syntactic interoperability can be achieved through the proper translations across different data formats. In this direction, QU4LITY provides a middleware platform that is described later in this section. Likewise, semantic interoperability is based on the semantic models of the project and related mechanisms that are described in the next section of the deliverable.

4.2 QU4LITY Middleware for Platform Interoperability

4.2.1 Background and Rationale

Most companies have their own proprietary platforms and IoT gateways. They communicate using standard data formats such as JSON or XML, but there is not always common agreement on the structure or the semantics of data shared. On the other hand the cloud infrastructure offered by big players (Amazon, Azure, MindSphere,...) is extensive. Interoperability between proprietary platforms is a challenge and companies might need the support of translation middleware to communicate with commercial platforms.

The objective is to build translator enablers according to the requirements and specifications set in other WPs and to use them in the pilots. This middleware translation system will provide a number of servers to accommodate different architectures (Request/Response, event driven) and data formats (JSON, XML …). The middleware will also enable the deployment of converters and flows that will perform the actual translation between protocols. See the following section for further detail on the middleware platform. The middleware translator is under development in task 5.4. The solution will be dockerised for easy deployment in pilots in this task. Guidelines on how to build converters and flows will be also provided.

4.2.2 Platform Description

The middleware platform for interoperability to be built in task 5.4 will consist of a Docker solution with three servers (see Figure 15):

1. An Enterprise Service Bus (ESB) server that will enable the deployment of software artefacts for the conversion of data between IoT gateways and platforms. Our solution proposes the usage of WSO2 (Web Services Oxygenated 2). WSO2 EI (WSO2 Enterprise Integrator) is a generic open-source platform that makes possible the integration of different applications, systems and data. It enables enterprise services to collaborate dynamically between SOA based systems. The ESB integration capabilities enable the creation of service endpoints, API, sequences of services and software artefacts to fully deploy converters. These converters will be placed as proxies

between the client and the final server for the application. The endpoints constructed for the converters will collect data from clients, transform it to the correct data format and resend the requests or messages to the appropriate broker or endpoint in its final destination.

2. An edge broker server for message publication and subscription. This broker will enable event driven architectures for those converters. Converters will be subscribed to different queues at the broker. When a message arrives in one of those queues, the converter will fetch it, translate it and resend the messages or requests to the appropriate broker or endpoint in its final destination.
3. Similar to the ESB, another integrator server will be included in the middleware platform. A Node-RED server will be also provision. Node-RED has similar integration capabilities to those of WSO2. For OPC-UA connection types, the implementation of flows (converters) is straightforward and easier to accomplish than with the other alternative. Many systems and platforms rely on OPC-UA for their communication and control. In this sense this server will work as the Operation Technology (OT) to Internet Technology (IT) adapter. Consequently, this software platform has been selected to be included.

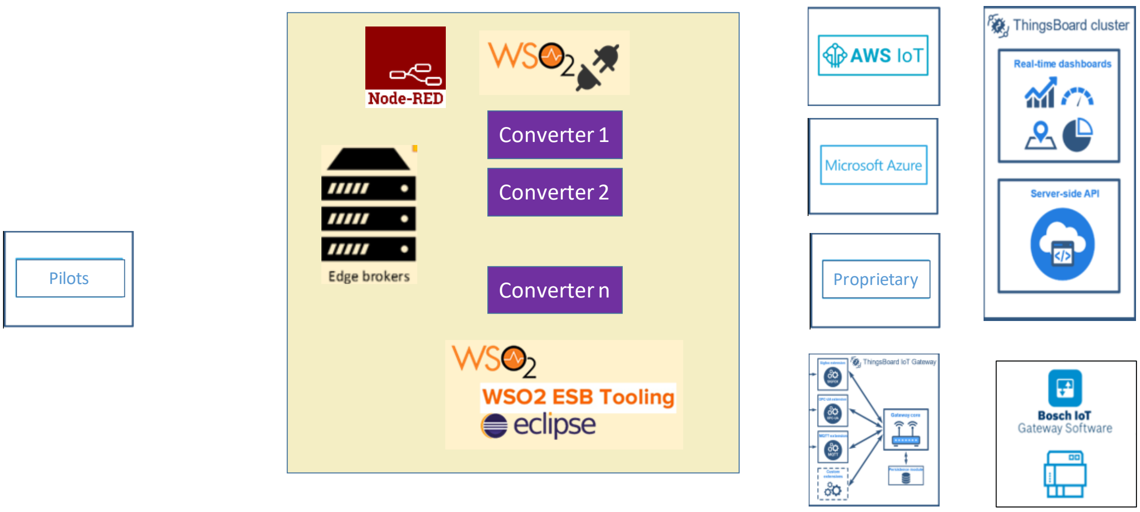

Figure 15: Middleware Platform for Interoperability

The ESB environment additionally offers a tooling for the design and construction of converters (See Figure 16). The WSO2 Developer Studio is an Eclipse plugin that enables the modelling of APIs and endpoints. Models are created using the graphical interfaces provided by the tool and generate XML files. The tooling is downloaded with Eclipse and could be easily included with the containerized solution.

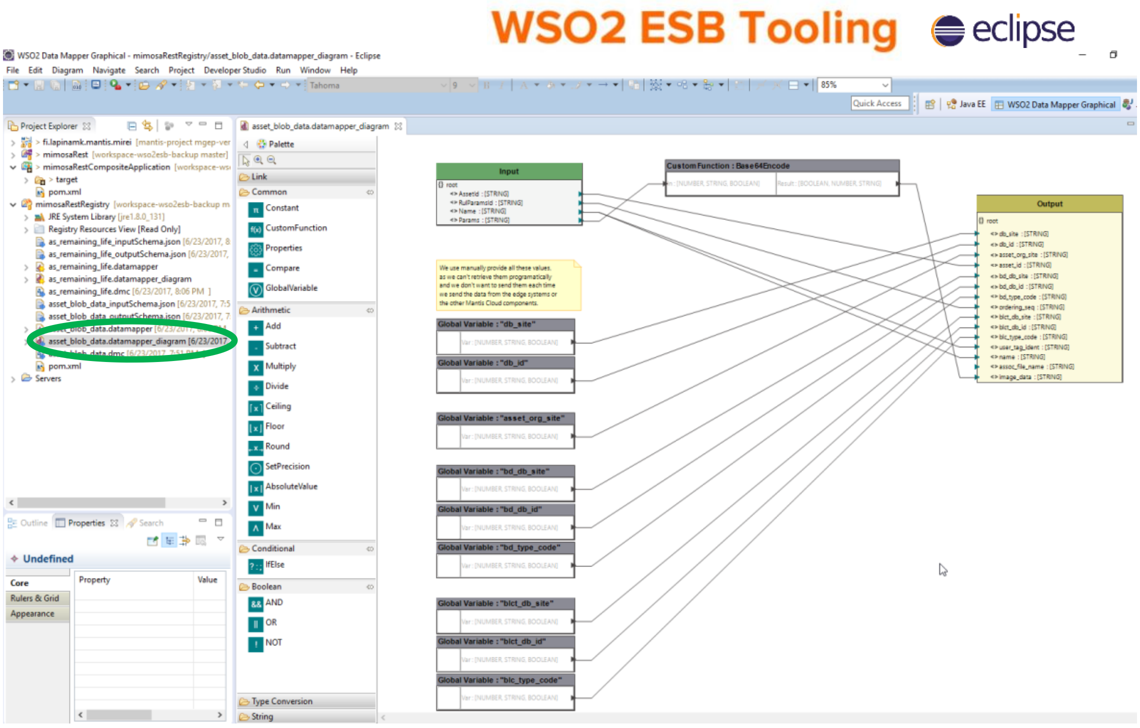

Figure 16: ESB Mapping Tooling

Similarly, Node-RED offers palettes of nodes and a workspace for the design and construction of flows (converters) (See Figure 17). These tools are included with the server.

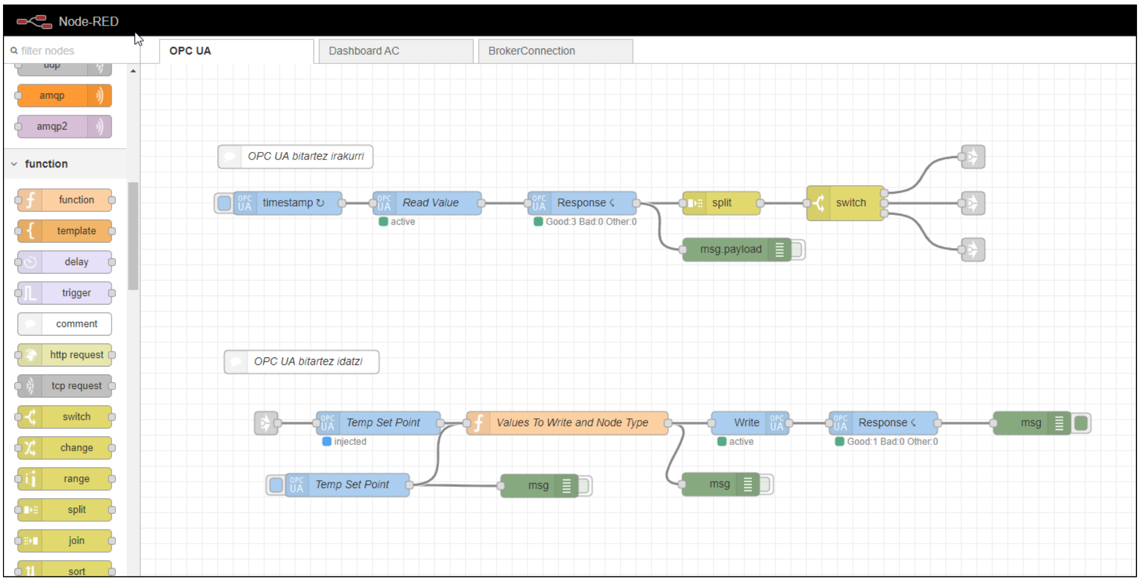

Figure 17: Node-RED flows

4.2.3 Guide for Using the Platform in QU4LITY Use Cases

As is described in deliverable D5.7 the middleware platform for interoperability is composed of three parts; 1) An infrastructure solution composed of three servers, 2) a number development tools and guidelines and 3) the converters constructed for the APIs.

The infrastructure part is a generic solution that will be offered using Docker technology. The Docker components will be stored in a repository (Docker Hub or similar) for anyone to download and use. Additionally, development tools and guidelines to develop specific converters will be provided. Two main tools are available. In the case of WSO2, an Eclipse Tooling is proposed for developers. Guidelines will indicate how to download and use this tool. In the case of Node-RED, the flow editor is included with the server and is accessed using the web browser. The guidelines are documents explaining how to construct translation artifacts both for WSO2 (APIs, endpoints, sequences …) and Node-RED (flows).

The converters are software components that can be deployed in the infrastructure (servers provided with Dockers). Those converters can be stored in any of the repositories outlined in section 2.1. Guidelines will indicate how to deploy those converters in the infrastructure and how they should be stored for version compatibility.

5. QU4LITY Semantic Interoperability Solutions

5.1 QU4LITY Semantic Interoperability Approach

One of the main approaches for achieving semantic interoperability in Cyber Physical Systems (CPS) involves the use of a set of common semantic models for representing the status of the physical world in the cyber world [Kunold19], [Kefalakis19b]. The use of a common semantic model by all the distributed elements of the CPS system ensures that:

- All the different elements of the CPS system understand the status of the plantfloor based on the same semantics. This concerns the operational status of production systems and processes, as well as the metadata that describes their state (e.g., their location in the shopfloor).
- State changes are consistently shared across different applications. Specifically, any changes to the status of the production systems or processes is reflected on the semantic model and therefore becomes readily accessible/available to other applications.

QU4LITY will follow a similar approach to semantic interoperability based on the semantic models that have been specified as part of WP2 of the project and in-line with the QU4LITY-RA semantic interoperability concept that is outlined in the previous section. These models will provide a basis for semantic unification and state sharing across different applications and components of a CPS-based manufacturing system. The QU4LITY models will be used as outlined in the following paragraphs. In addition to semantic models (i.e. ontologies), the project will offer a set of more lightweight digital models with simpler semantics that could be also used to boost interoperability across the different elements of ZDM application.

5.2 QU4LITY Semantic Data Models and Ontologies

5.2.1 Background and State of the Art

Semantic modelling and ontology engineering have been widely used to facilitate semantic interoperability among different platforms and sources in a heterogeneous information system. Ontology is an explicit, formal specification of a shared conceptualization of a domain of interest. Ontology engineering unites a series of methodologies and methods for building ontologies. An information model provides the ability to abstract different kind of data and provides an understanding of how the data elements are related. A semantic model is a type of information model that supports the modelling of entities and their relationships. The total set of entities in a semantic model comprises the taxonomy of classes that can be used to represent the real world. Semantic modelling can help defining the data and the relationships between entities.

Task T2.5 of QU4LITY project has specified a range of digital models and vocabularies to represent the data related to product quality such as plants, materials, production processes and quality processes etc. The specifications of these digital models should be based on popular existing industrial standards for data interoperability such as B2MML, CAEX, PLCOpen, COLLADA, AutomationML, MTConnect and MIMOSA and so on.

A detailed survey of these models and what they offer is beyond the scope of this deliverable and can be found in relevant surveys [Peres16], as well as in the outcomes of task T2.5 of the project. In terms of standard-based models for industrial automation, notably mentions include:

- IEC 61512 BatchML, which is an XML based implementation of the ANSI/ISA-88 Batch Control family of standards.
- IEC 62769 (FDI), which includes an information model that represents automation systems' topologies, including field devices and the communication networks that interconnect them.
- IEC 62264 B2MML, which is an XML based specification and implementation of the ANSI/ISA-95 family of standards.
- ISO 15926 XMplant, which provides support for digital modelling of plant information, based on the ISO 15926 specification. It covers the structure, the geometry and 3D models about a plant.
- IEC 62424 CAEX, which supports XML-based representation of plant information, including all the components of a plant in an hierarchical structure. It adopts an object-oriented philosophy.
- IEC 62714 AutomationML, which provides an XML-based format for the description of engineering and automation processes in a plan. It is based on three other standards, namely CAEX (for plant information), COLLADA (for geometry and kinematics) and PLCopen (for control applications).
- OPC UA's Data Model, which defines a generic object-oriented Data Model (DM), which focuses on devices representation.
- MTConnect, which provides an XML-based format for exchanging data between the shop-floor and IT applications, including data about devices, topologies and components characteristics.

Also, the MIMOSA association has produced a set of standards for condition monitoring and maintenance, which can be used for standardizing information representation and access as part of asset management and maintenance applications. Specifically, the MIMOSA Open Systems Architecture for Enterprise Application Integration (OSA-EAI) specifications are destined to address one of the most common problems of the development of maintenance systems, which is the fact that information and data are fragmented across different systems such as engineering, maintenance, operations and reliability related systems. MIMOSA models facilitate a common and unified understanding of business requirements, which boosts communication and collaboration across team members. Moreover, it provides a foundation for designing databases and datastores where maintenance information will be persisted. It also boosts data re-use, data sharing and data repurposing across different maintenance systems operated in different factories and across the supply chain.

The OSA-EAI specifications are based on the representation of the information objects required for a business application based on information model that adheres to OpenO&M standards. The MIMOSA information model complies with ISO standards such as:

- ISO 13374-1: Condition monitoring and diagnostics of machines -- Data processing, communication and presentation -- Part 1: General guidelines and
- ISO 13374-2: Condition monitoring and diagnostics of machines -- Data processing, communication and presentation -- Part 2: Data processing.

Furthermore, the construction of the digital models should also take into account existing industrial data exchange platforms and reference architectures such as the International Data Spaces (IDS) and Reference Architectural Model Industrie 4.0 (RAMI 4.0). Overall, semantic interoperability in QU4LITY takes into account the output of T2.5, which has reviewed the aforementioned standards and reference architectures in the first stage of the project to guide the development of the digital models for QU4LITY. T2.5 has specified an initial set of ontologies that can support ZDM applications.

5.2.2 Use of Ontologies and Digital Twins in QU4LITY

The development of the domain ontologies follows the structure of existing upper-level ontologies like Basic Formal Ontology (BFO) to facilitate interoperability among domain ontologies that are built in its terms through a process of downward population. Existing domain ontologies related to advanced manufacturing developed in some previous and on-going projects should be reused as much as possible. For this purpose, Industrial Ontologies Foundry (IOF) has been created and QU4LITY partners are playing active roles in it. The aim of is to co-create a set of open ontologies to support the manufacturing and engineering industry needs and advance data interoperability. It involves government, industry, academic and standards organizations to advance data interoperability in their respective fields.

To comply with the global version of the project, the design of the ontology-based models uses User Story Mapping (USM) method to analyze user stories and stakeholders' requirements which are defined in T2.1. By this way, it is possible to define a series of competency questions and extract a list of concepts that covers entire domain based on usages, roles and activities. These concepts will then be compared with the vocabularies defined in existing domain ontologies to produce a shared vocabulary for QU4LITY project. It is this shared vocabulary, and its associated links to an ontology, which provides the foundation and capability of machine interpretation, inferences, and logic.

Another critical aspect to consider for data interoperability is Digital Twins (DT) which has been widely implemented in many advanced industrial sectors. The concept of DT is fostered by the various existing technologies such as IIoT, CPPS, AI, 3D modeling, system simulation, digital prototyping etc. A basic DT model consist of three main parts including: physical products in Real Space; virtual products in Virtual Space; and the connections of data and information that tie the virtual and real products together. In some cases, DT data and services are integrated into the DT model to better describe the scenarios in the factories of the future. To handle the heterogeneous data from different platforms and sources, there is a trend recently to combine semantic technologies with DT. Semantic models can capture complex systems in an intuitive fashion, which can be summarized in standardized ontology languages, and come with a wide range of off-the-shelf systems to design, maintain, query, and navigate semantic models. This characteristic makes semantic modelling a promising paradigm to address the challenges that DT development is facing currently. Currently the concept of Cognitive Twin (CT) is under development, by integrating augmented semantic capabilities with DT, to identify the dynamics of virtual model evolution, and enhancing the decision-making based on DT.

As previously mentioned, the development of QU4LITY digital models and domain ontologies is not started from scratch. Several models that are closely related to QU4LITY scenarios have been developed in previous projects, which can be reused and could greatly facilitate the development of QU4LITY digital models. Among them, the MPFQ-model, which was developed as part of the EU-project "inteGration of pRocess and quAlity Control using multiagent technologies (GRACE)" (GRACE consortium 2011), has been selected to reuse. The MPFQ-model is named after its four main elements: Material (M), Production Process (P), Product Functions /Features (F), Product Quality (Q). This model focuses on the manufacturing phase of the product while considering the strong interactions between product design and plant planning. At the manufacturing phase the planned product quality is brought into reality by assembling procured materials within production processes. The final product produced satisfies or dissatisfies the customer requirements and is being sold on the market. The four elements of the MPFQ-model and the interrelations among them are illustrated in Figure 18. More details about this model can be found in the deliverables corresponding to T2.5.

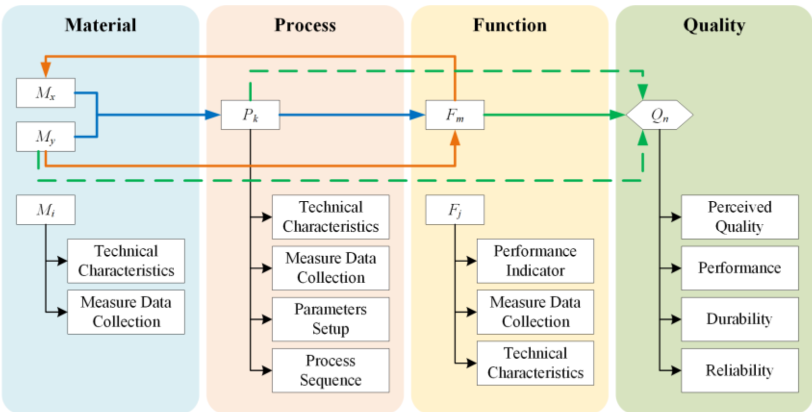

Figure 18: Four main elements of the MPFQ-model and their interrelations

In some other projects participated by QU4LITY partners in relevant domains, like FALCON, DIVERSITY and Z-BRE4K etc., several domain ontologies have been developed and will become main components of IOF. Some of them, e.g. the predictive maintenance ontology and the Product-Service Systems (PSS) ontology (https://www.industrialontologies.org/), will be reused to support the development of QU4LITY domain ontologies.

Currently, the development of digital models and domain ontologies are still under development. A draft conceptual version of a structure integrating the MPFQ-model, DT model, Multi-Agent Systems (MAS) and ontology has been constructed by the moment. It will be modified and improved according to the progress of the project. This part of work is mainly conducted by T2.5 in WP2 and close collaboration with this task is expected in the future.

5.2.3 Guide for Using the Semantic Models in QU4LITY Use Cases

The function of sematic models in QU4LITY is to integrate and merge various data sources by the rules and relations defined in the ontologies. An ontology plays the role of a meta-model for diverse data formats, enabling information search and storage, so that to provide an understandable framework for both the machine and the human. It can facilitate the implementation of advanced data analytics, such as data mining or machine learning, thus, to realize intelligent tasks like pattern recognition, defect detection, sequential decision making and so on.

Some existing projects, such as FALCON Virtual Open Platform and Z-BRE4K platform and DIVERSITY, have already shown how semantic models can promote the ZDM vision. For example, a typical product-oriented predictive maintenance task involves cloud service, manufacturing machines, like milling machines, and measuring machines, like CMM. These milling machines and CMM machines are usually provided by different machine suppliers, thus, data from these machines have different formats and types. In the case that anomaly of a part is detected, it requires certain actions for maintaining milling machine performance. It means that milling machine data is associated with CMM data and the cloud service should facilitate convergence of all the data types with consideration of association between data sources. In this case, semantic models, such as the previously mentioned MPFQ-model, are capable of retrieving correlations among different data sources that are related to that defect based on the entities and relations defined in the ontology. For process-oriented AQ tasks, take aircraft development as an example, different kind of simulations models are developed and performed at the early stage of an aircraft program but there is a lack of flexible solution supporting industrial multi-model integration for system verification and multi-disciplinary optimization for decision-makings within the company and with the extended enterprises. In this case, ontology can be used to support model-based systems engineering by defining the correlations among the developed meta-models, and models.

In QU4LITY, the upper-level ontology will be defined first following the BFO methods and keeping align with IOF. Then, two domain ontologies will be developed based on two pilots, i.e. the GF pilot digital machine and part twins for zero defect manufacturing, and the Airbus pilot trade space framework for AQ manufacturing systems' design. The GF case is product oriented and the Airbus one is process oriented. These two cases will be used to demonstrate how a specific domain ontology is developed and applied to support AQ vision.

In terms of development tools, Protégé will be used to create, edit and verify the ontology correctness. Protégé is the most used tool for the development of ontologies, either the development from the scratch, and the merging, importing, querying and export of ontologies. It is a free, open-source platform that provides a suite of tools to construct domain models and knowledge-based applications with ontologies. In Protégé it is possible to create ontologies based on different types of expressiveness, being perfect for modelling a knowledge environment. Additionally, there are many plugins to be used with Protégé, e.g. to support the validation phase and to export the ontology in different formats.

The complete instruction about the semantic model development and their applications in pilots will be delivered in the final version of this task together with any updates derived from task T2.5.

5.3 Lightweight Digital Models for ZDM

5.3.1 Background

To offer simplified consolidation and management of digital information across distributed applications, QU4LITY will also extend an existing digital models infrastructure to support industrial use cases. Specifically, Digital Models developed in the scope of background projects of the partners (i.e. H2020 FAR-EDGE, H2020 PROPHESY) will be considered, including:

5.3.1.1 Models for Representing Factory Data and Metadata

The following constructs support the representation of 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, as can be seen in Figure 19. 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.

Based on the above entities it is possible to represent the different data sources of a digital shopfloor in a modular, dynamic and extensible way. This is based on a repository (i.e. registry) of data sources and their manifests, which keeps track of the various data sources that register to it.

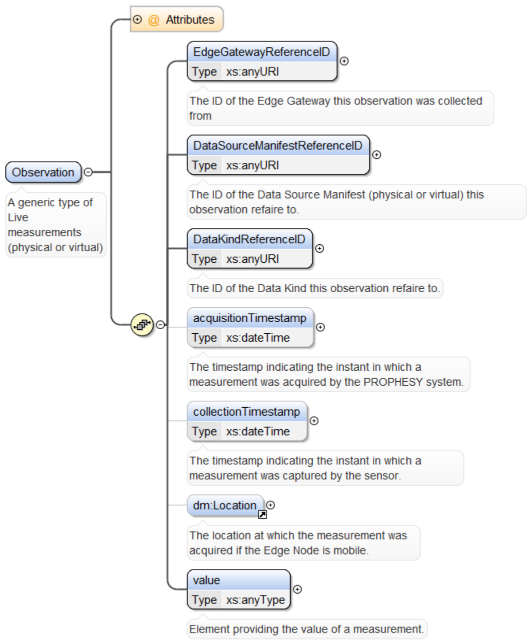

Figure 19: Snapshot of the Observation Entity

5.3.1.2 Models for Representing Factory Data Analytics Metadata

To facilitate the management and configuration of analytics functions and workflows over the various data sources, several analytics-related entities are also specified, including:

- **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 FAR-EDGE, 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.

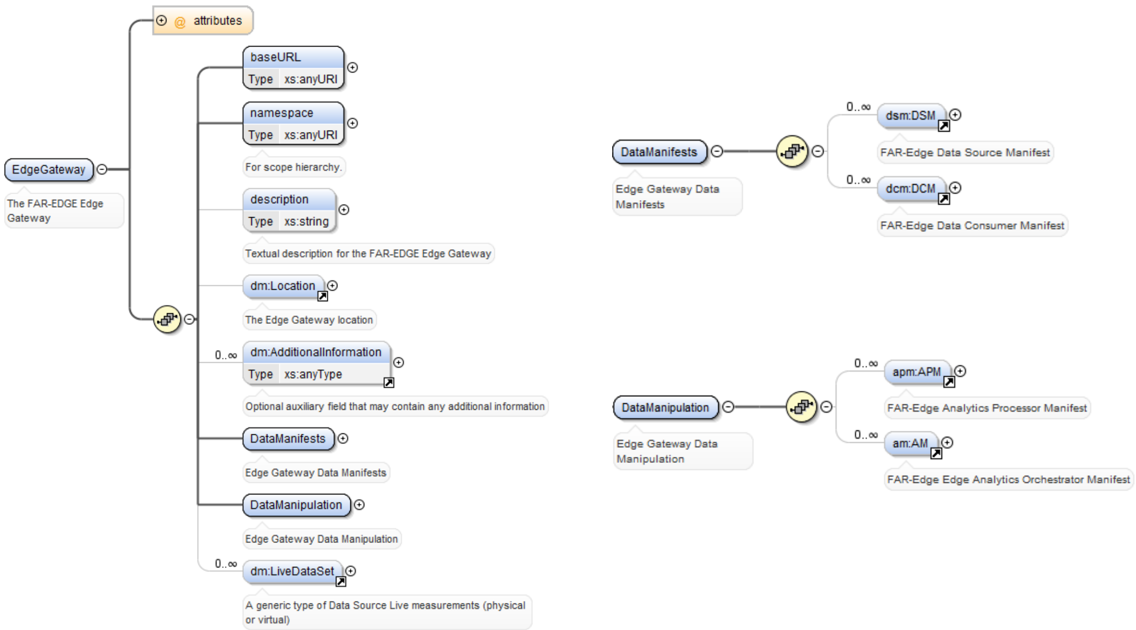

Figure 20: Snapshot of the Digital Models Structure developed in FAR-EDGE & PROPHESY

These Digital Models for distributed data analytics follow a hierarchical structure, which defines the different relationships between the various entities. For example, an edge gateway comprises multiple data source manifests. Each one of the latter is associated with a data source definition. Likewise, LiveDataSets are associated with instances of data sources i.e. data sources manifests. Figure 20 illustrates a snapshot of the digital models structure, which shows the association of each edge gateway with data source manifests and data analytics manifests. A more detailed presentation of the hierarchical structure of our data models is beyond the scope of this deliverable and can be found in [Kefalakis19a].

5.3.1 Lightweight Digital Models for ZDM QU4LITY Extensions

The lightweight digital models presented above offer some special characteristics in order to be adaptable in various scenarios which includes the ZDM. The Entities that facilitate this functionality are the **Data Kind** (DK), **Observations** and **Additional Information**.

5.3.1.1 Data Kind

As introduced above this entity specifies the semantics of the data source data, which provides flexibility in modelling different types of data. It can be used to define virtually any type of data in an open and extensible way. This can be achieved by being able to describe the type, format and data kind of the values that are produced in any given system. More specifically the "kind" of the data is represented with the QuantityKind attribute which is an abstract classifier that represents the concept of "kind of quantity". A QuantityKind represents the essence of a quantity without any numerical value or unit. (e.g. A sensor -sensor1- measures temperature: sensor1 has quantityKind temperature). The Data Kind is not only used to describe Data Sources that are captured by the ZDM system but also Data Sources that are produced from the system (i.e. an analytics algorithm). The root element of the Data Kind entity is the "DK" and is depicted in Figure 21.

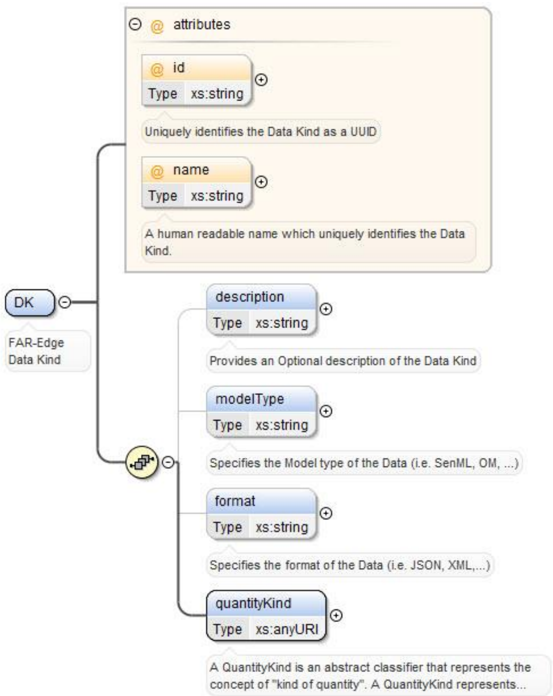

Figure 21: Data Kind entity

5.3.1.2 Observation

As introduced above the Observation entity models and represents the actual data/measurement that stem 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. An Observation is associated with a timestamp and keeps track of the location of the data source in case it is associated with a mobile (rather than a stationary) data source. Hence, it has a location attribute as well. Observation holds the measurement/result of a Data Source at the "value" entity which is of type anyType. This means that it can support any type of value (even complex structures) that are identified from the Data Kind it id referencing. Following the Data Kind and Data Source the Observation is not only used to describe data that are captured by the ZDM system but also data that are produced from the system (i.e. an analytics algorithm produces Observations).

The root element of the Observation entity is the "Observation" and is depicted in Figure 19 above.

5.3.1.3 AdditionalInformation

AdditionalInformation is a generic entity which allows the extension of the existing data model with additional attributed that may be required. A list of AdditionalInformation entities are used in the EdgeGateway (see Figure 20) and Core digital model structures in order to provide an optional unlimited auxiliary field that may contain additional information and can be used for further extensions. More specifically, it serves as the root of the type definition hierarchy for any schema and it has the unique characteristic that it can function as a complex or a simple type definition, according to context. The root element of the "AdditionalInformation" model is depicted in Figure 22.

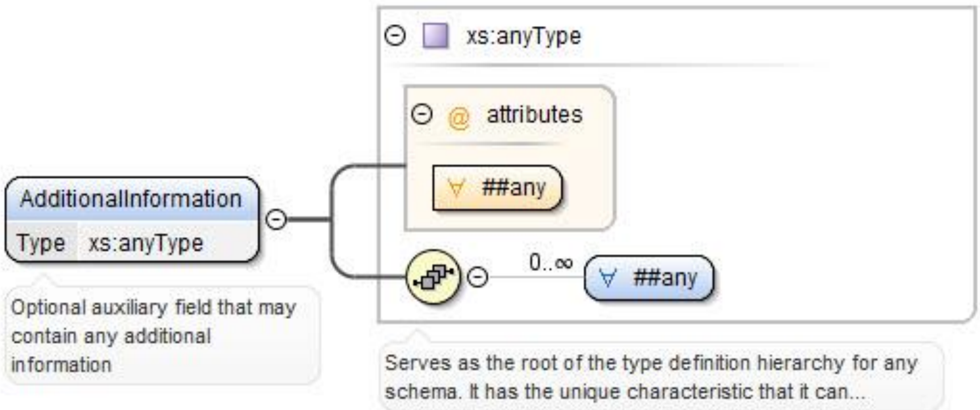

Figure 22: Additional Information entity

6. Conclusion and Future Outlook

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 as presented in Section 3. The QU4LITY partners have started the "containerization" of the digital enablers that have been presented in earlier deliverables of WP3. These digital enablers

The integration and interoperability solutions that are presented in this deliverable are fairly 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 version of the deliverable was destined to be a report that would describe the project's approach and enablers for integration, packaging and interoperability. The second and final version of this deliverable (namely deliverable D3.14) will present the actual implementation of the integrated enablers and their integration/ use in the QU4LITY pilot systems, including the 14 pilots of the project and additional pilot systems that will be implemented as part of the project's Open Call processes.

7. References

[Datta18] Datta, S.K.; Bonnet, C. Advances in Web of Things for IoT Interoperability. In Proceedings of the 2018 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW), Taichung, Taiwan, 19–21 May 2018.

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

[Faltinski12] S. Faltinski, O. Niggemann, N. Moriz, and A. Mankowski, "AutomationML: From data exchange to system planning and simulation," IEEE International Conference on Industrial Technology (ICIT), 2012, pp. 378–383.

[Ganzha17] Ganzha, M.; Paprzycki, M.; Pawłowski, W.; Szmeja, P.; Wasielewska, K. Towards Semantic Interoperability Between Internet of Things Platforms. In Internet of Things Book Series (ITTCC); Springer: Cham, Switzerland, 2017.

[Gyrard15] Gyrard, A., Serrano, M., and Atemezing, G. A. (2015). Semantic Web Methodologies, Best Practices and Ontology Engineering Applied to Internet of Things. IEEE World Forum-Internet of Things (WF-IOT)

[Kefalakis19a] Nikos Kefalakis, John Kaldis, John Soldatos, Mauro Isaja, "Digital Models for Data Analytics and Digital Twins in Industrial Automation Applications, Introduction of a Common Interoperability Registry, in the Proc. Of The Thirteenth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, held in Porto, Portugal during September 22 - 26, 2019 (UBICOMM 2019).

[Kefalakis19b] Nikos Kefalakis, Aikaterini Roukounaki, John Soldatos, "Configurable Distributed Data Management for the Internet of the Things", Information 10(12): 360 (2019)

[Kunold19] I. Kunold, H. Wöhrle, M. Kuller, N. Karaoglan, F. Kohlmorgen and J. Bauer, "Semantic Interoperability in Cyber-Physical Systems," 2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Metz, France, 2019, pp. 797-801.

[Lanza18] Jorge Lanza, Luis Sánchez, Juan Ramón Santana, Rachit Agarwal, Nikos Kefalakis, Paul Grace, Tarek Elsaleh, Mengxuan Zhao, Elias Z. Tragos, Hung Nguyen Dang, Flavio Cirillo, Ronald Steinke, John Soldatos: Experimentation as a Service Over Semantically Interoperable Internet of Things Testbeds. IEEE Access 6: 51607-51625 (2018)

[Lee15] J. Lee B. Bagheri H.-A. Kao "A cyber-physical systems architecture for industry 4.0-based manufacturing systems" Manufacturing Letters vol. 3 pp. 18-23 2015.

[Mathew06] Mathew, Avin D. and Zhang, Liqun and Zhang, Sheng and Ma, Lin (2006) A review of the MIMOSA OSA-EAI database for condition monitoring systems. In Proceedings World Congress on Engineering Asset Management, Gold Coast, Australia.

[Nilsson18] J. Nilsson and F. Sandin, "Semantic Interoperability in Industry 4.0: Survey of Recent Developments and Outlook," 2018 IEEE 16th International Conference on Industrial Informatics (INDIN), Porto, 2018, pp. 127-132.

[OSA-CBM] MIMOSA OSA-CBM, "Open System Architecture for Condition-Based Maintenance", available at: http://www.mimosa.org/mimosa-osa-cbm

[Ouksel99] A. M. Ouksel and A. Sheth, "Semantic interoperability in global information systems," ACM Sigmod Record, vol. 28, no. 1, pp. 5–12, 1999.

[Peres16] R. S. Peres, M. Parreira-Rocha, A. D. Rocha, J. Barbosa, P. Leitão and J. Barata, "Selection of a data exchange format for industry 4.0 manufacturing systems," IECON 42nd Annual Conference of the IEEE Industrial Electronics Society, Florence, 2016, pp. 5723-5728, doi: 10.1109/IECON.2016.7793750.

[Souza 18] H. Souza, "How to dockerize any application", May 2018, available at: https://hackernoon.com/how-to-dockerize-any-application-b60ad00e76da, last accessed at: March 2020

[VanderVeer08] H. van der Veer and A. Wiles, "Achieving technical interoperability," European Telecommunications Standards Institute, 2008.

List of figures

Figure 1: Three Phase Methodology for Establishing and Validating the Project' s Infrastructure for Components Packaging and Integration................................10
Figure 2: Stack Management environment Preferences....................................16
Figure 3: Docker Stack Management/Monitoring Tool Preferences.......................17
Figure 4: Code Management tool Preferences..18
Figure 5: Repository Management service Preferences 18
Figure 6 A Complete Git branching model 21
Figure 7: The Edge Computing Paradigm Illustrated..23
Figure 8: Classical System Architecture vs. Virtualization.................................24
Figure 9: Management System...26
Figure 10: TTTech Nerve Blue – Deployment of three different types of workloads 26
Figure 11: Management of workloads...27
Figure 12: Nerve Blue – Management of edge devices and workloads..................28
Figure 13 -Engineering's Hyperlab leaflet...31
Figure 14: Snapshot of the QU4LITY Reference Architecture 35
Figure 15: Middleware Platform for Interoperability..52
Figure 16: ESB Mapping Tooling 53
Figure 17: Node-RED flows 53
Figure 18: Four main elements of the MPFQ-model and their interrelations 58
Figure 19: Snapshot of the Observation Entity 61
Figure 20: Snapshot of the Digital Models Structure developed in FAR-EDGE & PROPHESY 62
Figure 21: Data Kind entity..63
Figure 22: Additional Information entity 64

List of tables

Table 1: Stack Management environment Preferences 16
Table 2: Docker Stack Management/Monitoring Tool Preferences 17
Table 3: Code Management tool Preferences 18
Table 4: Repository Management service 19
Table 5: List of QU4LITY Digital Enablers, Their Owners and Their Mapping to QU4LITY-RA 37
Table 6: QU4LITY Digital Enablers to be Packaged and Distributed in-line with the QU4LITY Integration Principles and Infrastructures 40

List of Abbreviations

AM	Analytics orchestrator Manifest
AP	Analytics Processor
APD	Analytics Processor Definition
APM	Analytics Processor Manifest
AQ	Autonomous Quality
B2MML	Business To Manufacturing Markup Language
CBM	Condition Based Maintenance
CMM	Computer Maintenance Management
CPS	Cyber Physical Systems
CT	Cognitive Twin
DAE	Distributed Analytics Engine
DCD	Data Channel Descriptor
DCM	Data Consumer Manifest
DDA	Distributed Data Analytics
DK	Data Kind
DLT	Distributed Ledger Technology
DR&PP	Data Routing and Pre-processing
DSD	Data Source Definition
DSM	Data Source Manifest
DSS	Decision Support System
DT	Digital Twin
EAE	Edge Analytics Engine
EI	Enterprise Integrator
ESB	Enterprise Service Bus
HLF	HyperLedger Fabric
HPC	High Performance Computing
HTML	HyperText Markup Language
IDS	International Data Spaces
IIC	Industrial Internet Consortium
IIoT	Industrial Internet of Things
IIRA	Industrial Internet Consortium Reference Architecture
IOF	Industrial Ontologies Foundry
IoT	Internet of Things
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
LSTM	Long Short-Term Memory
MAS	Multi-Agent Systems
MES	Manufacturing Execution System
OSA-EAI	Open Systems Architecture for Enterprise Application Integration
QA	Quality Assessment
QADM	Quality Assessment Data Model
QCH	Quality Clearing House
PL	Private Ledgers
RA	Reference Architecture
RAMI4.0	Reference Architecture Model Industrie 4.0
RUL	Remaining Useful Life
SARP	Secure Analytics Results Publishing
SIEM	Security Information and Event Management
SPT	Security Privacy and Trust
SUM	Shipping Unit Manifest
USM	User Story Mapping

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

Partners: