DIGITAL FACTORY ALLIANCE

Open Autonomous Quality Services Engineering and Processes

Position Paper

24/02/2022

Authors

Birgit Obst (Siemens)

Feryal Fulya Horozal (ATB)

Marcel Altendeitering (ISST Fraunhofer)

Andrea Chiodi (Synesis)

Published: Open Autonomous Quality Services Engineering and Processes

Internal identification

WP ● White Paper PP ● Position Paper TR ● Technical Report PR ● Policy Report MR ● Management Report

PP-OT-2022/0003

Contributing Organisations

Co-funded by the Horizon 2020 Programme of the European Union. Grant agreement ID: 825030

Table of Contents

Executive Summary

This document gives an insight into several technological approaches towards open autonomous quality services engineering and processes.

After a short introduction and localization within the QU4LITY Reference Architecture, information about simulation framework building blocks for factory wide multiscale ZDM process modelling and multi domain simulation is presented in chapter 2 and as well as technologies for adaptive digital shopfloor automation, in chapter 3. Regarding service interoperability you will find description of Open secure Industrial Data Space (IDS) and autonomous data management for ZDM in chapter 4 as well as an Open API approach in chapter 5.

The document will close with a conclusion.

1 INTRODUCTION

Within the QU4LITY project, open autonomous quality services engineering and processes addresses several aspects.

First, with open autonomous quality (AQ) services we address the ability to enable zero defect manufacturing (ZDM) by autonomous quality control. This guarantees a high automation level with analytical, predictive, and prescriptive decision support including human centred automation.

Second, with definition of engineering and processes of such open autonomous quality services we address the architectural service interoperability.

Therefore, we want to give an insight in the technologies of:

- Factory wide multiscale ZDM process modelling and multi domain simulation
- Adaptive digital shopfloor automation
- Open secure Industrial Data Space (IDS) and autonomous data management for ZDM
- and Open APIs.

These technologies are located within the QU4LITY Reference Architecture as shown in Figure 1.

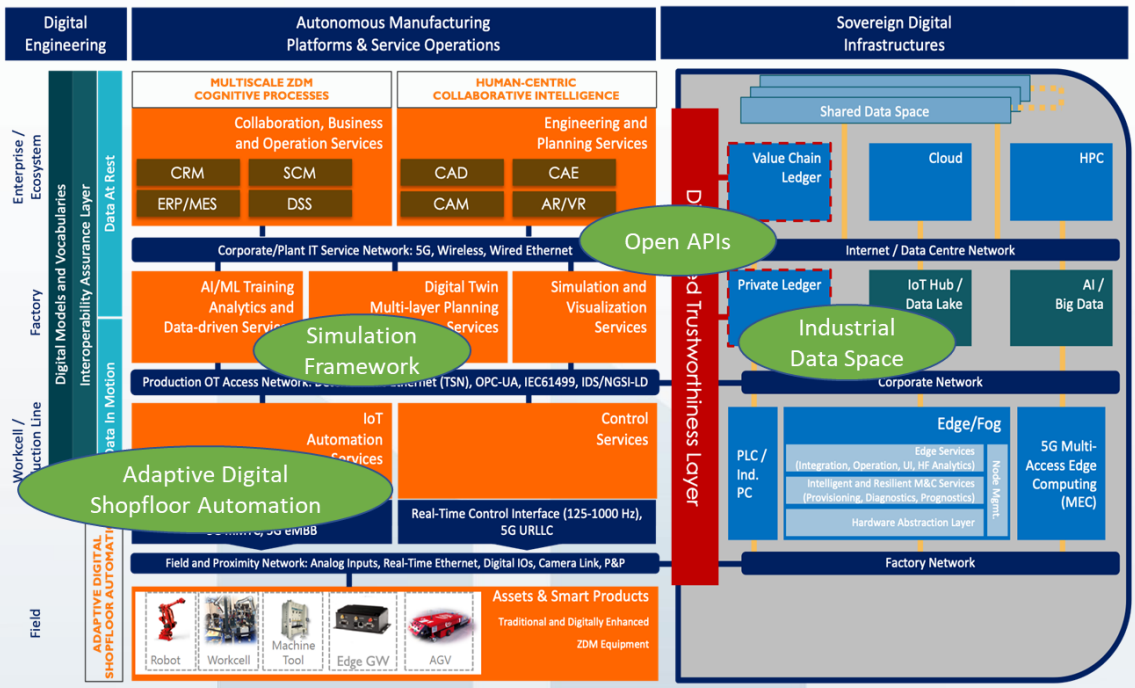

Figure 1 Quality Reference Architecture including service technologies

The deeper insight into these 4 technological approaches will give you an idea of what to consider for autonomous quality service establishment and value creation towards zero defect manufacturing, facing the challenges of Industry 4.0. Nevertheless, these are only selected topics and do not cover all aspects.

2 THEMATIC 1: SIMULATION FRAMEWORK

Quality services for analysis, predictive and prescriptive decision support may start with a simulation-based evaluation. For an easy approach, a multi-domain simulation framework for multi-stage zero defect manufacturing is developed, defined by building blocks with functionalities shown in Figure 2.

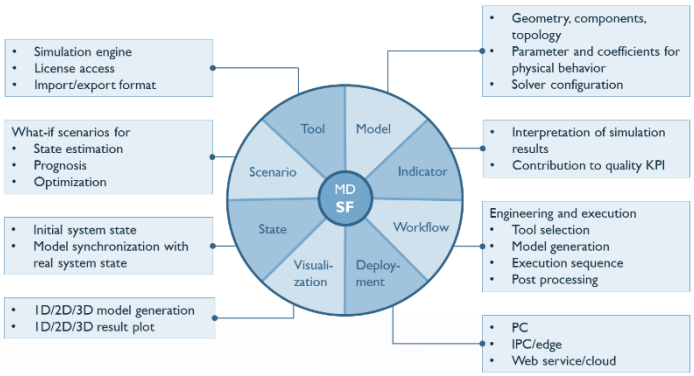

Figure 2 multi-domain simulation framework building blocks

Quality supporting services will be able to engineer and process simulation models to evaluate the current or future status of quality and zero-defect manufacturing KPIs:

1. Model: Model generation and configuration
    - A simulation model needs to be defined, loaded, or adapted regarding e.g., geometry (like CAD), the selection of components from libraries for systems or the topology of how elements are connected and are interacting.
    - In addition, the components (dimension, function, …) need to be configured and parameterized, as well as the interacting connections between them. And, to define the physical behaviour (material stiffness, flow, reaction, …), coefficients for algebraic or differential equations are necessary.
    - To solve the model corresponding equation system the configuration (e.g., step size, convergence, …) of appropriate (to be selected) solver (linear, nonlinear) is necessary.
2. Tool: Tool selection and interoperability
    - To execute a simulation model, the simulation engine must be selected.
    - For most of the simulation tools a licence is necessary, so for execution the access to the licenses must be defined.
    - Additionally, and especially for automated execution, all the manipulations regarding the simulation model and its evaluation need an input and output, which need to be in the correct format for import and export.
3. Scenario: Definition of simulation experiment for application and evaluation

- Depending on the application of the simulation the evaluation delivers a specific system state (state estimation).
- Gives a forecast, based on a specific state, and based on the future boundary conditions (prognosis).
- optimizes parameters for a given problem regarding defined target in several iterations.

4. State: System state definition for execution
    - To run a simulation experiment, the state variables and boundary conditions need initial values. Usually, a data pre-processing is necessary to define a model consistent system state and set the state values.
    - For online applications a continuous model synchronization with the real system is necessary. For every synchronization step the state values need to be set as well as specific model parameters to also calibrate the model continuously if necessary.
5. Visualization: Visualization as user interface
    - For some cases it is necessary to visualize the system during model generation or adaption, to support the user in defining geometry, topology or parameter setting etc. Otherwise import of model defining files is an option.
    - In almost the same manner the simulation results need to be visualized, to support the user in result interpretation in 1D, 2D or 3D plots. Otherwise, the results are exported in files.
6. Deployment: Infrastructure of simulation engine to run an application
    - Usually, simulation tools are running on a PC, but also applications using HPC infrastructures are common.
    - For distributed evaluation on local devices, simulation models are more and more able to run on edge devices, e.g., to support model predictive control or advanced analytics and diagnosis.
    - For applications with high computational efforts, also distributed cloud computing is an option to run simulation.
7. Workflow: Engineering and execution of control sequence for simulation application
    - Select the appropriate simulation tool.
    - Model the system behaviour or adapt specific model variables.
    - Define the evaluation sequence of scenarios whether the execution is a state estimation, a prognosis, or an optimization.
    - Interpret the simulation results and prepare them for further treatment.
8. Indicator: Simulation output as contribution to quality or ZDM key performance indicator
    - The evaluation of simulation model has the aim to support in decisions. Therefore, the decision supporting indicators, as results from simulation, need to be interpreted towards the set target. This process can be done manually by the user, but also automated.
    - Regarding the targeted quality and ZDM key performance indicators (KPI), the simulation outputs are semantically linked within the autonomous quality control service for performance management systems.

The multi-domain simulation framework delivers the necessary interfaces based on these functional blocks, to get quality services within the framework architecture as services, operating and exchanging data with the data space layer connected to the industrial data space (IDS) via Open APIs (see Figure 3).

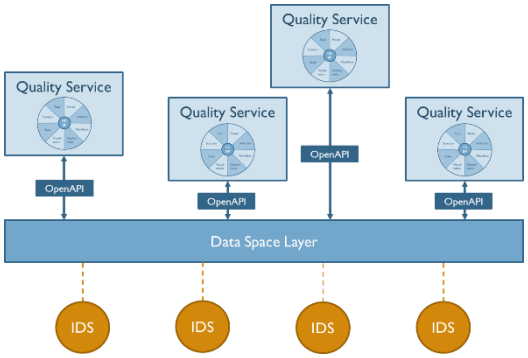

Figure 3 multi-domain simulation framework architecture

To connect simulation tools and their simulation models for execution it is necessary to create a framework counterpart as so-called wrapper for the different simulation tools. The wrapper must support the simulation framework interfaces and must operate with the simulation tools regarding their interoperability. For any simulation tool to be integrated into the autonomous quality control services by the multi-domain simulation framework such a simulation wrapper must be prepared and implemented before and is not part of the framework itself.

For the Proof of Concept, the framework supports the simulation tool Tecnomatix Plant Simulation, which is the one used by the pilots for quality issues. The main important element is the so called "Experiment Manager". With this Plant Simulation component, several experiments can be defined by setting model parameters and plotting the relevant results, e.g., KPIs. A Python wrapper was developed to get access to the tool engine on server side to run the experiments in an automated manner. To feed the Experiment Manager of Plant Simulation, a configuration file (e.g., Excel) can be used to define the experiments with their parameter variants. Due to licence issues the Plant Simulation model must run locally. The results, also stored in the Excel file, can be selected and plotted on the web service visualization. Comparing the experiments on the plots, the analysis towards the target can be evaluated and support in decisions. The application itself is a web service realized with Mendix.

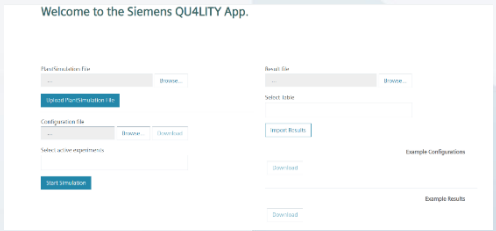

Figure 4 Graphical user interface or simulation framework

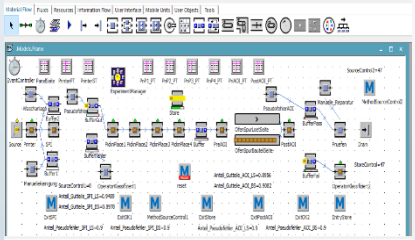

Figure 5 Plant Simulation model

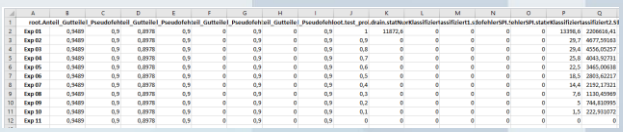

Figure 6 Experiment configuration in Excel

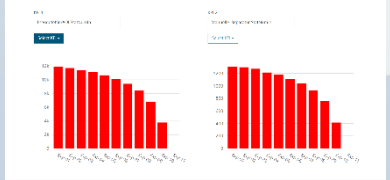

Figure 7 resulting KPIs for different experiment

3 THEMATIC 2: ADAPTIVE DIGITAL SHOPFLOOR AUTOMATION

Within the QU4LITY project, Adaptive Digital Shopfloor Automation supports the engineering of Zero-Defect Manufacturing (ZDM) processes based on the Autonomous Quality (AQ) paradigm. The concept of ZDM in QU4LITY is a holistic quality management approach that ensures both process and product quality by reducing product defects through corrective, preventative, and predictive techniques. It leverages data-driven technologies to guarantee that no defective products leave the production site. The domain of adaptive digital shopfloor automation includes functionalities supporting automated monitor and control of Assets and Smart Products in the physical world, contributing to the Automation and Control domain of QU4LITY. Automation and Control domain includes functionalities enabling efficient and reliable data exchange and intelligent control over the physical production processes and assets.

The main goal here is to provide the means for the development and deployment of adaptive digital shopfloor and ZDM processes. Within the scope of this goal, the project focused on engineering the capability of the digital shopfloor and ZDM processes to adapt themselves to various changing conditions of the shopfloor and to reconfigure themselves following the detection of defects, equipment degradation patterns and other related criteria while the machines/production lines are in operation. We contribute to adaptive digital shopfloor automation and ZDM processes by providing a comprehensive toolset of adopted and customised tools and technologies, which are used in the QU4LITY Pilots and Use Cases. This toolset covers a wide spectrum of technologies including data monitoring-based context awareness technologies, ontology-based data processing technologies, reconfigurable robotics technologies, automation technologies for Industry 4.0, composable digital shopfloor technologies, augmented reality technologies, AI and machine learning technologies and data exchange enablers. These technologies are introduced and analysed in depth in the project deliverable document "D5.6 Tools and Techniques for Adaptive Shopfloor Automation and ZDM Processes". Based on these technologies, the toolset for adaptive digital shopfloor automation in QU4LITY consists of the following tools and technologies:

1. Context awareness framework (used in the Continental Pilot)
2. Ontology-based data processing (used in the Airbus Pilot)
3. Adaptive visual quality inspection (used in the Kolektor Pilot)
4. NxTTech IDE and EcoRT runtime system (used in the NXT and ASTI use case)
5. AUTOWARE composable digital shopfloor verification and validation framework
6. Pacelab WEAVER augmented reality for operational support (used in the Continental Pilot)
7. Improved failure classification enabler (used in the Siemens Pilot)
8. Data analytics tool for additive manufacturing (used in the Prima Pilot)

The eight solutions contribute to many areas of the Digital Manufacturing Platform & Service Layer of the QU4LITY Reference Architecture including IoT automation services, data-driven modelling and learning services, simulation and human-centric visualization services, control services, field devices & products and engineering and planning services.

Context Awareness Framework is a generic framework for data monitoring and context extraction based on monitored "raw data" provided by systems/sensors as well as knowledge available in different systems, and possibly external services analytics. It processes data coming from connected data producers (systems, devices, equipment, products, processes) and can use this data and combine it with services provided by external service providers to extract the current situation, i.e., context, of these connected data producers. The context is then available to further use. The overall context extraction process follows the structure of sensory systems or CPS to collect & deliver context relevant data about data producers to reason for this context based on an ontology, which is further evolved and enhanced through similarity measures. Within the Continental Pilot the Context Awareness Framework was extended by an interface to enrich the context extraction mechanism with additional information provided by an external service, in this case a big data analytics service. This tool application makes use of cloud technology and the big data analytics technology with the support of the Context Awareness Framework.

Ontology-based Data Processing is a tool consisting of a set of integrated components for requirements management, architecture definition, visualization, system integration and verification of the design phase of an industrial system using ontology-based techniques for data and knowledge modelling. The domain knowledge is represented in ontologies so that not only the process information themselves are managed and contained, but also a high-level abstraction of business, for example, classes or types of process, requirements and so on is achieved. This allows ensuring a broad usage of the knowledge in similar contexts, and reuse of that information, so that different individual scenarios of process, process steps and their relations be represented under the same framework. This strengthens the model-based system engineering methods and thus facilitate the cooperation and communication between different domains and areas in terms of engineering. Furthermore, the ontology-based data processing also enhances the consistency of data/information flow and couple the different upper- and downstream tasks, which ensures the quality of engineering work.

Adaptive Visual Quality Inspection is a tool developed for the autonomous detection of defected items and their exclusion and from final products through visual quality check using adaptive robot trajectory generation. The main goal is to improve the visual quality by real-time fault detection through goal-directed acquisition of images for visual quality inspection in a feedback loop (e.g., changing the viewpoint or area of inspection), and possibly prediction of failures based on advanced analytics and artificial intelligence (setting process parameters, etc). The implementation is set at the JSI Technological Experimental Facility called ReconCell. The robot with an in-hand camera can adapt its position and by extension the camera viewpoint, in order to be able to see the required area of the inspected object. Appropriate positioning of the camera is crucial for proper visual inspection. It allows the robot to rotate the object either with the use of its own motors in the case of the passive table or rotate the inspected part continuously with the active fixture. Thus, the required viewpoints with respect to the object are achievable. The combination of different viewpoints enables optimal visual inspection. In the sense of reconfigurability, it allows the system to optimally change the vantage point. The robot changes the orientation of the objects, but also its vantage points.

NxTTech IDE and EcoRT Runtime System are based on the IEC 61499 standard for distributed systems. The NxTTech IDE is used to develop control applications presented as networks of functional blocks, to distribute and map parts of the application to different hardware, deploy, run, test, and debug the application. It is also used to develop libraries of predefined functional blocks that can be used to easily create applications by drag and drop. The EcoRT runtime system is ported to different hardware on which the IEC 61499 application should be executed. The NxTTech IDE and EcoRT runtime system are used in an AGV use case where a IEC 61499 control application is developed in the IDE and tested in the communication between AGVs when

negotiating the crossing of an intersection and also when exchanging information with other hardware like machines, lights or doors.

AUTOWARE Composable Digital Shopfloor Verification and Validation Framework is a framework for the integration and seamless interworking of the digital enablers of the autonomous quality paradigm. It is used for the validation of data transfer between machines and testing of data against data quality dimensions, which contributes to data accuracy and data quality during data transmission. The framework is based on two main tools: A test workflow tool designed to help throughout the software validation process, by helping the user to automate tests and requirements management, testing and documentation, and the Q-Digital Automation certification framework where core components are certified for concrete usage scenarios. The components can be tested against relevant standards, and these tests can be automated, and any bug or potential issue can be detected in the design and product configuration process before eventual implementation at customer premises. If the product undergoes these testing without any incidence and complies with the requirements demanded, it may also be finally certified against defined usage scenarios. The verification and validation process includes designing and implementing testing environments, coordinating test processes, defining test tasks, formal documentation reviews, code reviews, integration and unitary tests, acceptance and functional tests, dynamic and static methods, implementing security and quality standards and training customer staff.

Pacelab WEAVER Augmented Reality for Operational Support is a software suite for streamlining the authoring, management, and the execution of extended (augmented-mixed-virtual) reality applications for operations, maintenance, and training, and enables one to create, deploy and execute a wide array of virtual, augmented and mixed reality solutions from the same set of technical data. This software is leveraged in a use case of operation support where the goal is to support workers during their job with new augmented technologies and new wearable devices such as MS HoloLens and handheld devices such as tablets and smartphones. It allows for adapting to any identified risk of failure using digitally guided trouble-shooting support and entails the human-in-the-loop of ZDM in the shopfloors through extended reality operational guidance as a human machinery communication interface.

Improved Failure Classification Enabler is a machine learning based tool developed for pseudo error identification, reduction, and autonomous learning of process states, which enables product inspections with a high degree of inspection severity. Machine learning algorithms are particularly suitable for autonomous learning of process states. Such algorithms are adapted and integrated into an inspection gate in order to reduce pseudo errors and to relieve process experts during double inspections. This can help improve the overall product quality rate as well as increase the testing efficiency. To achieve this, information about the final product quality needs to be obtained during the ongoing manufacturing process. This tool is used in a use case of Siemens Digital Factory Division for manufacturing SIMATIC products. The core of the manufacturing process is the production of the circuit boards, which are later assembled with the housing parts to form the final product.

Data analytics tool for Additive Manufacturing is an enabler for adaptive digital shopfloor and ZDM processes. It contributes to managing the data exchange and cooperation between independent and heterogeneous tools. The role of this tool is to collect the meta-information generated from different tools in order to aggregate the information together with the process parameters collected from the machine tool. This functionality is realized leveraging the state-of-the-art and widely accepted communication protocol MQTT. The data analytics tool monitors different types of process parameters while manufacturing process is running and stores the data in a database management system. The tool is hosted in a device that provides the required resources in terms of memory, storage, processing, and networking capabilities. It is designed to be hosted in an edge device, provided with virtualization features. The components of the Data analytics tool can be allocated to one or more virtual machines.

4 THEMATIC 3: INDUSTRIAL DATA SPACE

Overview

To reach the goals of ZDM and AQ in light of complex supply chains and information networks we intended to leverage the standards of the International Data Space (IDS) and extend it with a component for autonomous data management. Specifically, we aimed to implement two IDS entities in a concrete usage scenario: the IDS Dataspace Connector (DSC) and the Data App. The Data App realizes autonomous data management by ensuring a high-quality standard in data transfers. We, hereby, intended an application that measures the quality of a data set and reports it as an easy-to-understand rating while offering compatibility with the IDS ecosystem. By reducing the complexity of this issue down to a concrete ration between zero and one the end user gains a tool to support decision making in inter-organizational data exchange. For example, the IDS compatibility allows to outsource the data analysis tasks to a partnering firm or research institute while sustaining data sovereignty and ensuring high quality data.

IDS Communication

The developed solution consists of two components: a Data Space Connector (DSC) and a Data App written in Python. For a simple integration in an IDS Environment the Data App can exchange data directly with the DSC. We therefore relied on an open-source version of the DSC[1], which includes a docker compose file for fast provisioning. Thus, it can be connected directly to another DSC at a production facility or the given data source in general. The presented Data App follows the guidelines of the International Data Spaces Association (IDSA) and the IDS Information Model[2], which means that it structures objects at the DSC as shown in Figure 8. All communication workflows shown are REST-based. To operate the DSC and the Data App need *Resources*, which can be created automatically through the Data App. Once a complete *Offer* has been created, the endpoint for data input of the Data App can be linked to it. From then on updated data gets pushed to the Data App as soon as it arrives at the DSC, the App processes it and pushes the results of the Data Quality Analysis back to the Connector. From there on other parties can subscribe their Connectors to the updated Resource if the Contract allows it. Since the DSC currently saves files as strings the Data App can automatically decode Base64 if necessary. In case a JSON file is provided the App selects the relevant values from it (configurable through the corresponding environment file) and converts it to a CSV file for simpler data handling. In case of a CSV file no conversion is needed.

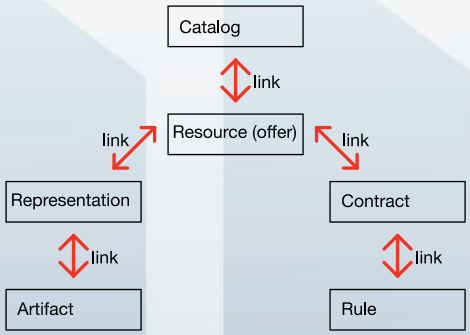

Figure 8 - IDS Connector Configuration

[1] https://github.com/International-Data-Spaces-Association/DataspaceConnector
[2] https://github.com/International-Data-Spaces-Association/InformationModel

Data Quality Analysis

For implementing the Data App, we used the programming language Python as it is well suited for data analytics tasks and offers many useful frameworks. However, traditional Python analytics have its limitations with regards to large volume files, so we used Apache Spark[3] as our processing engine to read in the input data. Upon creating a Spark RDD[4] (resilient distributed dataset) the data gets split up into smaller chunks, which are then analysed successively. Through this approach, the Data App is more scalable and can handle large amounts of data while utilizing well known python packages like SciKit-learn[5]. The utilized technologies are shown in Figure 9. To determine the quality of the provided data set, we combined four different data quality measures that are suitable for sensor-based inputs and cover a variety of data quality dimensions. Using an Isolation-Forest algorithm from the Scikit learn package each data block gets analysed for outliers and based on the commonness of outliers an **outlier measure** is calculated. Furthermore, a potential **concept drift** in the data set is assessed on a per block basis. To store the results a MySQL database gets created by the application which is tailored specifically for its needs.

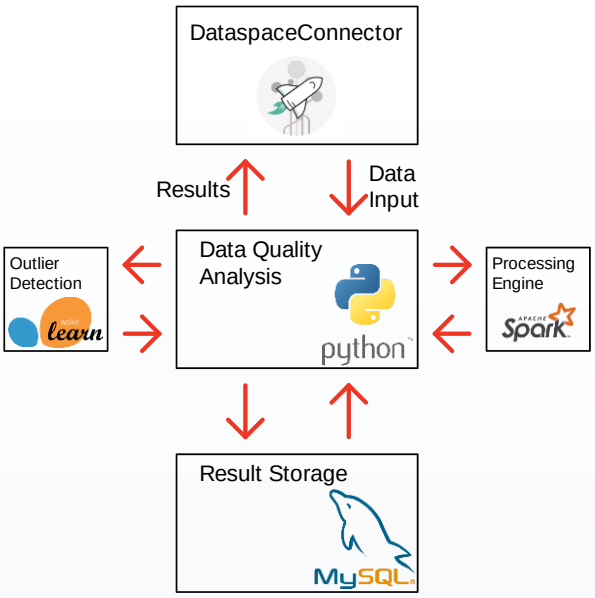

Figure 9 - Architecture Diagram

Secondly, the concept of the current block is saved as an approximation, meaning that the boundaries and averages of each sensor are remembered in storage. Through this knowledge the previous concept can be compared to the next one leading to a measure of Concept Drift. The last two data quality measures that are considered are the **No Value Measure** and the **Constant Measure**. The first one looks for sensors that do not provide data over the whole block and the second one detects sensors staying constant for a long time.

All these measures are saved in the database and then combined to an overall measure that is averaged over all blocks in the dataset. Finally, the result in form of an easy-to-understand ratio between zero and one is sent back to the DSC and filed as its own resource such that it can be read by another connector or a user directly.

[3] https://spark.apache.org
[4] https://spark.apache.org/docs/latest/rdd-programming-guide.html
[5] https://scikit-learn.org/stable/

Additionally, the created database includes a table that is formatted specifically for visualization with Grafana and always held up to date. This means that with very little effort an illustration of the analysed data as well as the calculated measurements can be added to the system if needed.

To evaluate the accuracy of the data quality analysis it has been tested with real-world data sets provided by the Mondragon pilot. The results reflected actual differences in the data quality of the datasets, which demonstrates that the algorithm is functioning and capable of effectively rating datasets based on their quality characteristics.

5 THEMATIC 4: OPEN APIS FOR ZDM PROCESSES

The QU4LITY project addresses the challenge of identifying guidelines for the adoption of common Open APIs, which could support the communication between the varieties of heterogeneous components composing a ZDM system.

The identification of a convenient Open APIs approach, to be supported and promoted by QU4LITY, began with an analysis of the APIs currently supported by the technologies of the QU4LITY partners. This was conducted by circulating a questionnaire where each partner could describe the technologies they support.

The resulting list of interfaces supported by QU4LITY partners' technologies is representative of the wide spectrum of interfaces that are available in the technologies suitable for implementation of ZDM processes. Some of them refer to the communication protocol on which they are based (for instance MQTT, OPC UA, AMQP, WebSocket, etc.) others refer to the software tools exploited for their implementation (Flink, Kafka, Spark, RabbitMQ, etc.).

The diversity of the several communication solutions adopted suggested the assumption of some fundamental criteria, which was considered as a rationale for the definition of an Open API strategy.

In order to provide interoperability and to enable the effective interconnection of different digital platforms and achieve the desired objectives of quality in ZDM, the APIs has not only to define the syntactic aspects of the messages exchanged via the network connections: it also has to provide enough details for the semantic aspects needed to exploit the API and make the interaction between tools possible.

Moreover, the semantics of the exchanged messages is likely at the origin of technological and architectural decisions, which could involve, among other aspects, the choice between synchronous or asynchronous communication.

The availability of an API definition language was considered among possible criteria to evaluate an API. SOAP [soap], for example, adopts the Web Services Description Language (WSDL) language to describe the syntax of the implemented functionalities. Other APIs adopt different dialects of IDL (Interface Definition Language) [idl] which offer similar functionalities. These forms of structured documentation also enable several design-time functionalities, like syntax checker, testing tools, generation of reference manuals, etc. The API description language could even extend beyond mere syntax, including information about the semantics of operations, based on some known and shared ontology.

However, while most of the existing definition languages are intended to extend a specific protocol, the last generation results in this area are moving toward a complete independence from any specific implementation.

Three aspects of interoperability were taken in account.

First, and more important, the **semantics of the exchanged information** must be defined to the maximum possible extent. Some of the several semantic aspects could be easily agreed, because of the availability of some underlying common knowledge. A simple example could be the unit-of-measure of an exchanged value. If agreed in advance, it should be properly documented and included in the "contract" between the parties; otherwise, it should be included in the exchanged messages as an auxiliary information, which in turn needs to be semantically well defined, for instance against a list of possible values either derived from a standard or agreed locally between partners. Again, documented and reciprocally agreed. Such a level of cooperation necessarily involves the developer teams, as is hard to be automated in any way. In other and more challenging cases, the actual meaning of the exchanged data could not even be represented in the message itself, as it requires a deep and reciprocal understanding of the collaborating subsystems. Even more so, the effort to document it at the best detail is a design duty.

Second, the language to be adopted in order to make the exchanged message **reciprocally understandable** must be well defined in advance. This brings to the selection of some specific syntax, which should be common to all or most of the endpoints in order to reduce the effort of interpreting them. However, several techniques exist to support the automated translation between different syntaxes, based on the assumption that their underlying logic is similar. As an example, conversions between XML and JSON syntax could be easily automated, even if some variety of resulting dialects needs to be constrained and controlled during the process. Another common method is to dynamically select the requested syntax. This is a common practice, for instance, in REST based services, where a simple additional parameter could instruct the server about the syntax to adopt (e.g., &fmt=xml or &fmt=json, &fmt=html, etc.)

Last, but obviously relevant, the variety of adopted **communication protocols** should be taken in account. However, the convergence toward IP based protocols, which could support interoperability at various level of the automation stack, from field to cloud, is nowadays a reality so no longer a matter of opinion. Nevertheless, as several higher-level protocols have been mentioned in the QU4LITY questionnaire, they should be considered. It is worth to notice that one of the main qualifying aspects, which lead to the adoption of a certain protocol, depends on the choice between synchronous or asynchronous communication. Demanding interoperability between these distinct dialogue approaches is not always possible, as the underlying design decisions still pertain to the semantic layer of the information exchange or, event worst, they depend on the behaviour of the collaborating subsystems along the flowing of time.

OPEN APIs adopted

With the aim to support the development and diffusion of more and more Open APIs, in particular, in the field of client-server services and broker-based interaction, some communities of developers started working on the implementation of specifications to facilitate the portability and maintainability of existent APIs, regardless their specific implementation details. Two of the most relevant results in this direction are the AsyncAPI specification [asyncapi], and the OpenAPI Specification (OAS) [openapi].

The OpenAPI specification defines a standard, language agnostic, interface specification for service-oriented APIs, which allows both humans and computers to discover and to understand the capabilities of a service without requiring access to source code, additional documentation, or inspection of network traffic. By defining an API by means of the OAS, a consumer can understand and interact with the remote service with a minimal amount of implementation logic.

AsyncAPI is an open-source initiative that seeks to improve the current state of Event-Driven Architectures (EDA), whose long-term goal is to reach a compatible representation of EDAs and REST APIs, aimed to support the automated management of protocol specifications.
AsyncAPI derives from the OpenAPI specification, aiming at extending its synchronous REST based orientation to the world of asynchronous protocols. Compatibility and interoperability between the two specifications is by design.

Both the initiatives are hosted by the Linux Foundation and supported by important actors in the business and technological world, a condition that ensure continuous development and general acceptance.
While OpenAPI and AsyncAPI are growing in parallel, easing their interoperability keeps being a primary design goal.

While OpenAPI is expressly designed to support service-oriented architecture, the AsyncAPI Specification aims to describe and document message-driven APIs. It is protocol-agnostic, so you can use it for APIs that work over any protocol (e.g., AMQP, MQTT, WebSockets, Kafka, STOMP, HTTP, Mercure, etc.).

OpenAPI or AsyncAPI definitions can be used by documentation generation tools to display the API, code generation tools to generate servers and clients in various programming languages, testing tools, and many other use cases.
They help in the automation of high-quality documentation and ready to reuse code generation. Other applications are API management, testing, and monitoring. They support the entire development cycle of service-oriented or event-driven architecture by providing a language for describing the interfaces of the systems, regardless of the underlying technology.

In order to demonstrate the effectiveness of the suggested approach, a QU4LITY pilot was selected as a workbench for experimentation, with the aim of supplying a proof-of-concept of the applied methodology.

Such a system is being developed by different partner, each developing its own specialized subsystem, all of them interconnected by a dense exchange of messages.
As the development team is composed by independent parties, each adopting its internal DevOps methodologies, it lacked a common Open API at the beginning of the project. This condition offered the opportunity to adopt the proposed QU4LITY ZDM data standard. While the pilot development is still in progress as of the writing of this document, the mentioned Open API specifications were successfully adopted as integral components of the team's DevOps environment, so enhancing its continuous integration and documentation process.

[asyncapi] "AsyncAPI Specification," [Online]. Available: https://www.asyncapi.com/docs/specifications/v2.2.0

[openapi] OPENAPI initiative, "The OpenAPI Specification," [Online]. Available: https://www.openapis.org/

[soap] W3C, "SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)," [Online]. Available: https://www.w3.org/TR/soap12/

[idl] Object Management Group, "Interface Definition Language," [Online]. Available: https://www.omg.org/spec/IDL

6 Conclusion

What you have seen here is a collection of technological approaches for open autonomous quality services engineering and processes within the QU4LITY project.

For analysis, predictive and prescriptive decision support you have learned about necessary building blocks for multi-domain simulation framework in multi-stage zero defect manufacturing towards model-based evaluation of the current or future status of quality and zero-defect manufacturing KPIs. The multi-domain simulation framework delivers the necessary interfaces based on these functional blocks, to get quality services within the framework architecture as services, operating and exchanging data with the data space layer and API.

The domain of adaptive digital shopfloor automation includes functionalities supporting automated monitor and control of Assets and Smart Products in the physical world, contributing to the Automation and Control domain of QU4LITY. Automation and Control domain includes functionalities enabling efficient and reliable data exchange and intelligent control over the physical production processes and assets. You have got an insight in a toolset for adaptive digital shopfloor automation tools and technologies in QU4LITY.

To reach the goals of ZDM and AQ considering complex supply chains and information networks we intended to leverage the standards of the International Data Space (IDS) and extend it with a component for autonomous data management. Specifically, two IDS entities were implemented in a concrete usage scenario: the IDS Dataspace Connector (DSC) and the Data App. The Data App realizes autonomous data management by ensuring a high-quality standard in data transfers. For a simple integration in an IDS Environment the Data App can exchange data directly with the DSC.

Finally, we approached the challenge of identifying guidelines for the adoption of common Open APIs, which could support the communication between the varieties of heterogeneous components composing a ZDM system. The OpenAPI specification defines a standard, language agnostic, interface specification for service-oriented APIs, which allows both, humans and computers, to discover and to understand the capabilities of a service without requiring access to source code, additional documentation, or inspection of network traffic.

These four topics are picked to give a selective overview towards open autonomous quality services engineering and processes. Of course, there are other aspects like user-centric ZDM, distributed process co-engineering and Augmented Reality or integrated services engineering and augmented operations management.

Join the Digital Factory Alliance: https://digitalfactoryalliance.eu/join-us/

Document prepared by
Digital Factory
Alliance