DIGITAL MANUFACTURING PLATFORMS FOR CONNECTED SMART FACTORIES

D3.6 BigData and Analytics Infrastructure (Final Version)

Deliverable Id :	**D3.6**
Deliverable Name :	**BigData and Analytics Infrastructure**
Status :	**Final**
Dissemination Level :	**PU**
Due date of deliverable :	**31/03/2021**
Actual submission date :	**20/04/2021**
Work Package :	**WP3**
Organization name of lead contractor for this deliverable :	**INTRASOFT**
Author(s):	**John Soldatos, Nikos Kefalakis**
Partner(s) contributing:	**FAGOR, IKER, ENG, LKS, ATLAS, FHG-ILT, VTT, TNO, TUDO, MGEP**

Abstract: This deliverable presents the BigData and Artificial Intelligence (AI) Enablers of the QU4LITY project, including their prototype implementation. It illustrates how these enablers can be used to solve general Zero Defect Manufacturing and Quality Management problems, including the problems of the QU4LITY pilots.

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

www.QU4LITY-project.eu

Contents

HISTORY

Version	Date	Modification reason	Modified by
0.1	05/10/2020	Initial Table of Contents shared with WP leader and contributors	INTRA
0.12	23/10/2020	Initial inputs on OpenVA	VTT
0.13	30/10/2020	Updates to the structure of the document	ATLAS
0.14	30/10/2020	Updates to the structure of the document	INTRA, MON
0.15	17/11/2020	Initial inputs on the TUDO/SIEMENS ML System	TUDO
0.16	19/11/2020	Updates to Section 1 – Purpose and Scope	INTRA
0.17	20/11/2020	Updates to Section 1 – Details on the Methodology	INTRA
0.18	04/12/2020	Inputs on QARMA Enabler – Addition of Documentation and User Guide	INTRA
0.19	14/12/2020	Inputs from ATLAS (RUL Estimation, Fault Identification)	ATLAS
0.20	15/12/2020	Inputs on Image Processing/Analysis for Detection of Anomalies in Quality Control	TNO
0.21	05/01/2021	Updates to QARMA Contributions	INTRA
0.22	11/01/2021	Introduction of New Chapter with the Main Requirements and ZDM use cases covered by the ML/AI enablers of the project	INTRA
0.23	12/02/2021	Incorporation of updates	ATLAS
0.24	21/02/2021	Updates to OpenVA	VTT
0.25	03/03/2021	Inputs on the FAGOR-FALINK	IKERLAN, MON
0.26	04/03/2021	Various Updates, First version of Conclusions	INTRA
0.27	10/03/2021	Updates to Section 4	FHG-ILT
0.28	16/03/2021	Updates to Section 3 and 4 on ikcloud+ and Machine learning for Anomaly Detection	FHG-ILT
0.29	16/03/2021	Final Updates on SPI Inputs (section 4)	TUDO
0.30	17/03/2021	Inputs to Section 2; Various Updates and Fine Tuning	INTRA
0.31	18/03/2021	Updates to the DataCrop platform in Section 3.2	INTRA
0.32	18/03/2021	Updates and Fine-Tuning of the Deliverable	INTRA
0.34	24/03/2021	Updated Inputs in Chapter 4	IKERLAN, ATLAS, INTRA
0.35	26/03/2021	Updates to Section 2	INTRA
0.36	26/03/2021	Preparation of version for Quality Control	INTRA
0.36	29/03/2021	Executive Summary Updates	INTRA

0.38	31/03/2021	Revisions following Quality Review by JSI (Section 3, Section 4)	INTRA
0.39	31/03/2021	Revisions following Quality Review by JSI (Conclusions, Executive Summary)	INTRA
0.99	31/03/2021	Fine Tuning	INTRA
1.0	31/03/2021	Version delivered to the Quality Manager	INTRA
1.1	08/04/2021	Comments from Quality Manager	UNP
2.0	12/04/2021	Final Version Addressing Comments from the Quality Manager	INTRA

Executive Summary

QU4LITY has specified and implement a novel approach to quality management and Zero Defects Manufacturing (ZDM), namely an intelligent and Autonomous Quality (AQ) paradigm. The realization of these paradigm relies on the combination of a variety of predictive, preventive, and reactive control strategies, which are implemented at both the product and the process levels. The implementation of these strategies is based on cutting edge digital technologies such as cloud computing, BigData, Industrial Internet of Things and Artificial Intelligence. QU4LITY introduces a novel approach to the implementation of quality management solutions based on the above-listed digital technologies. Specifically, it leverages a set of reusable digital components as building blocks of integrated solutions, which adhere to the structuring principles of the QU4LITY reference architecture. In the context of the QU4LITY project, there building blocks are conveniently called digital enablers. QU4LITY digital enablers span various technologies, including BigData and Artificial Intelligence (AI) technologies. The latter are integral elements of most quality management and ZDM solutions, as the latter are data intensive.

The present deliverable is devoted to the description of the prototype implementation and of the demonstrators of BigData and AI digital enablers of the QU4LITY project. Specifically, the deliverable presents the protype implementation of:

- **Three BigData and Industrial Internet of Things (IIoT) platforms**. These platforms provide the means for collecting and managing large volumes of quality data from production lines, including data streams with high ingestion rates. They also support functionalities such as analytics and visualization of ZDM and quality data. In QU4LITY they have been deployed and used for the on-line collection, management and visualization of industrial data in the context of the project's pilots. Specifically, they have been used to support pilot deployments in sites and production lines that did not have readily available data collection and data management platforms.
- **Seven Machine Learning (ML) systems for quality management**. These systems cover different ZDM and quality management use cases and tasks, including calculation of assets' Remaining Useful Life (RUL), failure detection, failure identification, testing of products for defects, identification of process parameters that must be avoided and more. As such these systems can be used to implement various maintenance, logistics and process control mechanisms, as part of wider and holistic ZDM strategies. From an ML perspective, the presented systems employ a wide array of machine learning techniques, including deep learning, classical supervised machine learning, rules mining, model based reinforcement learning, and unsupervised learning. The models employed combine state of the art approaches (e.g., popular deep learning techniques), and novel approaches developed by the project's partners (i.e., home-grown approaches). Overall, the QU4LITY ML systems (i.e. digital enablers) provide a very representative coverage of ML techniques that can be used for ZDM and

quality management, while extending the state of the art. In many cases they outperform conventional approaches and yield exceptional performance.

The above listed categories of digital enablers are combined and integrated in the scope of the QU4LITY pilots. Specifically, the IIoT platforms are used to facilitate on-line data collection and BigData management in the pilots, while enable the execution of ML enablers on top of them. The integration of the project's IIoT platforms with the ML enablers in various pilots leads to a multiplicative benefit. Nevertheless, each of the listed enablers is also individually exploitable e.g., the presented ML enablers do not rely on a specific IIoT platform for their operation, but rather are usable over any infrastructures that can provide them proper industrial data for the training and the operation of ML algorithms.

The digital enablers of this deliverable are generally reusable and configurable. This reusability has been already demonstrated in the scope of the QU4LITY open calls (i.e., the 1st Open Call of the project), where several of the enablers have been made available and are already used by the open call winners. The partners will take advantage of this reusability to support subsequent Open Call cycles (i.e., the 2nd Open Call of the project).

Apart from being reusable, most of the components and enablers that are presented in this deliverable are individual exploitable. As such they are among the main exploitable assets of the project, which will be advanced and used by the project partners following the end of the QU4LITY project. In this context, they will be also included, documented and promoted in the scope of the QU4LITY market platform in WP8.

1. Introduction

1.1 Scope and Purpose of the Deliverable

One of the main objectives of the QU4LITY project is to provide advanced data-driven, digitally enabled mechanisms for Zero Defect Manufacturing (ZDM) and Quality Management in Production Lines. In this direction, the digital manufacturing solutions of the project, including the solutions that are deployed in the project's pilots, provide the means for collecting and analyzing large amounts of production data towards deriving insights into defects, failures and other production quality issues. This is also in-line with the structure of the QU4LITY solutions, which is specified as part of the reference architecture of the project i.e., the QU4LITY-RA.

The practical implementation of the QU4LITY data-driven solutions for digital quality management and ZDM is based on a range of different BigData infrastructures, including:

- BigData platforms enabling the collection, storage, persistence, management, and visualization of quality-related production data at scale. These platforms facilitate scalable data-driven deployments of digital quality management systems.
- Data mining and data analytics algorithms enabling the extraction of knowledge and insights on factors that influence the quality of the production processes. These algorithms include various Machine Learning (ML) techniques, such as Deep Learning (DL) techniques.

These BigData infrastructures (i.e., platforms, algorithms) are reusable across different ZDM and quality management scenarios. Furthermore, they are modular and can be configured and integrated in different solutions in-line with the QU4LITY-RA. This is the reason why they are classified as Digital Enablers for Quality Management and ZDM.

The purpose of this deliverable is to present the final version of the BigData digital enablers of the project, including their practical prototype implementation and validation in production lines. The deliverable is classified as demonstrator and documents multiple demonstrators that correspond to different BigData platforms and analytics algorithms. Each of the BigData/Data-Driven demonstrators is described in terms of:

- **A business and manufacturing perspective**. This is practically a comprehensive description of the production quality problems solved by the demonstrator. It includes for example the identification of production conditions that lead to defects, or the extraction of knowledge about production parameters that improve production quality.
- **A digital technology perspective**. This is a technical description of the underlying digital technology of each demonstrator. It includes for example a description of the statistics or machine learning techniques used, as well as the data processing building blocks (e.g., connectors, data filtering elements, data

visualization elements) that comprise the demonstrator. Likewise, the IT elements that enable the reuse of the enabler (e.g., its Application Programming Interface (API)) are described as well.

- **A demonstration perspective**. This comprises a description of the demonstrator in the context of some pragmatic ZDM scenario in a manufacturing production line. In most cases the presented demonstrators have been integrated, deployed, and validated in the scope of the QU4LITY pilots in WP7. In the scope of the deliverable, the deployment and use the various demonstrators is provided. Nevertheless, the present deliverable avoids replicating information that is already provided in WP7 deliverables of the project: The goal is to present the implemented demonstrator without delving into the details of the pilot and its validation. In several cases, digital enablers are validated in the scope of the project’s Open Calls i.e. in scenarios other than the project’s pilots.

The present deliverable extends and deepens the contents of deliverable D3.5, which provided an initial report on the BigData technologies to be used in the QU4LITY project. Specifically, D3.5 provided an initial description of the approaches that are employed for data-driven ZDM in QU4LITY. It outlines the main data platforms and algorithms that would be used in QU4LITY. D3.6 extends D3.5 in the following complementary directions:

- **It strengthens the business description of each enabler**, through generalizing its use in ZDM scenarios. The present deliverable outlines the use cases where each of the enablers is applicable.
- **It deepens the technical description of each enabler.** In general, the enablers’ descriptions in this deliverable are richer and more comprehensive that the corresponding descriptions in D3.5. Furthermore, in several cases, the present deliverable updates or revises earlier descriptions to better reflect the final implementation choices with respect to the enablers.
- **It describes a demonstrator for each enabler**. The present deliverable is of a “Demonstrator” nature. As such it describes a demonstrator of each enabler, including relevant visualizations like block diagrams and snapshots (e.g., images) from the actual deployments in a production line.

1.2 Methodology

The overall methodology for the specification, implementation, and validation of the BigData enablers of the QU4LITY has been presented in D3.5 and is revisited in Figure 1. The first phase of the work involves the specification and documentation of the main BigData approaches to be employed in the project. The specifications were led by the QU4LITY use cases, as well as by the QU4LITY-RA. This led to the production of D3.5. Building on these specifications the partners have been actually implemented, integrated and validated BigData platforms and algorithms for ZDM scenarios. The respective implementation and validation work led to the demonstrators that are described in the present deliverable.

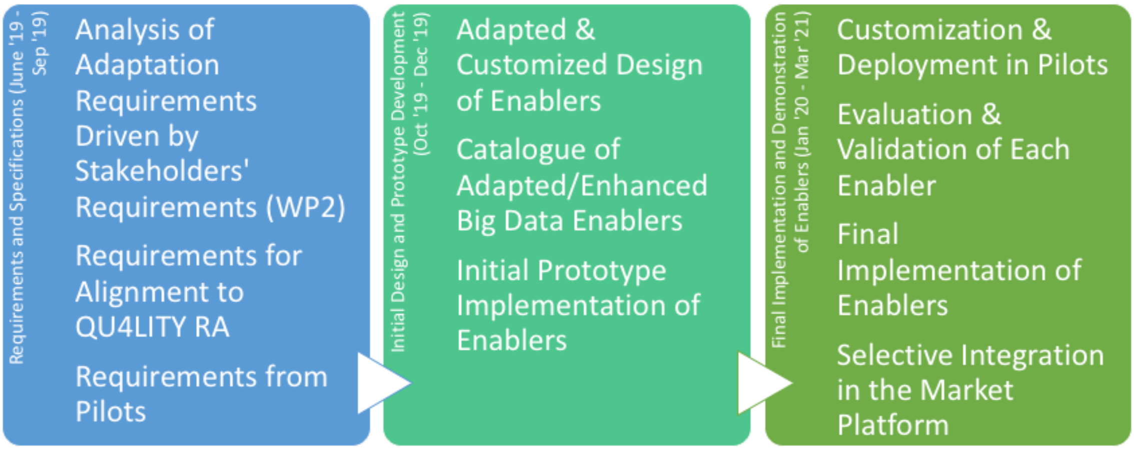

Figure 1: Methodology for Deliverables D3.5 & D3.6

Overall, the transition from the specification phase to the implementation, deployment and validation phase for each demonstrator, involved the following activities:

- **Initial Technical Specification of the Enabler (indicative timelime M13-M15)**: Initial technical specifications of each enabler in implementation detail (e.g., APIs specification).
- **Initial Implementation, Testing and Validation of the Enabler (indicative timeline M15-M18)**:Initial validation of the enabler based on requirements and artifacts (e.g., datasets) from the production line (TRL>=4-5).
- **Fine-Tuning of the Enabler and Deployment in the Production Line (indicative timeline M19-M22)**: Improvements and refinements to the structure, design and implementation of each enabler.
- **Packaging and Generalization of the Enabler (indicative timelime M20-M25)**: Generalization of the enabler and packaging of the technical component to facilitate its reuse and distribution. As part of this activity, each enabler/demonstrator was also integrated in the market platform of the project (in WP8) (TRL>=5-6).
- **Preparation and Validation of Final Demonstrator (indicative timelime M24-M27)**: Fine-tune of the implementation of each demonstrator, along with its final validation in real-life operational settings (e.g., production lines of the pilots and/or the open calls) (TRL>=5-6).

1.3 Relation Deliverable D3.5 (Version 1)

As already outlined, D3.5 was a report with the main specifications of the various digital enablers. Rather, the present deliverable (D3.6) describes the implementation of the actual demonstrators with reference on their deployment and validation in real-life ZDM scenarios. Moreover, leveraging on the experiences from the actual deployment and validation of each enabler, the present deliverables generalizes their use in other ZDM and quality management scenarios as well. Likewise, their technical implementation is generalized and presented in terms of reusable elements such as Application Programming Interfaces (APIs).

Note that this deliverable presents the implementation and demonstration of the enablers that were specified in D3.5. It includes some enablers (i.e. BigData/AI platforms and ML algorithms) that were not detailed in D3.5, yet implemented during the last year of the project's lifetime. Overall, D3.6 refers to a richer set of enablers that those introduced in D3.5.

1.4 Relation to Other Deliverables

The present deliverable is closely related to various other QU4LITY documents/deliverables:

- **D2.1/D2.2 Analysis of User Stories and Stakeholders' Requirements (Version 1 & Final Version).** These deliverables provided requirements that boosted the design and specification of the various enablers.
- **D2.3/D2.4 Autonomous Quality Vision for ZDM and Quality Management Excellence (Version 1 & Final Version),** which provided information about how the BigData components of the project could be developed/customized to support the Autonomous Quality paradigm of the project.
- **D2.11/D2.12 Reference Architecture and Blueprints (Version 1 & Final Version),** given that the demonstrators of this deliverable are deployed in-line with the RA of the project and in-line with relevant solution blueprints specified in deliverable D2.11/D2.12.
- **D3.3/D3.4 HPC and Cloud Resources for ZDM (Version 1 & Final Version),** given that some of the components and demonstrators of this deliverable are cloud-based.
- **D3.7/D3.8 Fog Nodes and Edge Gateways for ZDM deployments (Version 1 & Final Version),** since some of the algorithmic enablers of the project are deployed in fog/edge nodes, while some of the platforms that are outlined in this deliverable adhere to the edge/fog computing paradigm.
- **D7.1/D7.2 Detailed Pilot Specification and Report on Pilot Sites Preparation (Version 1 & Final Version),** which describe the various pilots' where the BigData enablers of this deliverable are actually deployed and validated.
- **D7.3/D7.4 Zero Defects Machines (Version 1 & Final Version) and D7.5/D7.6 Zero Defects Processes Pilots (Version 1 & Final Version),** which detail the pilot systems where the enablers of the present deliverable are used. Note that this deliverable refers to the pilot systems, and in some cases presents performance results derived in the pilots. However, the present deliverable is focused on the technical, technological and function description of the BigData and Analytics digital enablers, not on the description of the pilot systems per se.

There are other related deliverables as well, but the above-listed ones are the most closely linked to D3.6.

1.5 Deliverable Structure

The rest of the deliverable is structured as follows:

- Section 2 illustrates the requirements that have driven the development of these enablers. Emphasis is paid in illustrating the ZDM and Quality Management problems addressed by the analytics enablers (i.e., the ML/DL/AI components) of this deliverable. In conjunction with the presented IIoT and BigData platforms, these enablers boost the autonomous quality vision of the project.
- Section 3 following this introduction presents the BigData and IIoT (Industrial Internet of Things) platforms that are used as Digital Enablers in QU4LITY ZDM scenarios. The platforms support different phases of a data pipeline for ZDM, including data collection, data preparation, data management and data visualization.
- Section 4 illustrates the Data Mining techniques for ZDM Scenarios, which has been designed, implemented, and validated as digital enablers in the QU4LITY project. They comprise various ML/DL algorithms and support a variety of ZDM use cases such as calculation of the RUL (Remaining Useful Life) for machines, detection of failure conditions, identification of production parameters that lead to defects, detection of defected parts and more. As part of the section, the business, the technical and the demonstration aspects of each enabler are presented.
- Section 5 is the final and concluding section of the deliverable. It summarizes the BigData enablers of the QU4LITY project and their use within and beyond QU4LITY scenarios.

2. Driving Requirements and Reference Use Cases

2.1 Driving Requirements for IIoT Platforms Supporting ZDM Use Cases

QU4LITY scenarios for Quality Management and Zero Defect Manufacturing are based on the collection and processing of digital data to enable proactive identification of potential defects and initiation of actions for preventing or mitigating the situations that lead to the quality issues. Typically, the detection of such abnormal situations requires the collection and analysis of large volumes of data that enable the development and operation of deep learning systems. As evident in latter section where the machine learning enablers of the project are presented, most of the algorithms used fall in the realm of deep learning to leverage the fact that DL algorithms yield much better performance than classical machine learning when large amounts of data are available. Therefore, to support the QU4LITY use cases there is a need for enabling infrastructures that facilitate the management of BigData, including IoT data. In this general context, the BigData and Industrial Internet of Things enablers of the platform must address the following requirements:

- **Management of large volume of data**: In real-life quality management scenarios, there is a need for collecting very large volumes of data to train deep learning algorithms.
- **Management of streaming data**: To facilitate fast detection of abnormal situations (e.g., on-line scenarios) there is also a need for collecting, managing and analyzing data streams with high ingestion rates.
- **Scalable Storage of BigData**: The BigData and IoT enablers should provide support for storing arbitrary volumes of data in scalable ways, based on the establishment of data lakes. This concept is fully in-line with the BigData building blocks of the QU4LITY-RA as described in deliverables D2.11 and D2.12.
- **Data (Streams) routing:** In a typical digital manufacturing scenario, various applications (e.g., Quality and Manufacturing Management (QMM), Computerized Maintenance Management System (CMMS), Enterprise Resource Planning (ERP)) access data streams from diverse Cyber Physical Production Systems (CPPS). Therefore, there is a need for dynamically routing data from CPPS producers to application consumers, which must be supported at the level of IIoT middleware.
- **Data Filtering and Pre-processing**: To economize on bandwidth and storage, digital manufacturing applications for ZDM must support filtering of unwanted information. Likewise, pre-processing of streams is essential to support analytics functions close to the field and in the cloud.
- **Cloud Integration**: Quality management applications need to take advantage of the scalability, elasticity, and quality of service of cloud computing infrastructures. Likewise, they must leverage the cloud to integrate data and services from diverse data management systems and CPPS. Thus, there is need for supporting integration of CPPS to the cloud i.e., ending up implementing ZDM and quality management applications as cloud applications.

- **Configurable Analytics**: ZDM applications can benefit from the ability to configure and program different analytics functions over the digital data. This should be provided as a functionality of the BigData and IIoT platforms that support the QU4LITY applications.
- **ML Support**: The Industrial IoT platforms to be used in QU4LITY should support the development and execution of Machine Learning algorithms. The Digital Enablers listed in Section 4 use all one or more Machine Learning techniques.
- **Digital Models for ZDM and Quality Management**: In support of ZDM use cases, BigData and IoT platforms that comprise digital models for ZDM processes are required. All of the project's platform have been enhanced with such digital models.

The development of the project's BigData platforms (see Section 3) was driven by these requirements. The implemented platforms provide support for the above listed requirements. They bear similarities in their IoT and BigData functionalities, yet they feature differences as well, as they have a different area of focus. Note that the BigData platforms of the project are true enabling infrastructures of ZDM use cases i.e., they can be used to support a variety of different quality management use cases beyond the project's pilots.

2.2 Driving Requirements and Overview of Analytics Algorithms

IoT and BigData platforms are enabling infrastructures for the development, deployment and operation of analytics algorithms, notably of algorithms that extract insights and knowledge on how to support or improve autonomous quality management processes. Based on the type of extracted knowledge and the most common quality management processes, the ML-based techniques for quality management are driven by the need to provide one of more of the following functionalities:

- **RUL (Remaining Useful Life) Calculation:** The calculation of the RUL of an asset provides a foundation for implementing effective preventive and predictive maintenance towards avoiding failures and unplanned downtime [Si11]. Avoiding failures leads to avoidance of defects as well. In most cases, RUL calculation leverages sensor data readings from a machine (or tool) on various parts made before the machine (or tool) breaks towards determining the End of Life for the machine (or tool) on any new (unseen) sensor readings.
- **Fault Detection and Fault Identification**: The goal of these processes aims at identifying the faulty status of a product or process towards taking remedial actions against future failures or defects.
- **Determination of associations between production variables**: The aim of these analysis is to determine associations of input production variables that lead to certain outputs. This knowledge enables the tuning of production variables in ways that avoid faulty or defective outputs.
- **Determination of process parameter settings to avoid**: As part of this analytics prosses a set of values for various process parameters are constrained

towards ensuring that the percentage of products that do not pass quality testing is below a specific threshold.

- **Anomaly Detection:** The aim of anomaly detection processes is to distinguish the products that deviate from normal products leveraging on properties of the products such as their shape, thickness, surface and quality of materials.
- **Product Testing:** The goal of product testing processes is to audit certain characteristics of produces (e.g., volume, geometry) against predefined threshold values provided by the machine vendor.

Zero Defect Manufacturing aims at minimizing product defects based on combination of measures that span logistics, maintenance and process control. Therefore, all of the above functionalities can contribute to the implementation of automated ZDM strategies.

3. QU4LITY Big Data and IIoT Platforms

3.1 Overview of Big Data Platforms

As part of WP3 of the QU4LITY project, three Big Data platforms are deployed and used in various pilots. These platforms leverage the state-of-the-art Big Data infrastructures (e.g., Apache Kafka) and IoT capabilities (e.g., IoT Gateways), yet they also provide added-value features for data pre-processing and analytics. In-line with the QU4LITY Reference Architecture, these platforms serve as hosting environments for the data analytics and machine learning techniques that are described in the following section. Two of these platforms are enhanced versions of background BigData infrastructures of the platforms, which have been customized and/or enhanced to support analytics in ZDM scenarios.

The following table provides an overview of the three BigData analytics infrastructures / platforms that are used in the project, while following paragraphs elaborate on their capabilities and the ways they are customized and used in quality management and ZDM scenarios.

Platform	Description	IP Owner
DataCrop DDA	Supports Configurable Routing, Preprocessing and Analytics of Heterogeneous Data Streams in Industrial Environments	INTRA
Open VA	Supports Analytics and Visualization of IoT Data in Industrial Environments	VTT
ikCloud+ ML Platform	Enhanced version of the ikCloud+ Digital manufacturing platform of IKERLAN that supports ZDM related analytics by means of ML algorihtms	IKERLAN

Table 1: Overview of QU4LITY Big Data Platforms

3.2 DataCrop Platform -Distributed Data Analytics (DDA) Infrastructure

3.2.1 Platform Description – QU4LITY Foreground Developments

The DataCROP (Data Collection Routing & Processing) platform, initially developed in the H2020 FAR-EDGE & PROPHESY projects is an IoT platform consisted of two core components which are the Edge Processor Engine (EPE) and the Distributed Processor Engine (DPE). The Edge Processor Engine, as implied by the name, resides within the Edge Gateways of the infrastructure and is responsible in processing low level data streams collected from one Edge Gateway. The Distributed Processor 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 solution is the Distributed Data Processor (DDP). DDPs are the heart of the Analytics system where DDPs instances are combined from the EPE and DPE to provide a complex data processing solution.

A simplified version of the architecture focusing on the DataCROP interactions is shown in Figure 2. There, at the Edge layer, we can see the Processor-Engine which controls the Processor wrappers "wrappers" of the Processors hiding the complexity of the Management and configuration from the algorithm designer. The Processor-Engine is a configurable component utilizing the Processor Definition (PD), Processor Manifest (PM) and Processor Orchestrator (PO) data models for its configuration and management. The Processor Engines are facilitated by a centralized interface where the different components can be controlled which is called Processor Engine Open API. This Open API is used from the Toolbox Dashboards (see Figure 2) which acts as the configuration User Interface of the DataCROP. One equally important asset of the DataCROP is the DataCROP's Data Models. The Data Models are the heart of the system where the DataCROP data Models instances can be combined from the EPE and DPE to provide a complex Analytics solution. More information related to the data models can be found in deliverable D3.13 "Library of Integrated, Interoperable Digital Enablers (Version 1)" of T3.7.

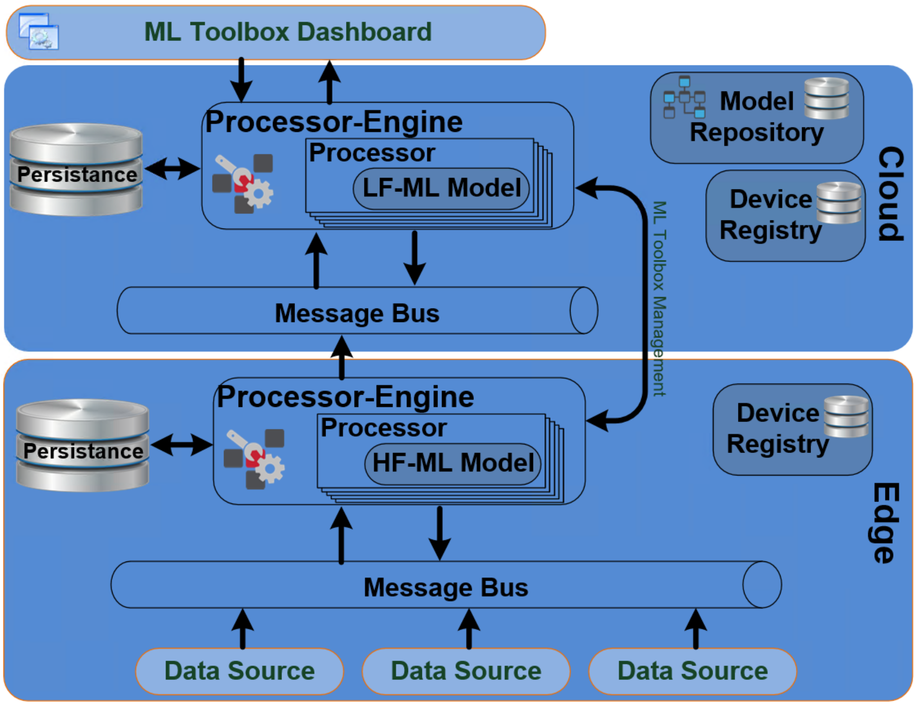

Figure 2: High-Level DataCROP focused functional diagram.

Internally by utilizing the DATACROP Processor Engine design we create an almost completely decoupled set of components, acting as micro-services listening and writing on the DATACROP message bus. This in essence means that the DataCROP is as flexible and extensible as it can be: an algorithm designer wishing to add to this DataCROP their own model for RUL prediction, only need to tailor an existing wrapper

(DDP) to the specific algorithm needs and programming language. The wrapper (DDP) would handle in a standardized way the required interactions with the Processor Engine and the subscription/publishing to the Message Bus. The subscriptions to the appropriate topics in the DataCROP message bus of the message bus would be used to get information as soon as it becomes available, and the publishing is for "writing" the results back to the appropriate topics as soon as they are produced. Figure 3 shows the loosely coupled nature of the DataCROP.

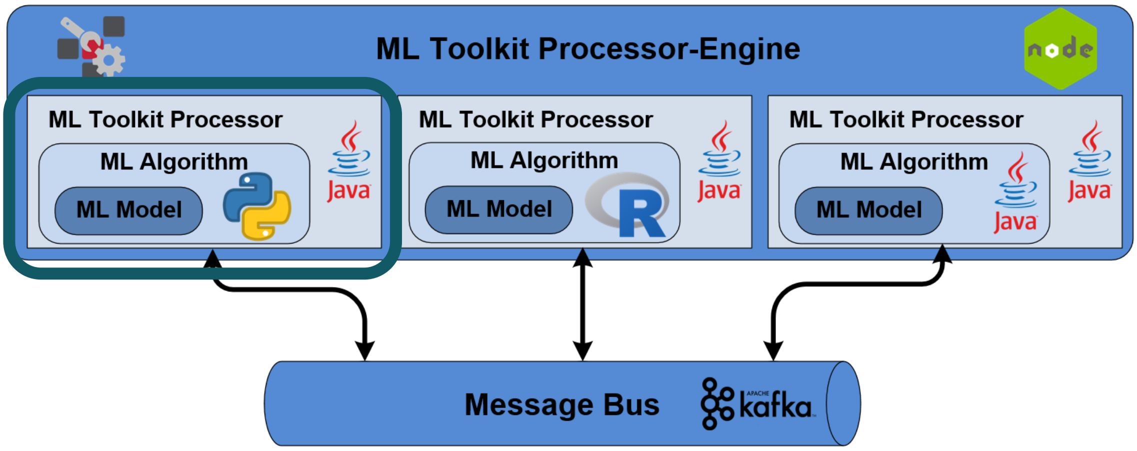

Figure 3: Loose-coupling Nature of DataCROP

3.2.1.1 Adaptation to QU4LITY Pilots

The Processor Manifest defines an analytics task based on a set of data sources and processors. The processors that can be used are limited by the supported Processor Definitions that are available in the model repository. In QU4LITY we have provided new processing functions by implementing new Processor Definitions, for the RIASTONE pilot, and registering them with the model repository. The new Processor Definitions where then have been used in Processor Manifests. Below we can find the steps that have been followed:

1. **Implement Processor Definitions**: Two new processor definitions have been implemented based on the specific interface that is specified by the DataCROP platform. The implementation defines how the processor reads its in-put data, how it processes them, and finally how it writes its output data.
2. **Register Processor Definitions in model repository**: Once the new processor types where implemented, they have been registered to the DataCROP model repository by using the UI offered by the platform that utilizes the exposed Open API. Information about the new processor type is provided in the form of an analytics Processor Definitions. After that, we were able to use it in the Processor Manifests based on the pilot's needs.
3. **Use Processor Definition**: Once a Processor Definition became available, it was used for constructing the Processor Manifests.

As mentioned above, for QU4LITY project DDP variation has been implemented, for an analytic algorithm that is written in Java (QARMA Algorithm) and subsequently introducing it to the wrapper not as an external executable script but as a Java library included in the project as an Apache Maven dependency.

Since the Wrapper is also written in Java (as we can see in Figure 3) this library is automatically be packaged in the Java wrapper JAR file, without the need to place the external algorithm script in a particular directory in the filesystem. The QARMA predictor exported methods available directly from use within the code:

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

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

Figure 4: The QARMA algorithm used in the wrapper as a library

The mission of the QARMA machine learning algorithm is, given a collection of sensor measurements as input to produce a prediction in JSON format. The prediction is then packaged in the "value" property of the outgoing message (of type "Observation") to be returned to Kafka.

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

3.2.2 Deployment and Use in QU4LITY Pilots

In the context of the RiaStone pilot a DataCROP instance was deployed on a VM provided at the pilot's shopfloor. For this specific use case only one Edge-Gateway was used since there was no need of a distributed deployment. The deployment diagram is depicted in Figure 5.

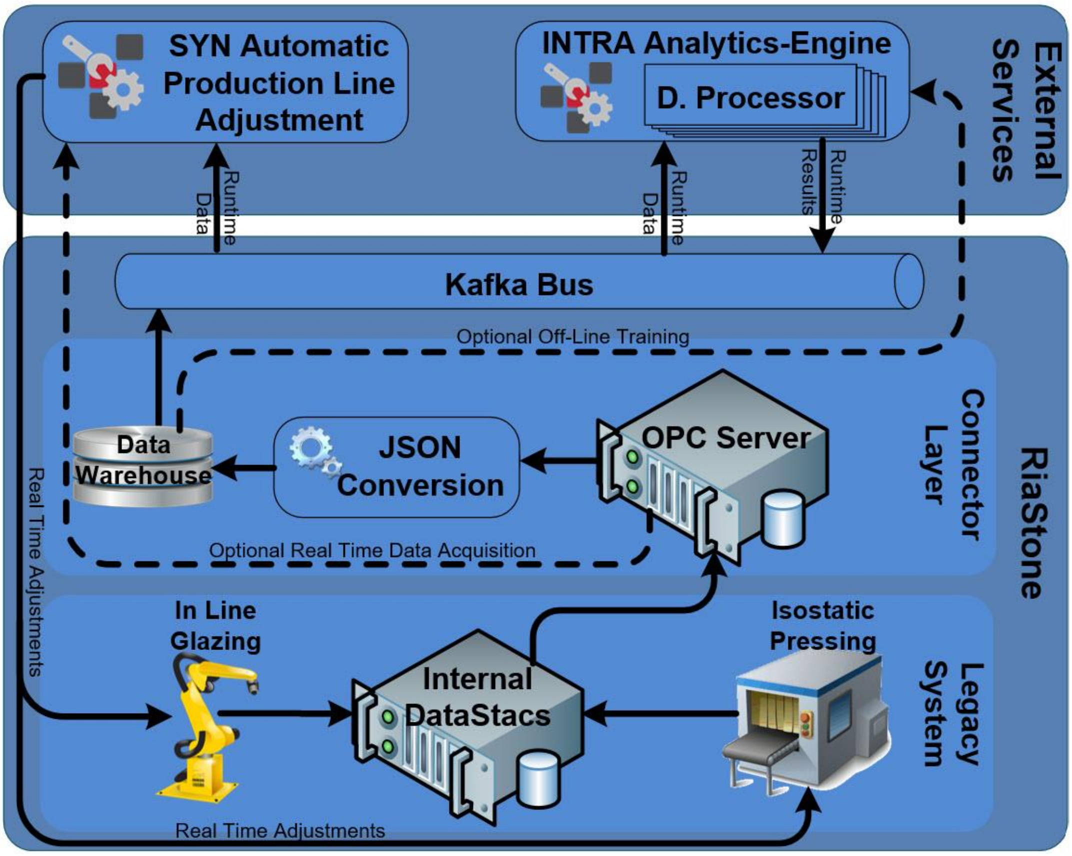

Figure 5: RiaStone Pilot deployment architecture

As we can see the deployment is divided into two logical groups:

- The RiaStone where the already available data collection infrastructure resides and
- The external services where we have the:
  - SYN Automatic Production Line Adjustment service and the
  - INTRA Analytics Engine (DataCROP).

The data are pushed from the Isostatic Pressing and the in-line Glazing machines to an internal DataStacs repository. From there they are pushed to the OPC Server where after they transformed in JSON format they are stored to the RiaStone data warehouse. From there and in the form of data streams they are pushed to the Kafka Bus where, for the runtime, the DataCROP solution subscribes in order to transform them in Observation format (i.e. based on the FAR-EDGE data model that has been detailed in QU4LITY Deliverable D3.13) and feed them to the QARMA algorithm through the Processor engine. Then the results are pushed back to the Kafka Bus (see Figure 5) in order to be used from the applications and services provided from RiaStone. For the offline algorithm training the data can be retrieved from RiaStone's data warehouse or from private sharing methods.

Overall, in the context of the RiaStone pilot, the QARMA4Industy algorithms described in the following section are integrated with the DataCROP solution.

3.2.2.1 DataCROP deployment

The core infrastructure of DataCROP is deployed effortlessly by taking advantage of the facilities offered by Docker. The latter is a set of platform-as-a-service (PaaS) products that uses OS-level virtualization to deliver software in packages called "containers". (https://www.docker.com/)

A kit to be used for said deployment includes the following:

- **INSTRUCTIONS.md**: a text file containing instructions on how to deploy the platform
- **test.yml**: a YAML file containing the configurations of the various Docker containers (Docker images, environment variables, networking, data volumes configuration etc.)
- **processors**: a folder containing sample algorithms imitating the ML toolkit's algorithms behavior.
- **data**: folder containing sample data to prepopulate the databases with for demonstration purposes

By following the instructions, one may both deploy and undeploy the various components DataCROP infrastructure. To do so, Docker Compose is being used. Compose is a tool for defining and running multi-container Docker applications. A YAML file is being employed to configure all application's services. Then, with a single command, one may create and start all the services from the aforementioned configuration. (https://docs.docker.com/compose/)

The two processor samples included in the deployment kit, packaged in JAR format, are:

- **echoer**: an algorithm that receives messages from a Message Bus channel and then redirects them towards another channel without changes
- **predictor**: an algorithm that receives messages from a Message Bus channel, redirects their crucial data to an external algorithm (imitating an actual ML algorithm) and then redirects the result (an assumed RUL calculation) towards another channel.

The processors are coded using the Java programming language and are packed in an archive format known as JAR. The processors are automatically placed in the Docker container representing the DataCROP component, and more specifically in the **root/processors/** folder.

To navigate there one must enter the console of the container. To do so, first we need to find its containerID using the "**docker ps**" command. Then we type "**docker exec -it <container_ID> bash**". This lets us enter the console of the container.

To manually add a processor (i.e. a custom ML algorithm packaged in a JAR), one must either place it in the "processors" folder of the installation kit before deployment, or copy the JAR from the outer operating system into the Edge Analytics Engine container with the command:

docker cp <path_to_jar> <edge_analytics_container_id>:/processors/<path_to_jar>

3.3 OpenVA

3.3.1 Platform Description – QU4LITY Foreground Developments

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.

VTT OpenVA is customized and used in FAGOR/MODRDAGON pilot. The following steps have been undertaken:

- **Identification and modelling of data contents**: The data content to be analyzed are modelled, in co-operation between software developers and data owners.
- **Setting up test cloud server in VTT Azure cloud** that is used during the project.
- **Collecting test data**: Relatively small data set will be collected and uploaded to OpenVA for initial testing.
- **Specification and implementation of data analytics algorithms**: Analytics algorithms are implemented based on data owner requirements. Moreover, the algorithms are tested using test dataset.
- **Customizing of the OpenVA user interface to the needs of the pilot**.
- **Specifying and implementing interfaces for continuous data transfer**.
- **Deploying the service to the server used after the project**.

3.3.2 Deployment and Use in QU4LITY Pilots

At this stage, VTT's OpenVA has been deployed in VTT's Azure cloud for using test data and not integrated to the actual data sources. Test data has been collected and uploaded to OpenVA for initial testing. Existing visualizations have been demonstrated using the test data set (see Figure 6).

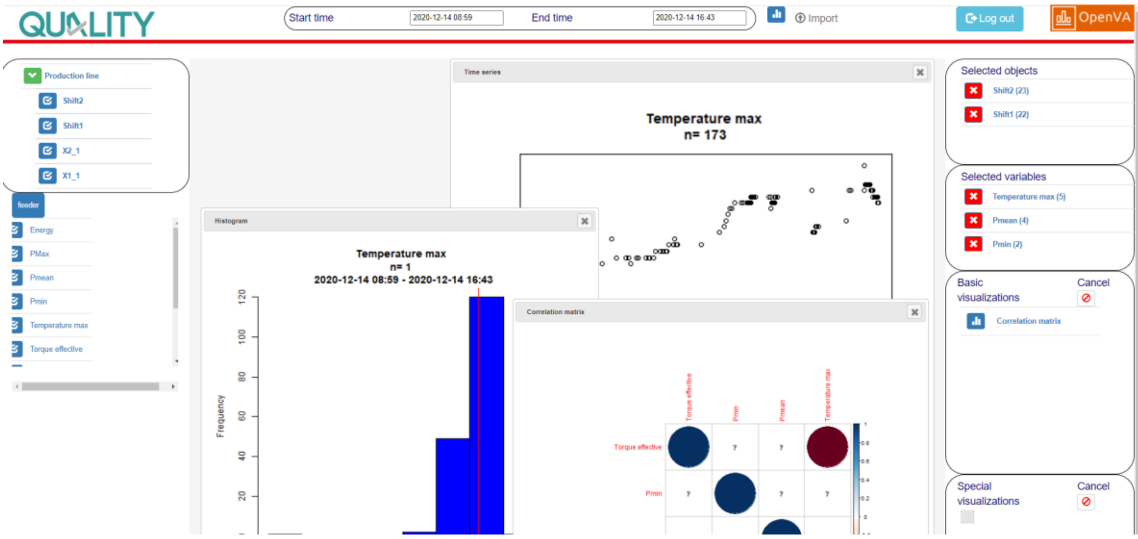

Figure 6: VTT OpenVA user interface using test data

Next steps include the implementation of specific analytics algorithms and integration to real data sources.

The deployment of the application follows the principles shown in Figure 7.

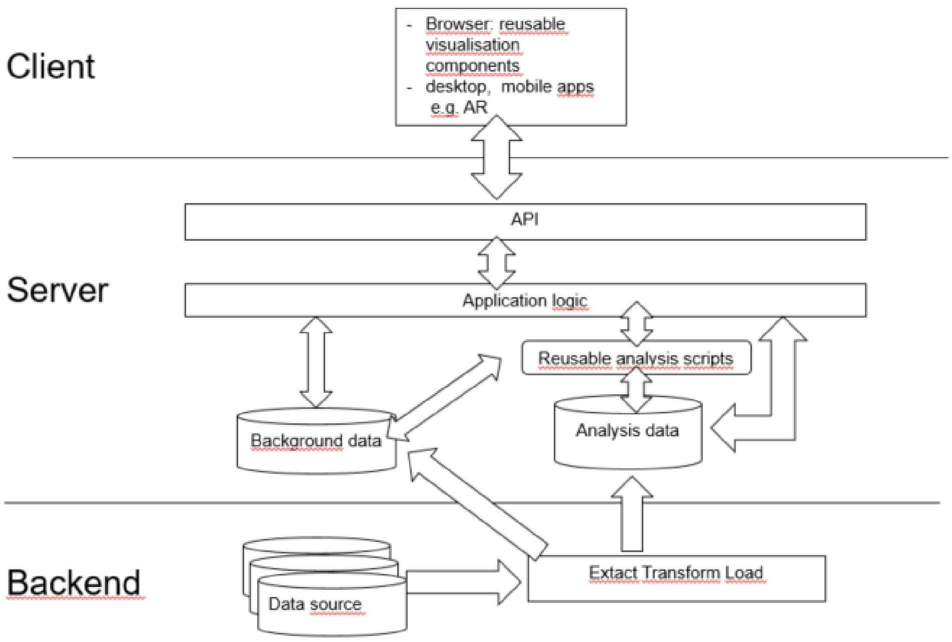

Figure 7: VTT OpenVA deployment architecture

3.3.3 Documentation and APIs

OpenVA is documented and its APIs described along with source code in https://github.com/pekka-siltanen/vttopenva. QU4LITY specific updates and additions (excluding confidential information) will be published in GitHub during the QU4LITY project when the pilot case is finished. A Docker container showing application usage with small set of "anonymized" example data will be also provided.

3.4 ikCloud+ ML Platform

ikCloud+ component aims to be a solution that can be linked into an existing platform in order to aggregate additional valuable information to the existing system. The main value the enabler provides is to add data analytics capabilities to a system that had not such capabilities, by adding different ML tooling managed together by Machine Learning Operations (MLOps) solutions. This way, the MON/IKERLAN pilot of the QU4LITY project is provided with an infrastructure that enables end to end simple model development orchestrated by the enabler. Note that "Anomaly Detection at Machine Level" ML Enabler described in the following section is deployed over the ikCloud+ ML platform that is described in this section.

3.4.1 Platform Description – QU4LITY Foreground Developments

ikCloud+ enhances base Kubeflow features providing a solution that can be plugged into an existing digital platform with minor adjustments. The platform performs an orchestration defining different automated workflows that go from initial data processing, through different model training and finishing in a model evaluation step where the best performing candidate model is selected for deployment. This way, the existing platform could be provided with tools such as malfunction prediction, early maintenance, or error prevention.

Considering that these principles are a must to provide autonomous quality, ikCloud+ becomes valuable component to assist any system to compel with the ZDM principles.

The component makes use of new technologies that have arisen in the recent years providing highly scalable and adaptable solution to deal with actual data analytics problems. The technology stack used, consists of different tooling that provide very flexible behavior for each of the core steps of the system. Thus, leaving the component highly adaptable for any significant scalation of the host system, mainly because all the tools are intended for working in cloud platforms.

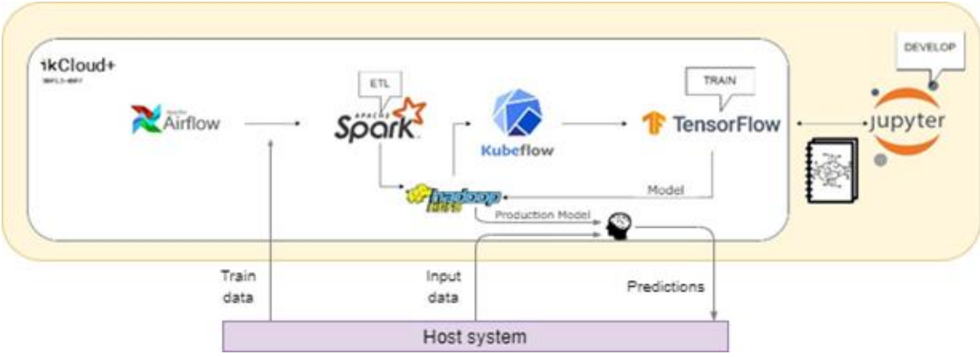

Figure 8: ikCloud+ enabler as a standalone component

The tooling used is composed by Apache Airflow for workflow definition, who orchestrates the periodic ETL. This ETL is then performed by Spark, which is a distributed in memory processing tool that enables endless scalability at the time of data processing. This way, whenever the data processing increases significantly, more processing power can be easily added to the component by adding more Spark workers that share the computation between them. The main goal is to periodically prepare the raw data for analytical tasks, so data is already formatted to train and validate ML models.

The information transformed by Spark is then stored in HDFS, a distributed file system that is built on high availability and resiliency on mind. HDFS acts as the common file system for the whole component, storing train data, intermediate models and deployed models.

Candidate models are defined in Jupyter notebooks, which is backed by Tensorflow, provisioning different model implementations and a ML development framework to the notebooks. Then models defined in Jupyter notebooks are provided to Kubeflow, who then run the defined pipeline in order to train and test the models against the generated datasets in the ETL phase. Additionally, hyper parameter tunning may run on the models that can benefit from this, and finally Kubeflow deploys the best performing model.

IKERLAN ikCloud+ internal overview

ikCloud+ component is a standalone component that can be plugged into a system that provides data to it. Then, the component ingests this train and input data coming from the host system and process it through different steps, finally generating a model for predictions. The main steps are the following:

- Definition of the model in the development stage (DEVELOP).
- Ingestion and transformation of the input data (ETL).
- Training and Validation of the model (TRAIN).
- Deployment of the model for prediction.

The development phase is where different candidate models are defined. Here, a collection of state of the art or advanced model implementation are defined. Afterwards, these models can be introduced in the Kubeflow pipeline, to be trained and tested automatically.

The ETL step (which stands from Extract, Transform, Load) consists of data loading from the source, transforming it in the way of making a cleanup of non-significant variables and data transformation and saving it within the HDFS storage. This step makes the provisioning of training and test data for the different models that are evaluated on the training phase.

The training step trains all the models specified in development phase with the clean data acquired in the ETL. Here data is split between train and test data and previously defined models are then trained and their performance is evaluated. This training usually adapts the model's hyperparameters (as connection weights in neural networks) optimizing it by minimizing the error in the predictions.

Finally, when all the models have been trained and tested, the model providing the best results in the predictions is deployed.

3.4.2 Deployment and Use in QU4LITY Pilots

ikCloud+ component has been deployed in servers hosted by IKERLAN within a virtual machine and linked to the Pilot. Pilot's data has been collected and uploaded to ikCloud+ HDFS file system to perform the tests of the whole defined workflow, starting from Airflow scheduled tasks to Kubeflow's model validation and deployment.

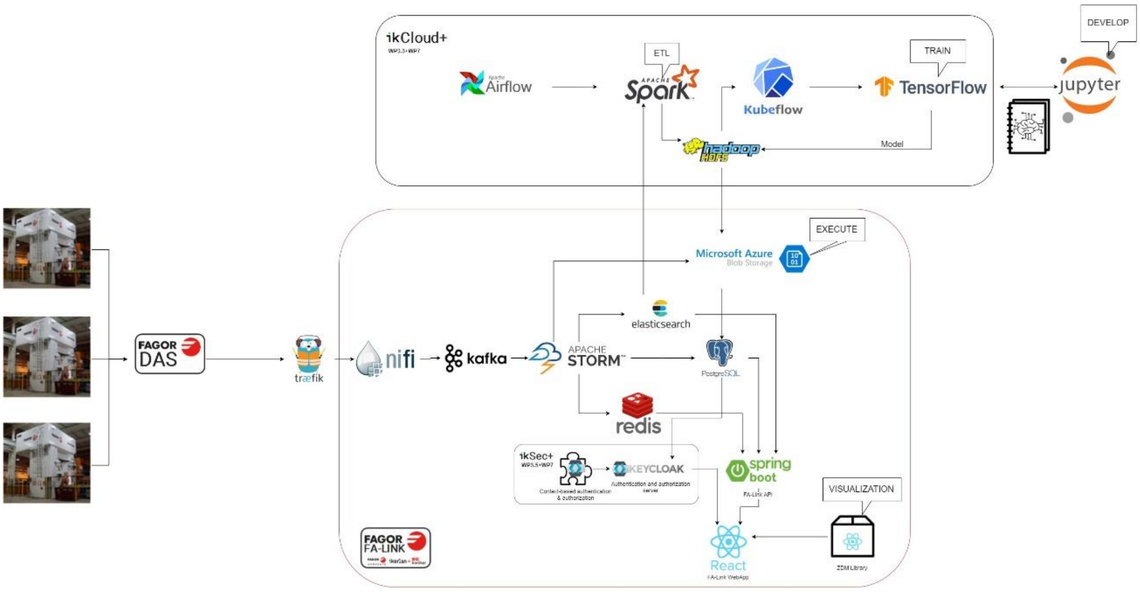

Figure 9: ikCloud+ integration in the Pilot

Although preliminary results have been obtained for one of the die used by the press, the solution still needs to be adapted to cope with all the dies used in the pilot. Further development is being completed in WP7.

In addition to the whole platform, ikCloud+ provides an out of the box Deep Learning solution based on autoencoders that is described in section 4.7

4. QU4LITY Library of Analytics Algorithms

The following table provides an overview of the QU4LITY algorithms for data-driven extraction of Quality Management and ZDM insights.

Algorithms	Description	IP Owner
RUL Calculation	Remaining Useful Life Estimation based on RNN/LSTM Deep Learning Techniques	ATLAS
Failure Detection and Identification	Collection of Supervised and Unsupervised Learning Techniques for Detecting and Identifying Faults in Production Lines	ATLAS
Quantitative Association Rule Mining (QARMA)	Data Mining approach that produces quantifiable rules based on the sets of features that appear frequently together in the training dataset	INTRA
Anomaly Detection in Production	Detection of anomalies in production using repetitively trained auto encoder and unsupervised learning techniques	TNO
Image Analyzer for Surface Inspection	Quality Inspection of using Deep Learning and Model Based Reinforcement Learning	FHG-ILT
Improved Failure Classification in Solder Paste Inspection (SPI)	ML powered Inspection of the volume and geometry of the solder paste against predefined threshold values provided by the machine vendor to identify and classify failures	TUDO
Anomaly Detection at Machine Level	Inspection of Failures using Deep Learning Techniques AutoEncoder Networks	IKERLAN

Table 2: Overview of QU4LITY Analytics Algorithms for ZDM

Following paragraphs provide details about each one of the above algorithms.

4.1 RUL Estimation

4.1.1 Digital Enabler Overview

The software components of the current Digital Enabler focus on the reusability. Although the solutions are developed to satisfy specific pilot requirements, they are implemented using technologies that allow them to be easily deployed in any other use case sharing similar characteristics with the current ones.

A set of data analysis web services and reusable libraries are implemented utilizing Industry 4.0 technologies to enable ZDM. The implemented components are incorporated to a Smart Maintenance Platform (SMP), which offers custom services by enabling or disabling specific web services to address different data analysis challenges.

The selected programming language for the components of the SMP is the C# utilizing .NET Core libraries to enable cross-platform deployment and the ASP.NET framework for the web service development. Each web service offers a separate API implementation, which allows it to communicate with any other software component even outside the SMP. To further increase the ease of deployment Docker technologies are utilized, which guarantee the quality of service under different working environments.

The developed software components presented in Sections 4.1 and 4.2 are part of the SMP offering Remaining Useful Life Estimation, Product Cycle Identification, Fault Detection and Fault Identification capabilities.

4.1.2 The Business and ZDM Perspective

The Smart Maintenance Platform (SMP) is going to be the vehicle to drive solutions for ZDM into the market. In terms of the actual offering, the digital enabler is designed to be reusable in estimating the Remaining Useful Life. One key element that broadens its applicability and reusability, is the possibilities behind the RUL approach, as the tool estimates the time remaining until the next "failure". Each user, each client, can decide what point of deterioration, of degradation, of offset from quality requirements consists a "failure"; what "zero" means to them. Then, the RUL tool can be trained the with appropriate datasets, to meet this point, with the support of the ATLANTIS team. Hence, the use cases that the enabler can support are endless, as endless as the users' equipment may be.

4.1.3 The Technological Perspective

The objective of the RUL component is to estimate the Remaining Useful Lifetime (RUL) of a unit based on run-to-failure historical data. To achieve that it utilizes a type of Recurrent Neural Networks, the Long Short-Term Memory (LSTM) networks.

Recurrent Neural Networks (RNN) are nets that extract temporal dependencies and allow information to persist though loops of identical nodes, creating a chain reaction as it is presented in Figure 10.

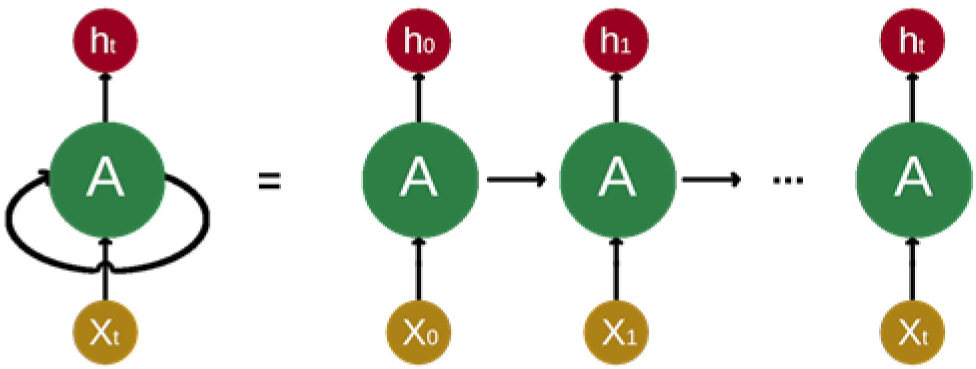

Figure 10: An RNN net as a chain of identical modules

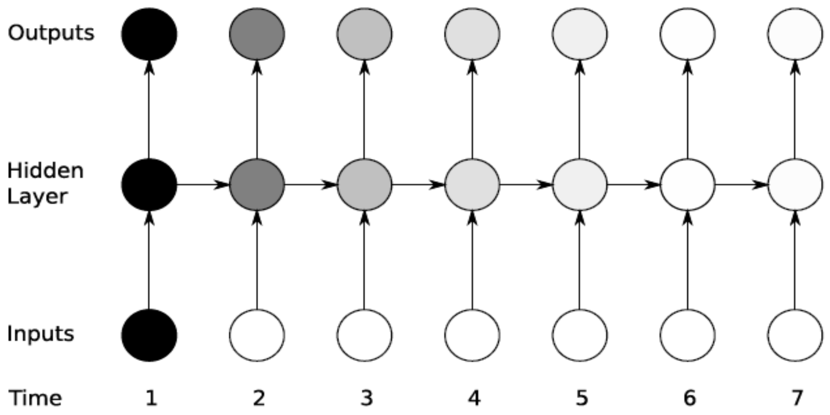

Figure 11: RNN nodes that depict the vanishing gradient phenomenon. The nodes sensitivity to input of Time 1 is depicted with the various degrees of shading. The final nodes do not consider the input of time 1 in any degree.

When there are long term dependencies the phenomenon of vanishing gradient occurs, resulting in potentially insufficient or incorrect computed weights. Figure 11, illustrates the vanishing gradient phenomenon [Akhila18], where the degree of shading denotes the nodes sensitivity to the input. As it is presented, the further the nodes the less the input is considered.

Long Short-Term Memory (LSTM) networks are a special kind of RNN capable of learning long-term dependencies and avoid the vanishing gradient problem utilizing input, output and forget gates. The gates’ usage can enable the safe keeping of information. In Figure 12, the memory cell remembers the previous input from the hidden layer if the forget gate is open and the input gate is closed.

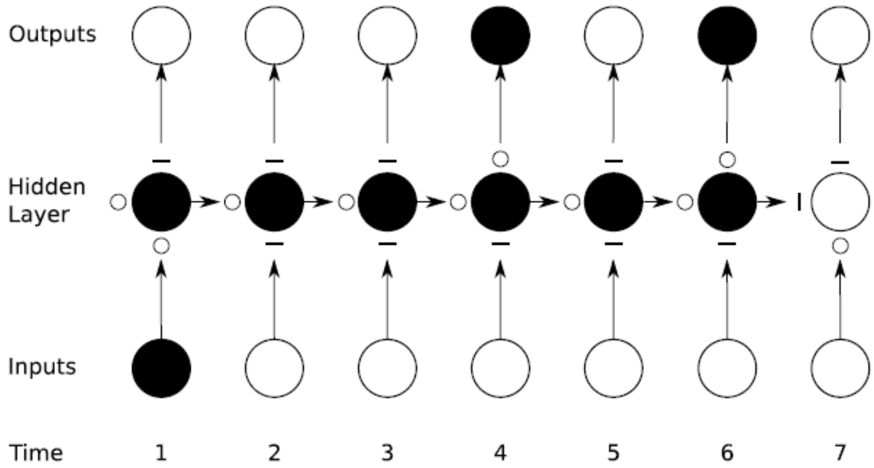

Figure 12: LSTM addressing the vanishing gradient problem. The sensitivity to the input of Time 1 is described with shading levels and the gates' state is described by either 'O' for open or '—' for closed.

After a training phase based on historical run-to-failure data, the RUL module is able to estimate the Remaining Useful Life of the equipment. As it is presented in Figure 13, for the online estimation the module receives incoming sensorial data, pre-process them, forwards them to the trained LSTM model and either save or publish the estimated result.

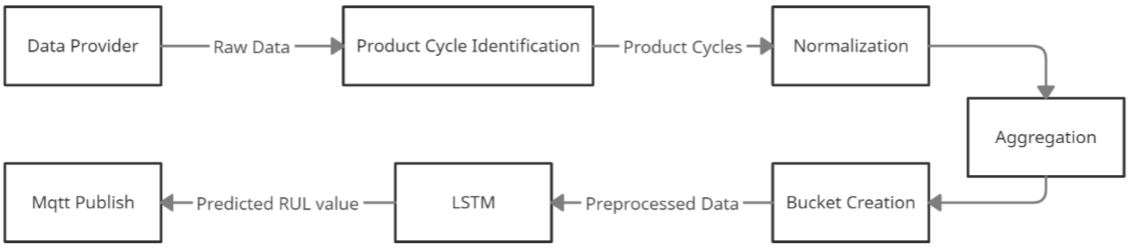

Figure 13: Online estimation of RUL flow

Several pre-processing and transformation steps (i.e. Product Cycle Calculation, Normalization, Aggregation, Bucketing) are applied to the run-to-failure historical data, in order to reduce noise and enhance the quality of the estimation, as presented in Figure 14.

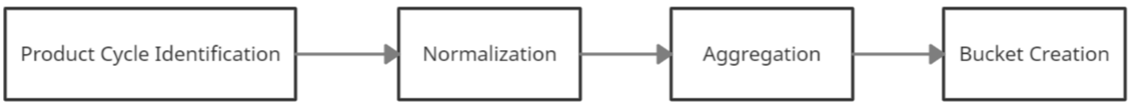

Figure 14: Historical data preprocessing flow

Raw data are categorized in Product Cycles either with a simple condition-based generator or a Machine Learning-based one. The simple generator does not require any prior training and in some scenarios its results are satisfying.

4.1.4 Use in QU4LITY Pilots

The RUL estimation module was applied to the DANOBAT use case to provide a ZDM solution at system level, in order to satisfy customer quality requirements by systematizing quality control and lowering costs. A DANOBAT grinding machine was selected due to its high productivity and constant working conditions. Deviations or variability in the working conditions of the machine, might be indicative of a deterioration of machine condition that could affect the machining process and cause geometry or quality defects, which in turn, cause the need for extensive rework and the increase of the produced scrap. The studied model is a grinding machine, that delivers high accuracy in shape and dimension in cylindrical parts. It is composed of a grinding wheel, that produces the material removal from the surface of the part, and a regulating wheel, that drags the machined part making it spin and advance in the machining direction. The machine is also equipped with one dresser wheel for each of the above-mentioned elements.

Historical data from multiple sources, like the data presented in Figure 15 (i.e. engine intension), are used to train the Machine Learning model in order to estimate the RUL of the grinding wheel. The first step is the product cycle identification and as it is presented in the figure there is a clear pattern on every part production, hence the simple condition-based product cycle generator is used.

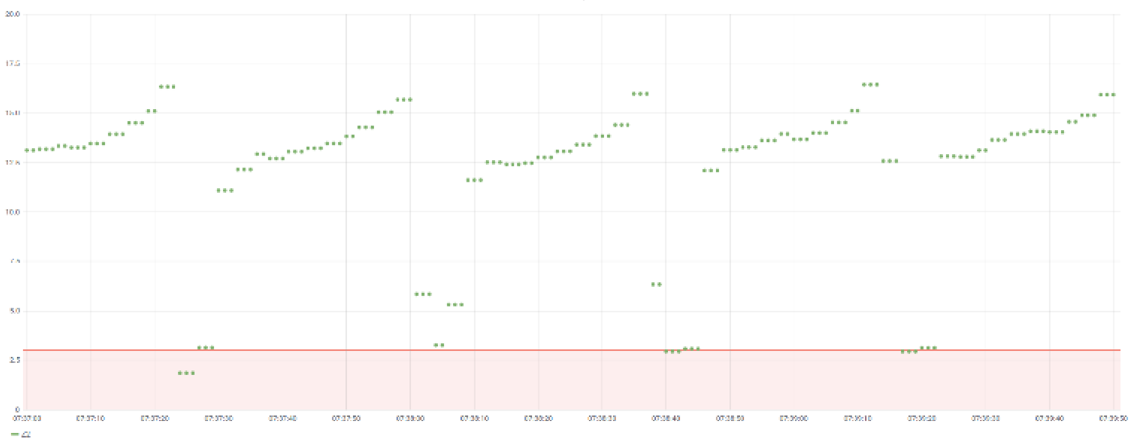

Figure 15: Grinding machine engine intension measurements depict the motif of a Product Cycle in the DANOBAT use case

Each measurement point is either assigned to a specific product cycle number or is dismissed from the LSTM input. This functionality provides simultaneously a way to distinguish the non-significant points of a measurement and an opportunity to later structure the input values based on their product cycles and not their timestamps.

The Product Cycles are preprocessed and transformed to an acceptable by the model format, after being normalized using z-score normalization and being aggregated to their minimum, maximum or mean value based on a bucketing policy. A bucket is a sequence of overlapping sets of product cycles of predefined size.

The preprocessed data are forwarded to a trained LSTM model for the RUL estimation. The architecture of the LSTM network is a double layered LSTM model, where the first layer contains 100 units followed by 20% dropout rate and a second layer of 50 units and 20% dropout rate. The last network layer is a dense output layer of a single unit.

The model uses a linear activation function, the mean squared error metric for the loss function and the mean absolute error metric for the LSTM's performance evaluation and requires training on run-to-failure historical data and as a result is case specific. It is important to note that due to the lack of excess data the same data set was used for testing and training and as a result the evaluation metrics shown below cannot be considered impartial or fully representative.

Once trained the model is loaded in a TensorFlow server (TFX). A TFX is a Google-production-scale machine learning platform, based on TensorFlow. It provides a configuration framework and shared libraries to integrate common components, needed to define, launch and monitor a machine learning system.

The RUL module process is based on asynchronous operations, so that multiple tasks can run simultaneously, each one of them corresponding to a different, if needed or desired, model and input address. The module's predicted values are promoted to different means of storage or communication, such as a database or an MQTT topic.

Figure 16, depicts the estimated (blue line) and the actual (green line) RUL estimation. The estimation follows closely the actual value and providing more data to the model can only increase the precision of the tool.

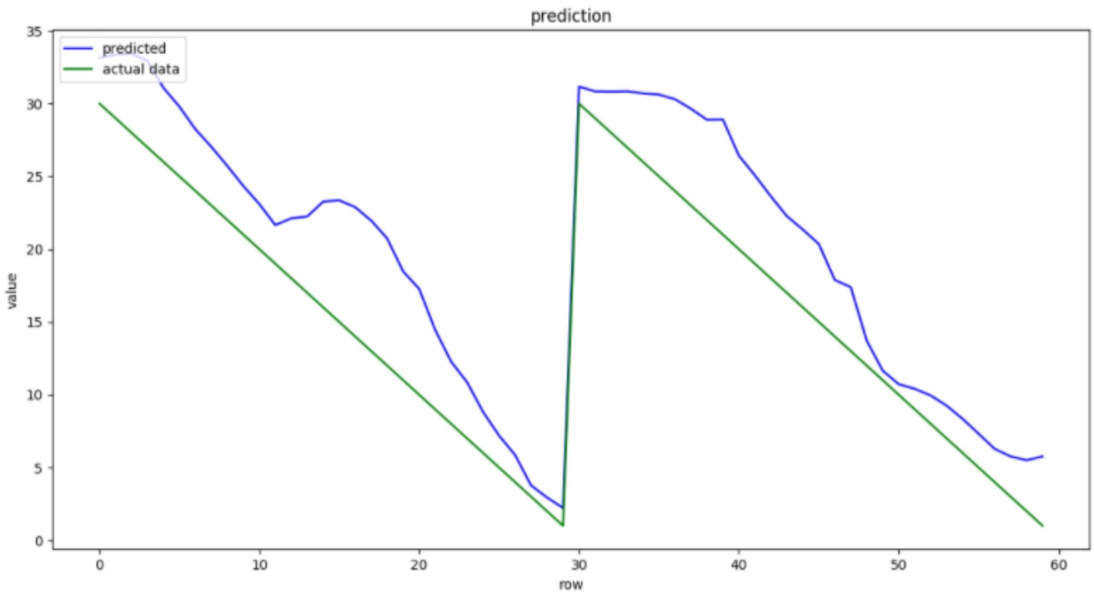

Figure 16: RUL estimation (blue line), alongside with the actual RUL value (green line)

The models' setup and parameterization, together with the assessment of their performance are the first sample of work that DANOBAT has considered. These first outcomes are promising for DANOBAT and they are considered a motivation, a type of proof of concept by the industry.

4.1.5 Demonstrator and User Guide

4.1.5.1 Input Dataset

RUL microservice as mentioned above is utilized for Remaining Useful Lifetime estimation based on run-to-failure historical data. The data are derived from sensor measurements of a DANOBAT's grinding machine. The various measurement types are provided in Table 3.

Table 3: Utilized sensor measurements.

Sensor Measurements (units)		*Description*
1	Engine Intensity (A)	Ampère consumption by the axis engine
2	Load (%)	% of KW consumed by the motor over its rated power
3	Position (mm)	Position of the wheel
4	Power (kW)	Power consumption by the axis
5	Speed (mm/min)	Advance speed of the axis
6	Temperature (°C)	Measured temperature

The analysis was focused on the Engine Intensity values with the frequency of one measurement per second.

4.1.5.2 Input Formats

The RUL microservice is part of the Smart Maintenance Platform and thus utilizes the platform's uniform input and output data format. For the instantiation of a new RUL Task from the microservice a JSON is sent through a HTML POST request which contains the required parametrization. The input sensorial data are derived from an MQTT broker. The parametrization JSON includes the appropriate topic id for the task to subscribe. The RUL microservice parametrization, as depicted in Figure 17, contains the MQTT topic for subscription and the ID of the desired LSTM model. The available models are trained and deployed to a Tensorflow server.

```
InputPropertiesDto ∨ {
    label                   string
                            nullable: true
    topic                   string
                            nullable: true
    modelId                 integer($int32)
}
```

Figure 17: RUL configuration JSON

The sensor values are received and preprocessed through the Data Provider, microservice of the SMP, which is responsible for the data retrieval. Through a custom per pilot plugin the Data Provider receives raw sensor values, identifies the Product Cycles via a Rule-based process, and transforms the incoming values to the SMP's universal data format as depicted in the JSON below. The results are published to the internal to the platform event bus, which is supported by a messaging broker.

```
# Data Provider's Output
{
  "measurements":[
    {
    "name":"measurement",
    "fields":[
      "time",
      "value_1",
      "value_n",
       "pc"
     ],
     "values":[
        [
          1572353787000,
          0.64,
          n_value,
          1
        ]
     ]
    },
    {},
    ...
  ]
  }
```

4.1.5.3 Output

The RUL's estimated values are forwarded to the Reporter microservice of the SMP, which is responsible for the output circulation by utilizing different means of storage or communication, such as a timeseries database or an MQTT broker. The output of the RUL task is displayed in the JSON below, where the Incident attribute depicts the microservice type while the value attribute contains the estimated RUL values.

```
{
       "Incident": "RUL_EVENT"
       "ReportTime": "date", (Date Format YYYY-MM-
DDTHH:mm:ss.SSSZ)
       "IncidentTime": "date", (Date Format YYYY-MM-
DDTHH:mm:ss.SSSZ)
       "Value": "value",
       "TimeAhead": "time_secs", (string)
       "Measurements": ["measurement"],
       "Notes": "null"
}
```

4.1.5.4 Run-time API

For a concise documentation of the RUL API the swagger tool is utilized. The following figures depict an overview of the RUL API and information for each available endpoint. It should be noted that all the endpoints conform to the SMP's API structure.

Figure 18 presents an overview of all the available API endpoints for the RUL microservice. The figures that follow present details for each one of these endpoints.

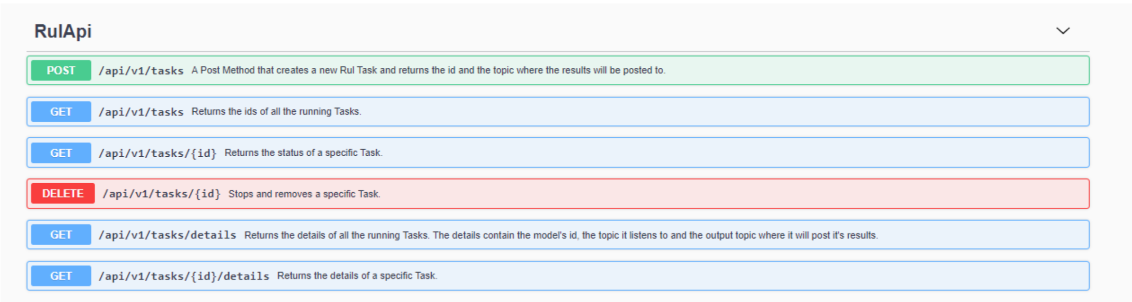

Figure 18: RUL API endpoints overview

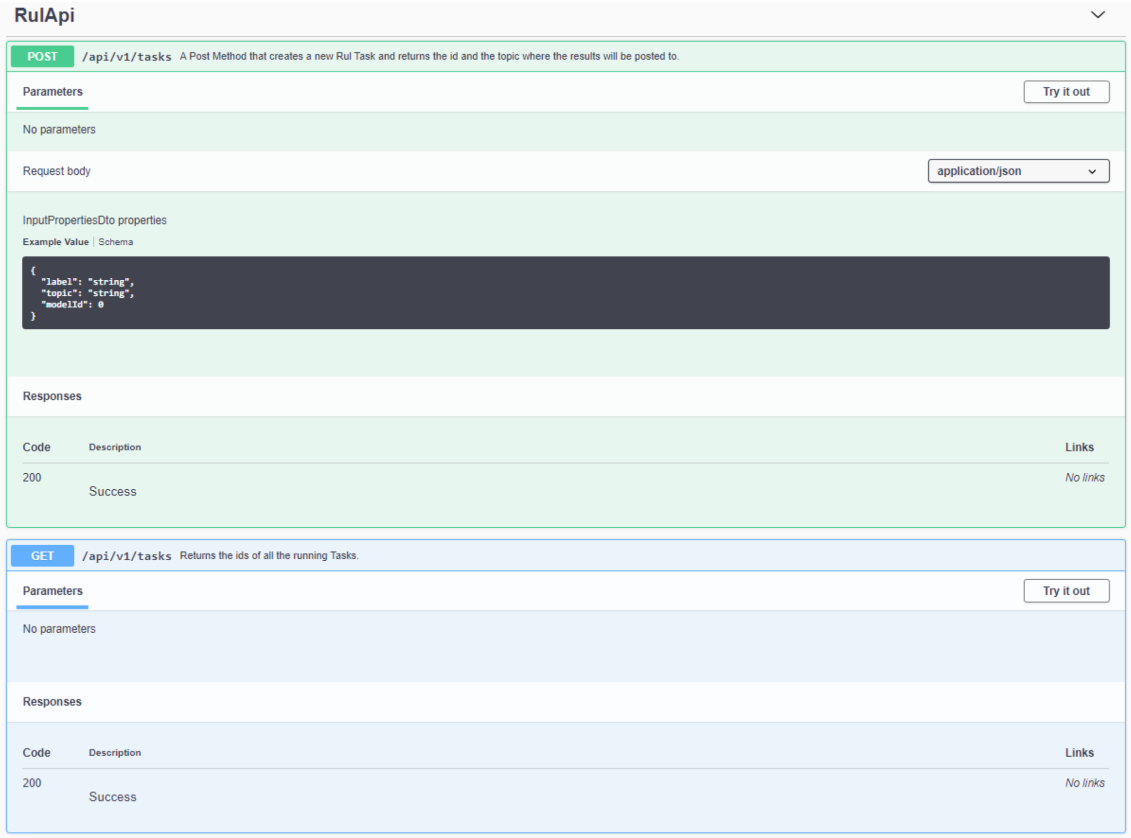

Figure 19: RUL API endpoints to create a new RUL Task (upper figure) or to get the IDs of the running RUL Tasks (lower figure).

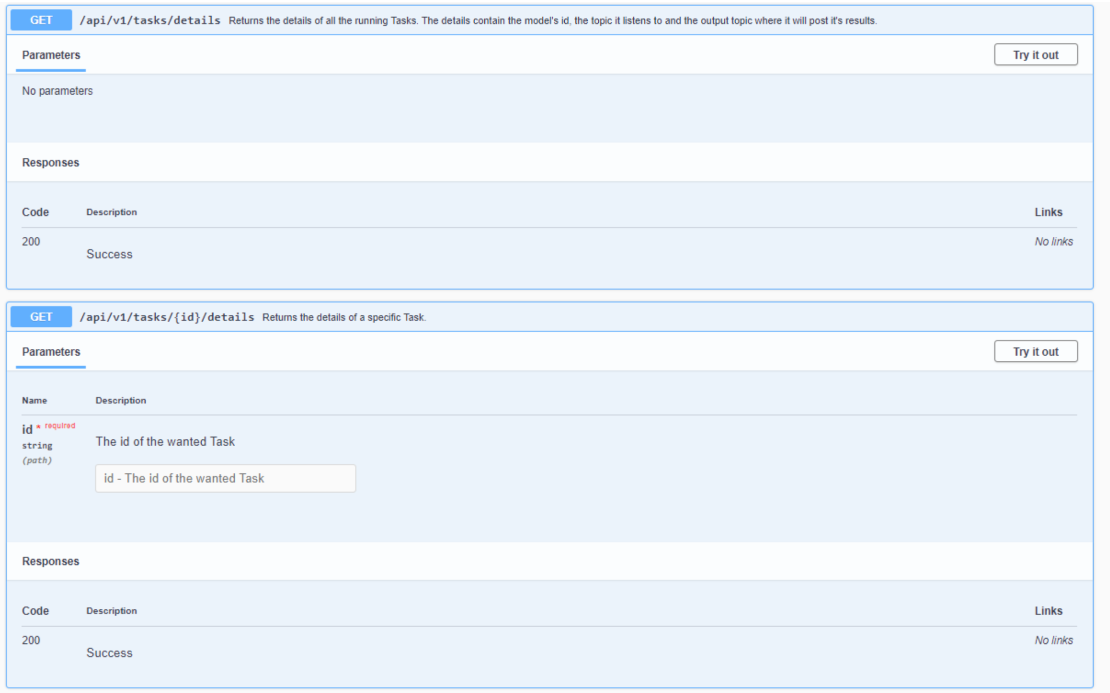

Figure 20: RUL API endpoints to get the details of all (upper figure), or specific RUL tasks (lower figure).

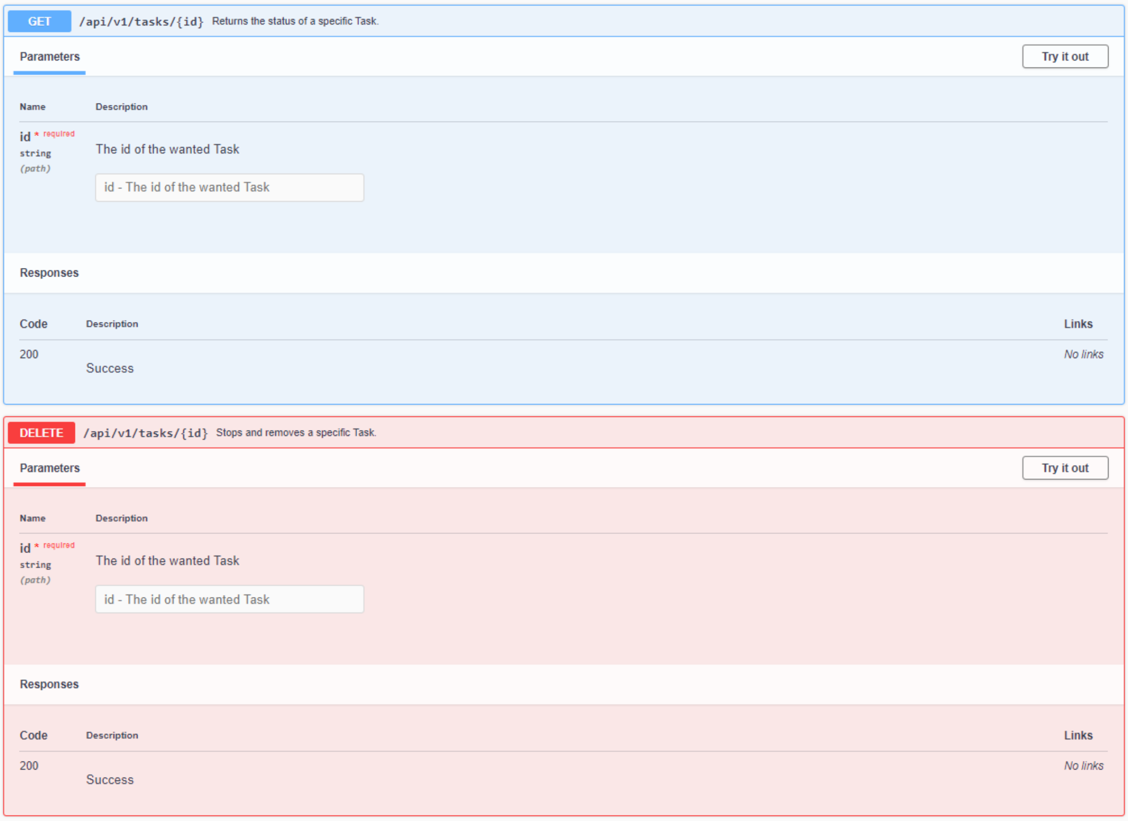

Figure 21: RUL API endpoints to get the status of a RUL Task (upper figure), or to stop and delete a running RUL Task (lower figure).

4.1.5.5 UI

The web interface of the SMP is implemented in the Dashboard module. The microservices of the SMP can operate either through the web interface or through the provided API endpoints. The Dashboard is a web application based on Angular, that incorporates multiple Grafana dashboards to allow the user to monitor the exported results and offers access to the platform's various modules.

Figure 22, presents the Data Provider microservice web interface for the instantiation of the data fetching tasks. Figure 23, depicts the web interface of the RUL microservice, while Figure 24 shows the Status page for the monitoring and control of all the running tasks.

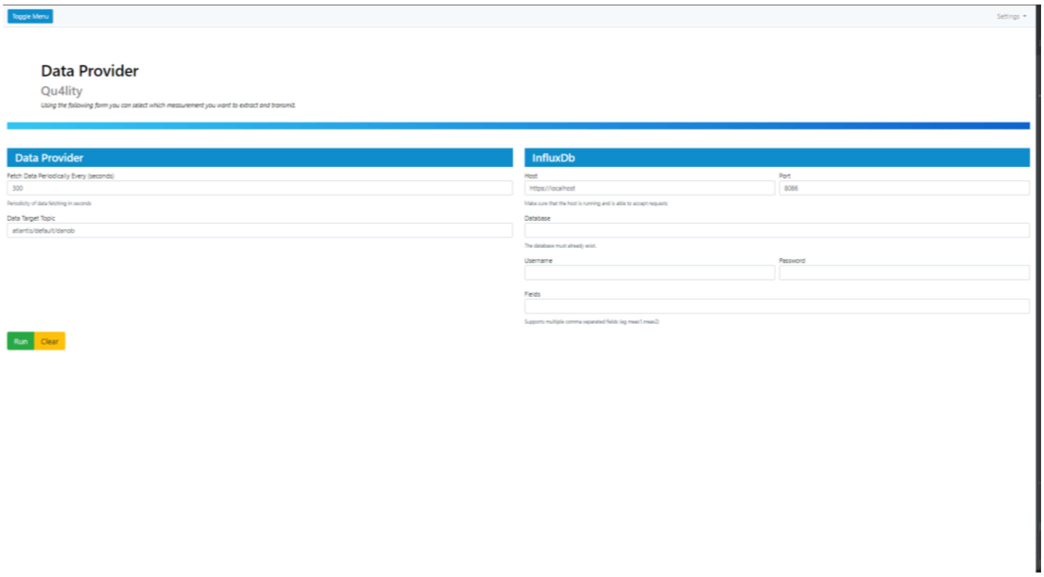

Figure 22: SMP's Dashboard page for Data Provider

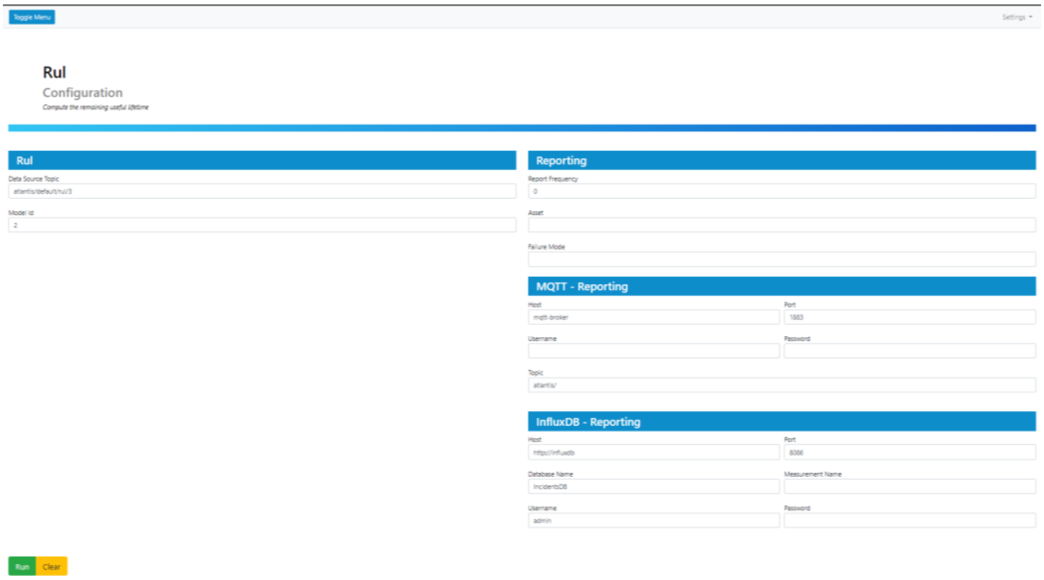

Figure 23: SMP's Dashboard page for RUL

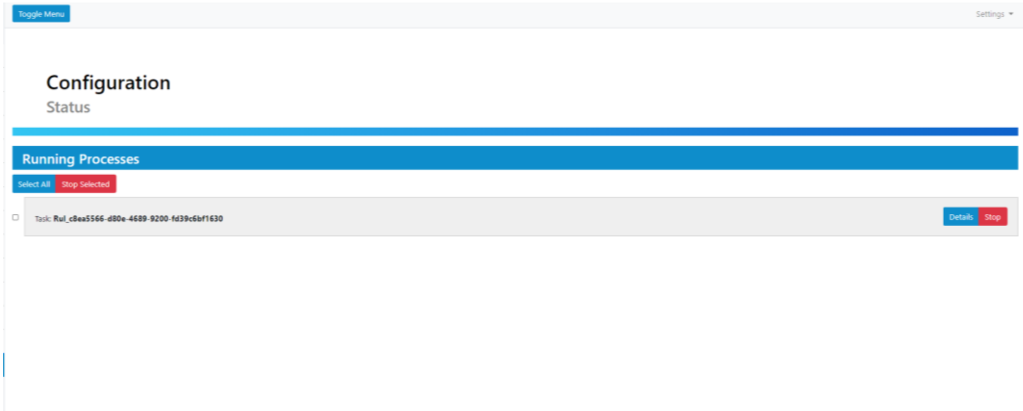

Figure 24: SMP's Dashboard page for the status of a running RUL Task

4.1.6 Use in Open Calls

The RUL digital enabler is one of the selected available even from the first Open Call to be used to validate the QU4LITY concept in areas beyond the pilots of the consortium partners. It has been offered as a dockerized solution, which can be deployed either on premises or on a remote location. Proposers were asked to present ZDM target pilots in-line with the QU4LITY concept and Reference Architecture.

It has been clearly communicated that this solution would be suitable for applicants that have: 1) Real-time sensor data, 2) Historical data, 3) Log files with previous hardware or quality faults (Fault and timestamp), in order to train the algorithms (link sensorial data with faults), 4) Correlation between faults and signals.

Default parametrisations for model training have been provided, however ATLANTIS would remain available for mentoring towards building custom solutions. It should be also mentioned that a generic approach for data fetching has been offered through the tool, however custom data transfer bridges could be developed with any available data handling system install on premises, to provide streams of data to the tool for the online evaluation of the RUL.

4.2 Fault Identification

4.2.1 Digital Enabler Overview

The Fault Identification digital enabler is part of the SMP platform introduced in Section 4.1. It is a separate module, enriching the SMP with fault identification capabilities, using a variety of algorithms, so that the users can select the most appropriate for their case. The Fault Identification module can be used to early detect deterioration and classify upcoming abnormalities depending on severity, thus allowing for better production and maintenance planning.

4.2.2 The Business and ZDM Perspective

In the journey towards ZDM, especially in the first steps, when not a lot of data are available but there is still need for data analysis with a clear value to be produced, one could turn to the Fault Identification Digital Enabler. This SMP module is complementary to the ones more oriented to predictive analytics and benefits from neural networks and classification approaches. With the plethora of techniques it contains, it can address ZDM problems of detecting and identifying early abnormalities for the start.

Taking into consideration the two sub-modules, the Fault Detection can be used from the cold start, without doing any kind of training, in order to detect if something is not working properly in a machine. This can be achieved through the monitoring of appropriate sensorial data that could indicate divergence from the optimal/desired operation level. Moving on, once there are some data collected and faults detected, the next step is to go on to the second sub-module, the Fault Classification. The collected experience can be now used to train a selection of algorithms, in order to classify detected anomalies in the sensorial signal into Failure Modes and to provide better recommendations to mitigate the fault to the maintenance engineers.

The next step, when there is more information collected and analysed, is to take advantage of the RUL Digital Enabler to gravitate more towards predictive maintenance approaches. It is worth mentioning that the Fault Identification is a very useful module of the SMP, because it allows early deployment with a very big value potentially from the get-go.

4.2.3 The Technological Perspective

The Fault Identification module is based on the combination of two sub-modules, the Fault Detection and the Fault Classification, utilizing a variety of supervised and unsupervised techniques. Each detected fault is fed to the fault classification sub-module for further assessment.

4.2.3.1 Fault Detection

For the Fault Detection, the reusable methods developed are based on three unsupervised Machine Learning (ML) techniques i.e., the Matrix Profile[1] (MP) motif discovery algorithm, the Faiss[2] indexing library and Change Point Detection algorithms) and a supervised ML technique based on the Long Short-Term Memory (LSTM) Neural Networks.

Matrix Profile-based Fault Detection

The MP algorithm is a novel, efficient approach for motif discovery and discord detection. It utilizes a sliding window to separate a time series into subsequences and then computes the distance of each subsequence with the entire time series. This element is called Profile Distance and the accumulation of all the Profile Distances

[1] https://www.cs.ucr.edu/~eamonn/MatrixProfile.html
[2] https://github.com/facebookresearch/faiss

provides a final matrix that contains low values in similar areas and high values in areas where an anomaly occurs, as it can be observed in the example of Figure 25. Moreover, the MP implementation offers the Profile Index structure which is an array that relates each point to its closest neighbor.

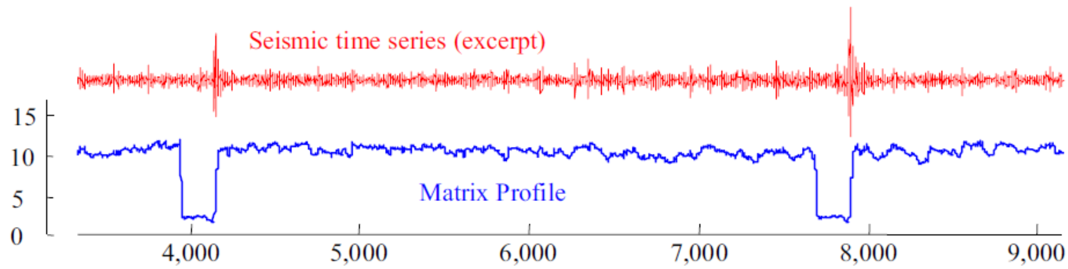

Figure 25: A Matrix Profile example in a seismic motif discovery use case. The similar patterns are depicted through low values in the Matrix Profile.

In order to use the Matrix Profile algorithm for Fault Detection, instead of searching for motifs (the lower values of the MP), the spikes are used to indicate a unique behavior in the signal, which might indicate that an extraordinary event (probably fault) has occurred in the machinery.

Faiss Indexing-based Fault Detection

Faiss is an open-source library developed by Facebook for similarity search and clustering of dense vectors. Given a set of vectors Faiss builds a data structure called index and for every new provided vector performs a search of the minimum Euclidian distance between the two of them. Through this search process, Faiss can calculate the k-th neighbor of its vector.

The Faiss index that is used is a binary flat index that copies data with no further encoding or organization and performs exhaustive search for the Euclidian Distance of the vectors.

For the Fault Detection goal, statistical features (like the mean, median, std, maximum/minimum values) are extracted from product cycle sensorial measurements that we know that represent the normal functionality of the machinery, to form the vectors for the Faiss library. Whenever a new product cycle is identified, the same statistical features are extracted and the closest vector of the index along with its distance is obtained. If the closest' neighbor distance is more than a given threshold a Fault is detected.

Change Point Detection-based Fault Detection

In specific use cases, a sudden change in the sensorial input might indicate a fault. A variety of Change Point Detection (CPD) algorithms are used:

- *Pelt algorithm* is an exact search CPD algorithm that produces an undetermined amount of evenly distributed Change Points. It splits the sequence into parts which

it processes in a serial way while also using a linear penalty to avoid overfitting. The minimum size of each subsequence, the stepping size, the cost function and the penalty value can be defined by the user.
- *Binary Segmentation* is a greedy and loose CPD algorithm with low computational cost. It recursively splits a sequence in two parts in a way that secured the minimum sum of the cost function of the two subsequences. The user can determine the cost function, the minimum size of each subsequence, the stepping size and the desirable amount of detected Change Points.
- *Window Sliding* is a loose CPD algorithm that is based upon the comparison of two neighbors. For each index it compares the deviation of its immediate past, left window, to its immediate future, right window. Once the deviation curve is produced search algorithms are utilized to compute the peaks, where a possible Change Point exists. It can be parametrized for the cost function, the size of the window, the stepping size for each subsequence and the desirable amount of Change Points.

The CPD algorithm are combined with different smoothing techniques in order to enhance their accuracy:

- *Moving Average* is a simple smoothing method that computes the mean value of a subsequence of a sliding window and it corresponds the value in the last index of the equivalent subsequence.
- *Exponential Moving Average* is a weighted extension of the Moving Average method. The weights favor the most recent values and as a result this method represents more accurately the trend of recent data.
- *Savitzky Golay* is a smoothing method based on the polynomial approach of a timeseries. Specifically, for each point and its surrounding window a polynomial approach is computed which in turn provides the new value of the index. For the polynomial computation, the least squares method is utilized.

LSTM-based Fault Detection

Finally, the LSTM networks' which are already presented in Section 4.1, are used for Fault Detection being trained with label sensorial input.

4.2.3.2 Fault Classification

Multiple approaches are developed for the fault classification process, which can either used separately or as an ensemble to enhance the accuracy in some use cases. All the approaches share the same main concept for the classification. Collections of historical sensorial measurements of specific Failure Modes are stored and whenever a new Product Cycle or set of measurements arrives, the distance from all the collections (i.e. Failure Modes) is computed and the lowest one (which is lower than a predefined threshold) is used to select the respective Failure Mode as the class of the new data.

Six different distance measures or approaches are utilized:

1. The Matrix Profile.
2. The Mueen's Algorithm for Similarity Search (MASS algorithm)[3].

[3] https://www.cs.unm.edu/~mueen/FastestSimilaritySearch.html

3. The MPDist[4] measurement.
4. The Dynamic Time Warping (DTW) algorithm [Müller 07].
5. Faiss indexing library with feature selection.
6. A combination of Faiss indexing with the Matrix Profile.

The MASS algorithm, that computes the Distance Profile of a query subsequence of a sequence and the MPDist measurement, which use a Fast Fourier Transformation function, are parts of the Matrix Profile implementation. *DTW* is a classical approach for similarity detection that utilizes stepping patterns and the creation of an optimal warping path whose total distance can be used as a distance measure. For feature selection in combination with the Faiss indexing, the Relief[5] algorithm is used to identify the most important statistical features that are computed from the raw signal, in order to improve the classification accuracy. Relief calculates a feature score that can be applied to rank and pick the most suitable features.

Matrix Profile-based Fault Classification

The Matrix Profile-based method computes a Matrix Profile with either a dynamic optimal window or a predetermined one. The Profile Index of the Matrix Profile is used to pinpoint the starting point of the subsequence that matches the most with the detected fault and the MP distance of that point is used as a comparison measurement.

MPDist-based Fault Classification

The MPDist method calculates directly the MPDist measurement between two time series. The matrix profile is based on the historical sample and the unidentified fault is the query for which the distance is determined. The window size is either determined manually, with CPD algorithms or with a correlation threshold.

MASS-based Fault Classification

The MASS method creates the Profile Distance of the query-the undetermined fault type with the historic sample and provide the minimum found distance as the comparison measurement.

DTW-based Fault Classification

The Dynamic Time Warping technique is developed with open beginning and end alignment and the asymmetric step pattern in order to compute the alignment that matches best with no limits burdening the process. This mode is especially effective for subsequence searching. The stemmed normalized distance is the fault type comparison measurement.

[4] https://matrixprofile.org/tsmp/reference/mpdist.html
[5] https://en.wikipedia.org/wiki/Relief_(feature_selection)

Faiss-based Fault Classification using feature selection

The Faiss method combined with statistical features computes a base index based on each historical sample's features and performs exhaustive Euclidian Search for each new unidentified fault.

Faiss-based Fault Classification combined with Matrix Profile

The Faiss with optimal window selection method uses an optimal window to separate the query into subsequences. The Faiss base vector is computed by the segmented and normalized historical sample in length determined by the optimal window. Each subsequence is then normalized with the min-max method and forwarded into the base vector. The final comparison measurement is the mean distance of all the subsequences with the base vector.

4.2.4 Use in QU4LITY Pilots

The Fault Identification module was applied to the MONDRAGON use case, in order to identify through detection and classification a faulty Product Cycle (referred as plunge), of the wheelhead of a grinding machine. The studied model is a grinding machine, that delivers high accuracy in shape and dimension in cylindrical parts. It is composed of a grinding wheel, that produces the material removal from the surface of the part, and a regulating wheel, that drags the machined part making it spin and advance in the machining direction.

In the grinding process various there are three main Failure Modes that need to be identified:

1. The *Early Beginning*, which signifies the premature contact of the wheelhead with the part.
2. The *Acoustic Signal Distortion*, which corresponds to acoustic anomalies found in the wheelhead's power measurements.
3. The *Vibration Distortion*, which corresponds to unintentional vibration of the wheelhead.

The data analysis tools receive as input various measurements from sensors install on the grinding machine, like the wheelhead's power and speed, the axis' position and torque, etc. The focus is on the wheelhead's power, which is used as the base measurement for all the detection and classification techniques and the X and C axis commanded speed, which are utilized in the preprocessing stage. Figure 26, Figure 27 and Figure 28 presents the effect of the various Failure Models on the wheelhead's power.

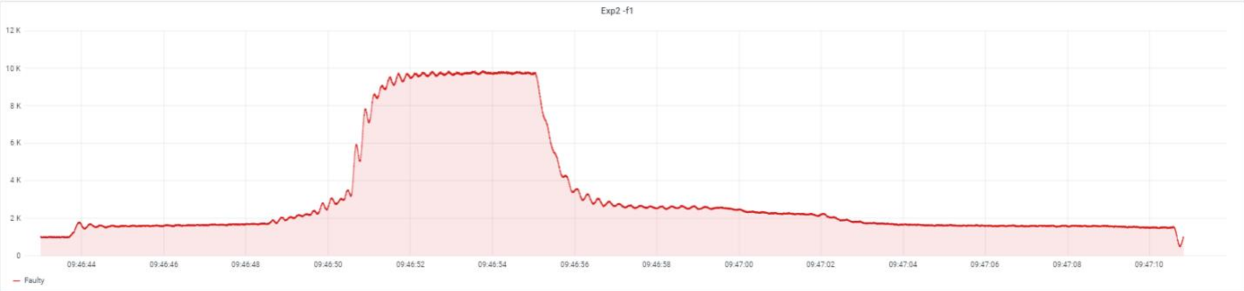

Figure 26: Early Beginning Failure Mode on the Wheelhead power measurement.

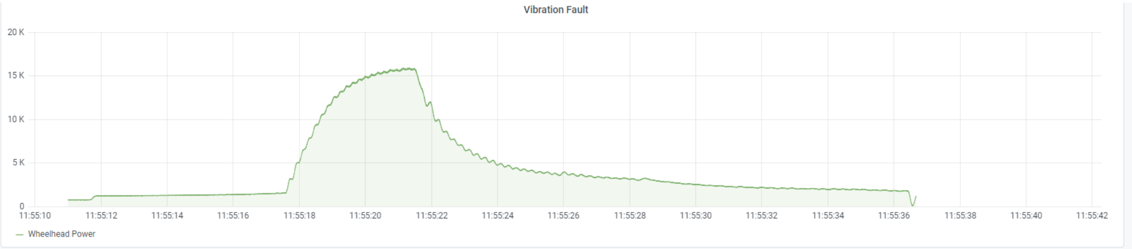

Figure 27: Acoustic Signal Distortion Failure Mode on the Wheelhead power measurement

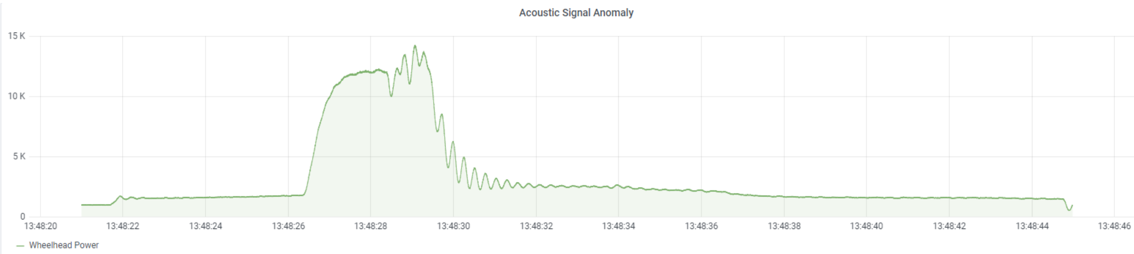

Figure 28: Vibration Distortion Failure Mode on the Wheelhead power measurement

The Fault Identification module receives raw sensor data, preprocess them with a product cycle identification method that is based on pattern matching techniques, trims the data sequence based on the CPD algorithms and filters the remaining values with a rule-based technique. The user is given with a choice to apply either fault detection, fault classification or both. The results are circulated to different means of storage or data broadcasting solutions, such as timeseries databases or MQTT brokers. The data flow in the modules core can be found in Figure 29.

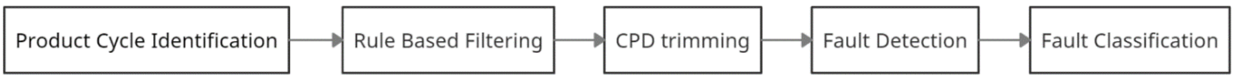

Figure 29: Fault Identification Data Flow

Product Cycle Identification

The Product Cycle Motif-based Identification technique utilizes the MASS algorithm to identify the data points that belong to one product cycle, a vital process for the upcoming stages. The MASS algorithm calculates the Profile Distance between the historic sample of a single product cycle and the incoming data sequence in order to locate the start of a potential match and as a result the beginning of a new product cycle. The Product Cycles that are detected given one product cycle in a historical sample are presented in Figure 30.

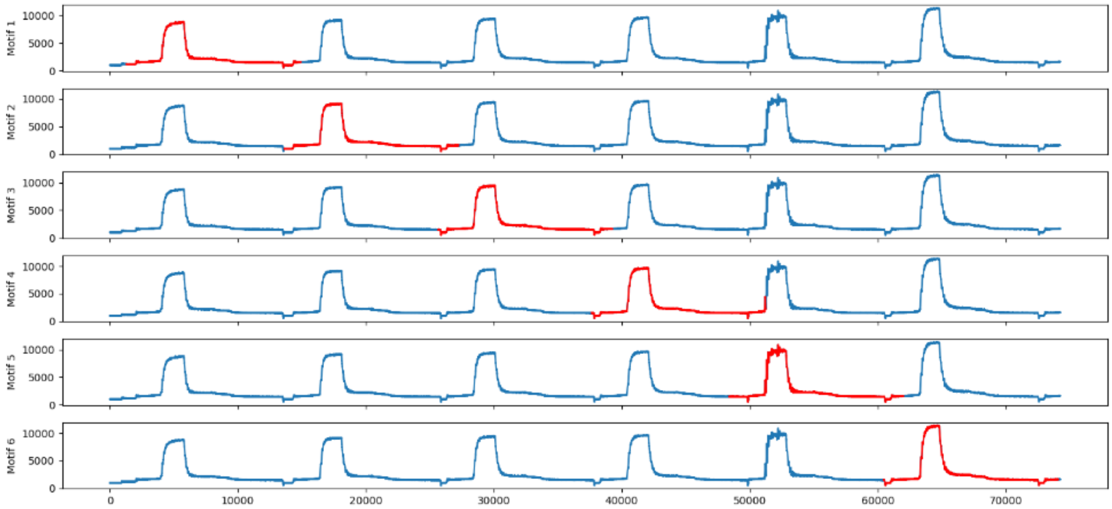

Figure 30: The detected Product Cycles for incoming data with the MASS method. One product cycle was provided as the historical dataset.

Rule-based Filtering

The next preprocessing stage is the diminishment of redundant information and it consists of two parts. The first one is the rule-based filtering that reduces the amount of data by preserving only the desired part of a product cycle while the second one applies the CPD algorithms to determine the starting and ending point for the trimming of the product cycle.

CPD Trimming

For better results the CPD algorithms are tuned alongside the smoothing methods in order to create the finest combination with the most appropriate parameters. For the CPD tuning, a service was developed where the ray python package is utilized to tune all the parameters of all the CPD models and retrieve the one that produces the most accurate results. For the tuning, a manually manufactured list of change points is also given.

Fault Detection

The Fault Detection methods that are developed, focus on the detection of early beginning type of faults. The early beginning type of fault indicates the premature connection of the wheelhead of a grinding machine with the part. In this fault type, as seen in the Figure 26 and Figure 31, the execution curve rises gradually instead of instantly, an attribute that differentiates the specific fault type from the rest while also making it the perfect candidate for Change Point Detection techniques.

Figure 31: The wheelhead power measurement in a normal execution

CPD-based Fault Detection

The CPD based unsupervised method retrieves the most suitable CPD methods with their parameters and calculates early and late change points as well as their difference. The Product Cycles with an early beginning type of fault will have a significant difference while the early and late points in the Product Cycles of a normal operation will be almost undistinguishable, as it is observed in Figure 32.

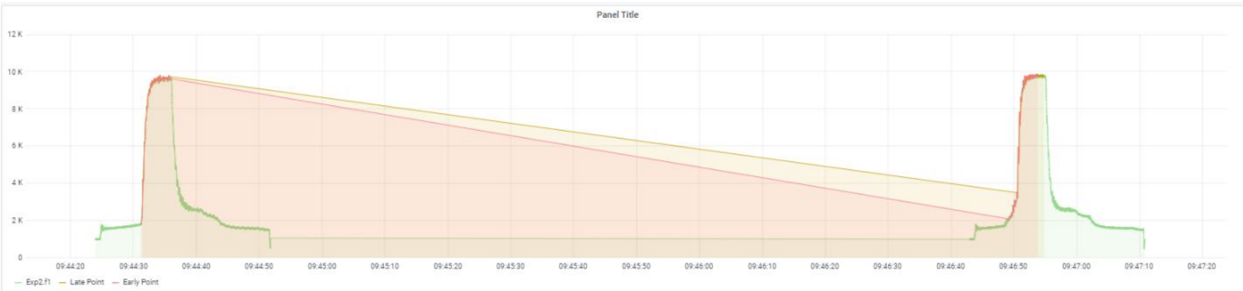

Figure 32: Depicts the difference in the findings of the CPD algorithms for normal and faulty mode of operation

LSTM Fault Detection

The LSTM network's architecture is a double layered model, where the first LSTM layer contains 100 units followed by 20% dropout rate and a second LSTM layer of 50 units with a 20% dropout rate. The last layer is a dense output layer of a single unit. The model uses a linear activation function, the binary cross entropy metric for the loss function and the accuracy metric for the LSTM's performance evaluation.

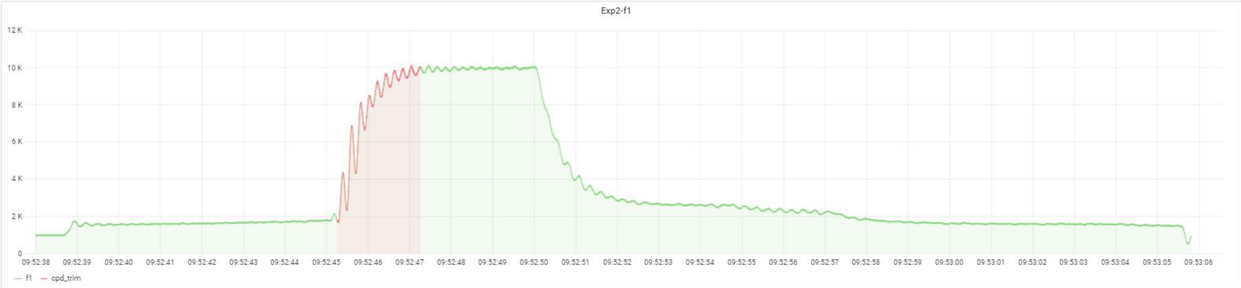

Figure 33: CPD Trimming based on early and later detected change points.

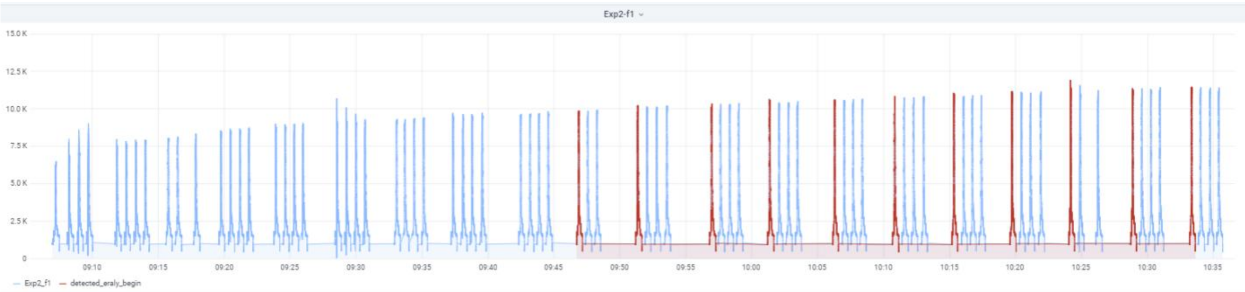

Figure 34: Depicts the LSTM detection results for Early Beginning fault type with the faults as red lines and the normal mode as blue.

The CPD algorithms determine the starting and ending points for the trimming of each product cycle that is then forwarded in the LSTM neurons. This trimming phase, illustrated in Figure 33, reduces any noise or excess points and improves the LSTM's detection capability. The model was trained with a validation split of 0.2 and achieved a F1-score of 1 and the results appear in Figure 34.

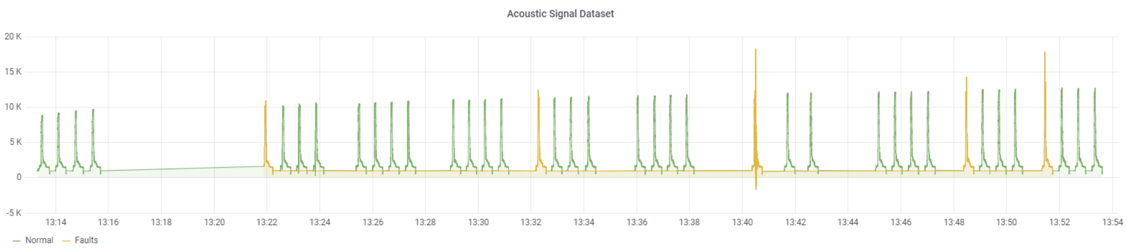

Figure 35: Depicts a dataset that contains acoustic signal type of faults. The faults are depicted with the yellow line and the normal mode with green.

Matrix Profile-based Fault Detection

In the previous sub-section, multiple historical product cycles mapping the normal behavior are provided as samples. The detection results mentioned in the next paragraphs are from a dataset that contains acoustic signal type of faults and other anomalies and is depicted in Figure 35.

The Matrix Profile implementations computes a Matrix Profile based on a historical sample of normal executions and the incoming input. The Matrix Profile window length can be computed either by a predetermined value or by a dynamic selection method. This method computes a Matrix Profiles with the correlation coefficient instead of the Euclidean distance for a variety of window sizes that gradually grow. When the maximum correlation of a Matrix Profile surpasses the correlation threshold the computations stop and the upper window is determined as the window size of the last Matrix Profile.

For the discord detection the Matrix Profile values are compared with a threshold that can be either a user specified value, or a value based on the mean and standard deviation of the data. The detection results for the acoustic signal dataset are depicted in Figure 36. The discords are depicted with a red star in the Matrix Profile diagram. The results can be compared to the ones depicted in Figure 35.

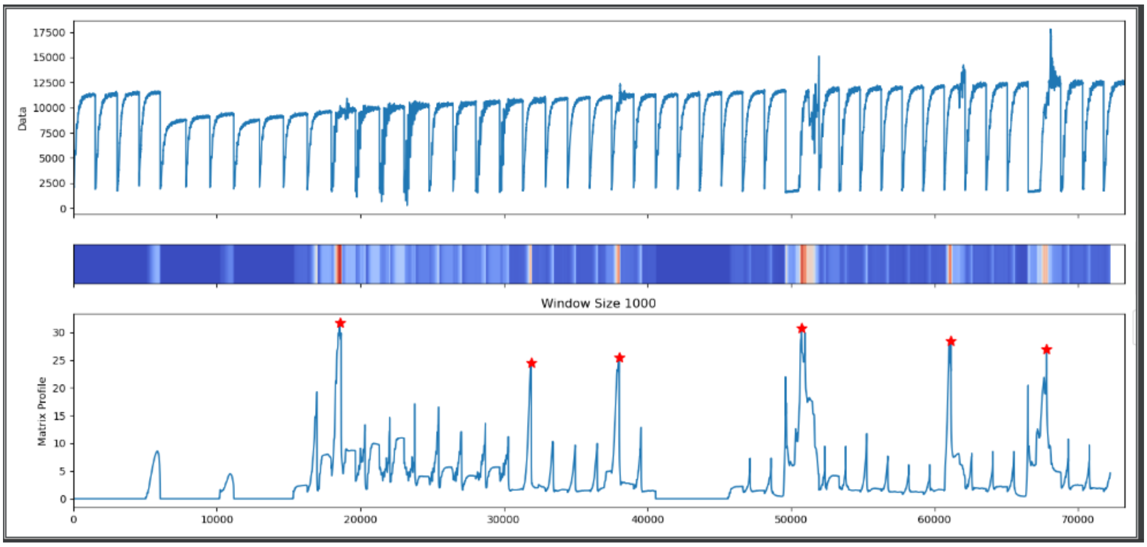

Figure 36: Depicts the Matrix Profile detected faults in the acoustic fault dataset.

Faiss-based Fault Detection

The Faiss-based method requires a weight feature tuning via the Relief algorithm to determine which statistical feature should be used and which should be omitted since they would only hinder the result. Once the most suitable statistical metrics are determined, the Faiss Detection method runs and utilizes the historical sample product cycles' statistical features to compute the base index.

The statistical features of the incoming data are then calculated and forwarded in the index and the furthest neighbor values are compared with a comparison threshold for discord detection. In this method the discord threshold, the statistical feature training and the base index creation can be computed only once and used for multiple detection processes.

The detection results for the acoustic signal dataset are depicted in Figure 37. The faults are depicted with yellow and the normal mode with green. The results can be compared to the ones depicted in Figure 37.

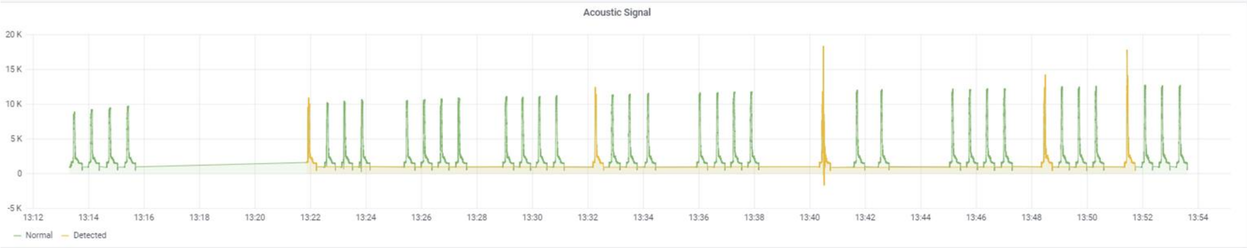

Figure 37: Depicts the Faiss detected faults in the acoustic fault dataset.

The evaluation metrics for the detection of acoustic signal type of faults are presented in Table 4.

Detection Method	Precision	Recall	F1
Matrix Profile	0.83	1	0.9
Faiss	1	1	1

Table 4: Fault Detection KPIs for the Matrix Profile and Faiss methods in a dataset that contains acoustic signal fault type

Fault Classification

In the Fault Classification section of the Fault Identification module, multiple faulty product cycles are provided as input and their most probable type is determined. All the motif discovery methods depend on sets of historical data of the various fault types. The motif discovery methods obtain the most appropriate match and calculate a distance measurement that ascertain the fault type of a product cycle via a comparison process.

All the methods successfully identified the errors based on their given labelling.
The pictures below depict the motif discovery process of the Matrix Profile method with a dynamically selected window size. There are three available fault types (i.e. early begin - Figure 40, acoustic signal - Figure 39, vibration - Figure 38) and one incoming product cycle that was detected as faulty.

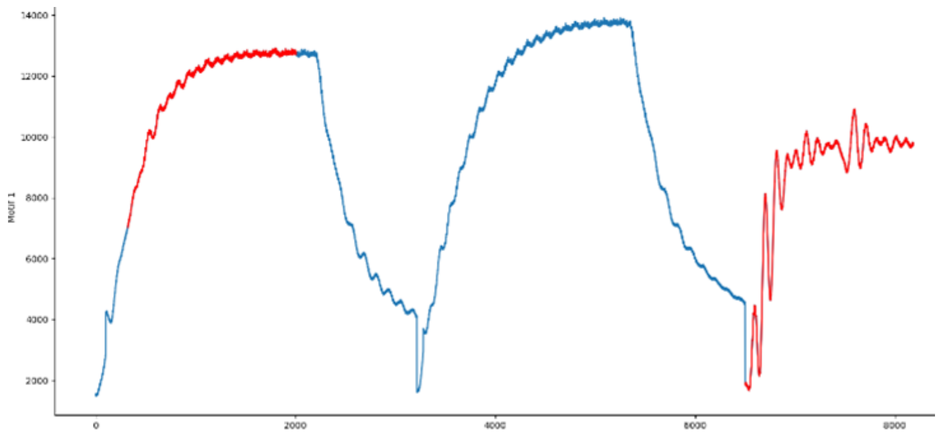

Figure 38: Vibration historical sample and an unidentified acoustic signal detected fault

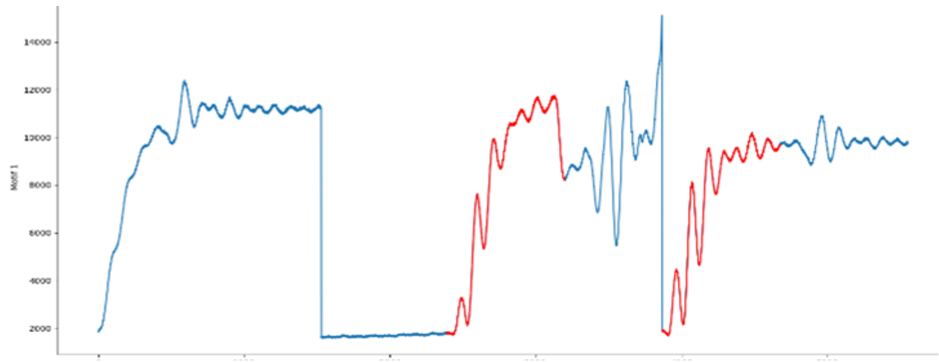

Figure 39: Acoustic Signal historical sample and an unidentified acoustic signal detected fault

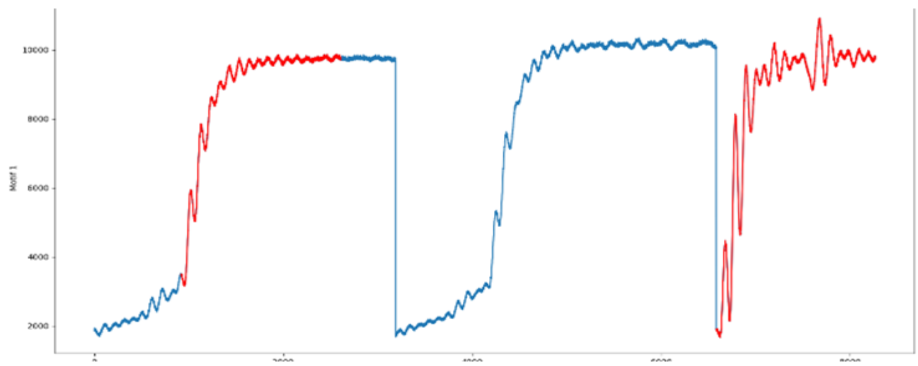

Figure 40: Early Begin historical sample and an unidentified acoustic signal detected fault

4.2.5 Use in Open Calls

The Fault Identification module has not been made available for experimentation in the first round of the Open Calls. However, the mentoring phase for the first Open Call winners could lead to a need for this digital enabler too. Providing the Fault

Identification as one of the Digital Enablers available for experimentation for the second Open Call is under consideration.

4.3 Quantitative Association Rules Mining

4.3.1 Digital Enabler Overview

4.3.1.1 Overview

QARMA is a system for extracting all valid, non-dominated quantitative association rules that can be derived/produced from a dataset that define multiple items and restrictions on zero or more of their attributes' values in the antecedents of the rule, and a single item and a restriction for a single attribute of that item. By a "restriction" of an item-attribute, we mean the requirement that the item-attribute's value is at least as large as the value specified in the restriction in the rule.

The system considers valid any rule that meets minimum required support and confidence levels. The system considers a rule r1 to be dominated by another rule r2 when:

- The consequent items of the rules are the same, as well as the consequent item's restricted attribute is the same,
- The antecedent items in r1 are a (non-strict) super-set of the antecedent items of r2,
- The restricted threshold value for r2's consequent item attribute specified is at least as big as the corresponding one specified for r1,
- All the restricted attributes of antecedent item attributes in r2 also appear restricted in r1, and the values in r1 are at least as big as those in r2,
- The support and confidence levels of r1 are at most as big as those of r2.

4.3.1.2 Possible Configurations

QARMA may be asked (i.e. configured) to:

- Produce only, those valid non-dominated rules that are "widest". A rule r2 is "wider" than r1 iff the pair of these rules satisfies all conditions 1-5 with the possible exception of the confidence level relation between r1 and r2.
- Produce all rules with up to a certain size of rule length.
- Consider the consequent's item attribute restriction to be specifying an equality ('=') rather than inequality ('>').
- Produce only rules with a given item-id in the consequent. This is particularly useful in a standard "supervised learning" context, where we wish to produce rules that classify instances, so we want only the "class attribute" to be the consequent item (in combination with the previous listed capability).

In the case when some attributes of some items take on too many distinct values in the dataset (e.g. the blood test values in the UCI ilpd dataset), it can be asked to consider only a "reasonable" subset of these values when creating rules containing restrictions on such item attributes.

There are more short-cuts and heuristics that QARMA can be asked to employ, but they are beyond the scope for this deliverable.

In the scope of QU4LITY QARMA has been customized and configured to support industrial use cases in Quality Management and Zero Defect Manufacturing. The industrial configuration and deployment of the framework is conveniently called QARMA4Industry.

4.3.2 The Business and ZDM Perspective

QARMA4Industry is applicable in three major types of Quality Management and ZDM problems:

- **Determination of associations between production variables**: Given production data, determine joint intervals of values for several (input) variables that imply the value of another (output) variable.
- **Remaining Useful Life (RUL) estimation**: Given sensor data readings from a machine/tool combination and related number of parts made before tool breaks, determine RUL for any new (unseen) sensor reading set.
- **Determination of process parameter settings to avoid**: Given production parameter runs leading to products that pass or not acoustic tests, determine a minimum set of production run variable settings combinations that lead to a minimum number of products that don't pass the (quality) acoustic tests.

4.3.3 The Technological Perspective

QARMA is a family of Quantitative Association Rule Mining algorithms. QARMA implements a Data Mining inspired approach:

- Sets of features that appear frequently in the dataset together are collected together, and then each feature is quantified (its value restricted in a numerical interval) with the goal to derive conditions that imply that a target variable among the features takes on a desired value
- Derived rules are of the form: $I\_1\in[l\_1,h\_1\ ]\ldots I\_n\ [l\_n,h\_n\ ]\rightarrow T\in[l,h]$

QARMA-based decision making is based on the following formula:

- Select all rules whose antecedent conditions are satisfied by this instance and add them to a set F.
- Sort the ruleset F in decreasing order of confidence and decreasing order of support on the training set.
- Remove all but the top-100 rules of the sorted set F.

- Each rule in F carries a weight equal to its confidence on the training set; decide as the instance's class the weighted majority vote of the rules in F.

4.3.4 Demonstrator and User Guide

4.3.4.1 Input Datasets of QARMA4Industry

QARMA works on datasets that can be viewed as sets of sensor readings ("transactions"), consisting of measurements of sensors ("items") that have various "attributes"; most often there is only a single attribute, namely the value of the measurement of that sensor, but it can also contain other information such as a timestamp etc. Each such measurement defines a transaction, for which the following data are available:

- A user-id of the sensor who makes the measurement.
- An item-id of the measured sensor.
- Attribute-value pairs of the form (attr-id, value) for some (at least one) of the measurement's attributes.

As an easy example, in a Predictive Maintenance (PdM) setting, the sensors attached to a machine/tool configuration would provide measurements of the property they are made to measure in periodic intervals; there could be measurements for the spindle torque, vertical displacement of the drill of a tool and so on. Each measurement from each sensor would constitute a transaction, and the set of measurements from all attached sensors at a particular moment in time would constitute a particular "user-history" (the terminology forms a legacy term from the original uses of the QARMA algorithm in the recommender system domain). In a supervised learning setting, at each moment for which measurements are recorded, one item would indicate the Remaining Useful Life (RUL) of the entire machine/tool configuration before maintenance is necessary. Such an item is usually called the "class attribute".

As a second example, in a medical domain example, the sensor readings may simply be instances of values for various blood tests, and a "class" value may indicate whether the given tests indicate the presence of a decease or not. In this case, the "items" are the various blood tests (what is commonly referred to as "attributes" in standard supervised machine learning literature), and each "item" has a single "attribute", that is its value. The class value is yet another item, with a single attribute, its value that can be zero (no decease) or one (decease present).

4.3.4.2 QARMA Input Formats

There are two major data formats in which data can be formatted for loading onto the QARMA database. For both formats, there is an implicit assumption that each “item” has exactly the same attributes as any other item in the dataset, and that there always exists one "major" common attribute, for which, in every user transaction, this attribute has a value in the transaction.

1st INPUT FORMAT: Extended MovieLens style format

The first input format is a text-based (ASCII) extension of the MovieLens CSV format [Harper16]. In this format, the first line of the file has the following format:

<attr1_name>::<attr2_name>::…::<attrn_name>

Where <attrk_name> is the name of the k-th attribute of each item in the dataset, besides the "main" common attribute of each item. If there are no "extra" attributes other than the "main" common attribute, this first "header" line can be omitted. Notice the use of the double column ("::") as separator.

The rest of the file contains lines of the form:

<user_id>::<item_id>::<main_attribute_value>::<attr1_val>::<attr2_val>…<attrn_val>

Where user_id is an positive long number uniquely identifying a user, item_id is a positive long number uniquely identifying the particular item the user has "purchased", main_attribute_value is a real number representing the value of the main attribute of the item, and attrk_val is a real number representing the value of the k-th attribute (whose name is <attrk_name> in the header), that can be the character '?' to indicate that the value is "N/A" or "unknown". Notice that double-column ("::") is used again as values separator.

An example of such an input file is the following (only the first 5 lines are shown for brevity):

amount_played::amount_won::ngpr

1000005332::1::0.5::1000000::0::1

1000005664::1::1::200000::0::1

1000005664::1::0.15::200000::0::1

1000005664::1::2::25000::0::1

1000008703::1::0.04::750000::400000::2

Notice that in the above dataset, the user with id "1000005664" has purchased item with id "1" 3 times, with different values for the standard "main_attribute" attribute, as well as for the "amount_played", "amount_won" and "ngpr" attributes. This particular feature is not commonly found in other algorithms in the ML/DM literature.

2nd INPUT FORMAT: CSV Format

The 2nd format is a text-based input format in which every line contains the entire history of a user. Rows therefore represent users histories, and columns represent the items in the dataset. Therefore, the j-th column of the i-th row is a value that represents the major value that the i-th user "paid" for the j-th item.

The 1st row of the file is a header with the names of each item in the dataset:

<item1_name>,<item2_name>,…<itemn_name>

Where <itemk_name> is the name of the k-th item in the dataset. Notice the use of the comma (‘,’) as separator.

The rows afterwards have the following format:

<item1_value>,<item2_value>,…,<itemn_value>

Where <itemk_value> is the value that the i-th user "paid" for the k-th item, and is ‘?’ if the user did NOT "purchase" the k-th item. Again, comma (‘,’) is used as values separator.

An example file containing data from the UCI ilpd (Indian Liver Patient Database) dataset is the following:

age,gender,TB,DB,AAP,Sgpt,Sgot,TP,ALB,AGRatio,sick

65,0,0.7,0.1,187,16,18,6.8,3.3,0.9,1

62,1,10.9,5.5,699,64,100,7.5,3.2,0.74,1

62,1,7.3,4.1,490,60,68,7,3.3,0.89,1

58,1,1,0.4,182,14,20,6.8,3.4,1,1

In the above file excerpt, the top row is the header, the 2nd row represents the data for the 1st user, the 3rd row represents data of the 2nd user and so on.

4.3.4.3 QARMA Outputs

QARMA writes ALL the rules it produces in its database (a MySQL RDBMS database).

The schema where these rules are stored is shown in Figure 41. The tables that are populated with the rules found are the tables:

- associationrule
- associationruleitem
- associationruleitemattrs

The other tables shown in the figure must have been populated (as well as some other tables not shown in the figure) before each run.

QARMA also writes its rules in a plain text file (whose name is provided as a parameter when invoked from the command line), with 1 line for each rule found, printing the rule first, then the support, confidence, lift, and conviction metrics of the rule on the dataset, as in the following example (from a fault-diagnosis in power grids domain):

```
…
[local_current_A.p >= 1804.2904489438188]&[local_current_A.price_negated >= -1981.6745015818599] ^ [local_current_B.p >= 2107.54438476501]&[local_current_B.price_negated >= -4078.560855142548] --> [fault_type.p = 1.0] (Supp=3.6333333333333333%/Conf=90.38142620232172%)(Conv=859.5175862068961%/Lift=5.216319326798098)
[local_current_A.p >= 1856.5393537627472]&[local_current_A.price_negated >= -2012.4584283319073] ^ [local_current_B.p >= 2107.54438476501]&[local_current_B.price_negated >= -4078.560855142548] --> [fault_type.p = 1.0] (Supp=3.58%/Conf=90.4040404040404%)(Conv=861.5431578947366%/Lift=5.217624494269357)
[local_current_A.p >= 2113.2047423272825]&[local_current_A.price_negated >= -4066.764823747749] ^ [local_current_C.p >= 1807.6831125547562]&[local_current_C.price_negated >= -1984.9702763519963] --> [fault_type.p = 0.0] (Supp=3.6066666666666665%/Conf=90.16666666666667%)(Conv=840.0000000000005%/Lift=5.181992337164751)
…
```

QARMA also outputs in the console in which it is invoked the total dataset coverage (in terms of the percentage of users covered by the rules found).

In the above example, the dataset contains two attributes, the standard "price" attribute (which is the actual measurement of a sensor in a user-history), plus the negation of the "price" attribute (i.e. the negative value of that measurement). This means that the 1st rule shown above reads as follows:

local_current_A ∊ [1804.2904489438188, 1981.6745015818599] ^
local_current_B ∊ [2107.54438476501, 4078.560855142548] → fault_type.p = 1.0.

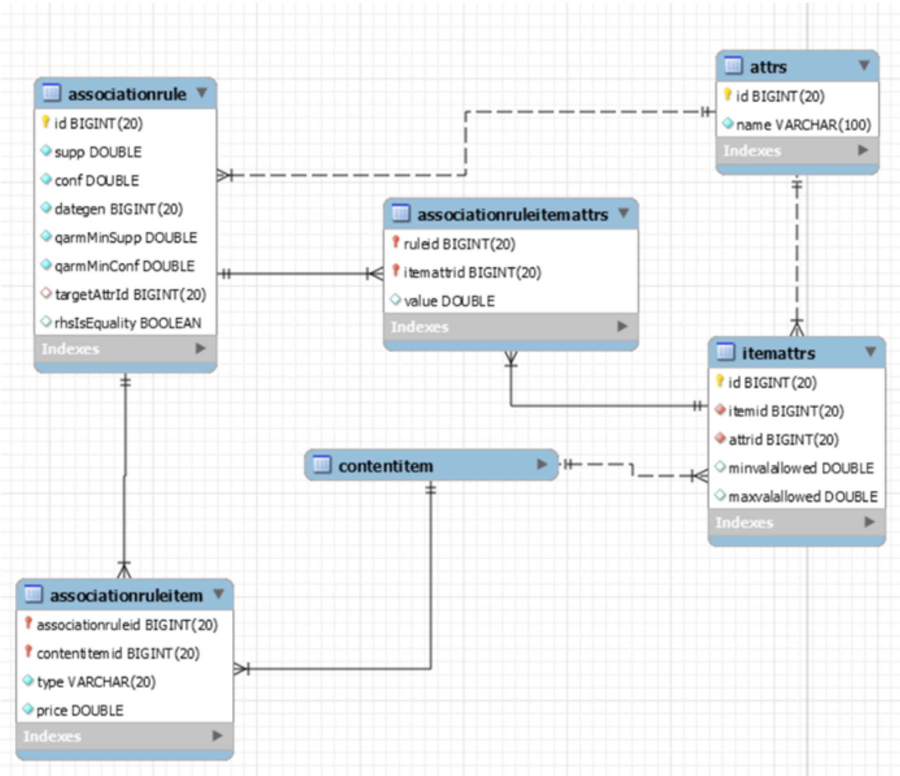

Figure 41: QARMA RDBMS tables for storing produced rules

4.3.4.4 QARMA Run-time API

Once QARMA has ran and populated the RDBMS tables with the rules it has found, a number of Java classes are available to the developer to take advantage of the produced rules. The following figures provide a short description of some of these utility classes and programs.

DBQuantRuleBetterFasterTester:

gr.ait.mytv.qarma

Class DBQuantRuleBetterFasterTester

java.lang.Object
gr.ait.mytv.qarma.DBQuantRuleBetterFasterTester

```
public class DBQuantRuleBetterFasterTester
extends java.lang.Object
```

test the performance of a set of QuantRule objects saved in the DB on a test set specified in a CSV file. Only works with QuantRule's that specify an exact value in their consequent (rhsIsExact must be true). The objective is to measure classification performance, by selecting from the top firing rules the majority vote as the class label.

main

```
public static void main(java.lang.String[] args)
```

invoke as `java -cp <classpath> gr.ait.mytv.qarma.DBQuantRuleBetterFasterTester <filename> <cons_item_id> [target_attr_id(-1)] [user_rows_range_to_include] [add_negative_price(false)] [exists_header(false)] [min_num_sat_rules_reqd(10)] [max_num_rules_2_vote(100)] [min_conf_level_reqd(0.0)] // value must be in [0,1] [do_same_consumeditem_all_attr_values_test(false)] [item_col_to_exclude(1...num_cols)[,]]* [debug_level_$classname$,$level$(0)]*` . where the filename specifies a CSV file containing the test-set, and the cons_item_id specifies the target item id that is the "target variable" in the classification problem that we're testing against (all involved rules must have this item as consequent.) The optional range (specified as <start_row>-<end_row>) where both start_row and end_row are in the range [1...num_user_rows_in_test_file] specifies the range of rows in the test set to consider. Obviously, the add_negative_price, exists_header, and item_col_to_exclude* parameters must be the same ones used to upload the original training set to the DB, if the file formats are the same. The parameter min_num_sat_rules_reqd specifies the minimum number of rules that must fire on an instance so that there will be a decision from the rules on the class of that instance. The program will print the overall rule accuracy on the test set.

20180618: Notice that the extra argument do_same_consumeditem_all_attr_values_test was added in case the dataset has multiple consumed items with same item-id for the same user, and multiple attributes (currently only needed for ILOT RG and HOL datasets).

20200815: If a file called "allowed_item_ids.txt" exists in the same directory from where the command-line is executed, then only rules whose antecedents and consequent are in the list of items described in the file (line separated) will be considered.

20200827: If the string "null" is passed for the first argument, then the program simply prints out the rules in the DB that have the correct target_item_id, target_attr_id and rhsIsExact is gotten from the 6th argument (normally standing for exists_header in the csv testset.)

Parameters:

`args` -

DBQuantRuleValidator:

gr.ait.mytv.qarma

Class DBQuantRuleValidator

java.lang.Object
    gr.ait.mytv.qarma.DBQuantRuleValidator

```
public class DBQuantRuleValidator
extends java.lang.Object
```

test the performance of a set of QuantRule objects saved in the DB on a test set specified in a CSV file and delete from DB rules not meeting the minimum requested support and/or confidence unless they do not fire at all in the test-set.

Notes:

- 20190724: new required value added to remove any rules that do not reach 100% confidence for consequent attribute level specified in the 2nd argument, unless target_attr_id is set to some value other than -1L (the default) in which case the value specified in the 2nd argument has no meaning (no effect) By setting a very low value (less than or equal to the minimum value the consequent item attribute may take on the training set), the feature is made ineffective (has no effect).
- 20180618: the do_same_consumeditem_all_attr_values_test test flag was also added to test with HOL and ILOT RG datasets correctly as well.

main

```
public static void main(java.lang.String[] args)
```

invoke as `java -cp <classpath> gr.ait.mytv.qarma.DBQuantRuleValidator <filename> <level_with_guaranteed_100_confidence> [min_req_support(original_supp)] [min_req_confidence(original_conf)] [min_req_lift(Double.NEGATIVE_INFINITY)] [min_req_conviction(Double.NEGATIVE_INFINITY)] [target_attr_id(-1)] [rhs_is_exact(false)] [user_rows_range_to_include] [add_negative_price(false)] [exists_header(false)] [required_cons_item_level(Double.NEGATIVE_INFINITY)] [do_same_consumeditem_all_attr_values_test(false)] [item_col_to_exclude(1...num_cols)]*` . where the filename specifies a CSV file containing the validation-set, the optional range (specified as <start_row>-<end_row>) where both start_row and end_row are in the range [1...num_user_rows_in_test_file] specifies the range of rows in the test set to consider; obviously, the add_negative_price, exists_header, and item_col_to_exclude* parameters must be the same ones used to upload the original training set to the DB, if the file formats are the same. If the parameter required_cons_item_level is specified (to a real number), all the rules will assume that this is the level of their consequent item, and compute support/confidence etc. according to this value. If the parameter do_same_consumeditem_all_attr_values_test is specified and true, then for a rule to be satisfied, all restrictions of an antecedent item must occur within the same consumed item (needed currently only for the HOL and ILOT RG datasets).

The program will print the support/confidence/lift/conviction for each of the rules in the DB as measured on the validation-set, and will delete from the DB any rule that does not meet the min_req_support or min_req_confidence or min_req_lift or min_req_conviction specified.

Parameters:

`args - String[]`

QuantRule:

gr.ait.mytv.qarma

Class QuantRule

java.lang.Object
    gr.ait.tlop.PoolableObject
        gr.ait.mytv.qarma.RuleBase
            gr.ait.mytv.qarma.QuantRule

All Implemented Interfaces:

`java.io.Serializable, java.lang.Comparable`

```
public class QuantRule
extends RuleBase
implements java.lang.Comparable
```

encapsulates multi-attribute quantitative association rules, with the restriction that there can be only a single quantitative attribute specified in the rule's consequent item.

Notes:

- 20190529: modified `toString(),toStringGeneralCase()` methods to include data in a more meaningful manner.
- 20190919: corrected a bug in the method `getConfidencePercLvlFastBV(BooleanVector)` which would only show up if the method was called without the method `getSupportPercLvlXXX()` having been called before to set the cache value of `_support` field; in such a case, a value of 100.0 would always be returned. Now the method is fixed.
- 20190919: introduced a new method `getExistConfidencePercLvlFastBV(BooleanVector)` that computes the "existential confidence" of the rule by requiring that in the denominator we count only those user-histories that satisfy the antecedents PLUS they include a purchase of the consequent (at any value for the target-attr). Similarly, introduced a method `getExistConfidencePercLvlFast()` with the same functionality as above -but doesn't require the BV impl functionality.
- 20200826: added the method `toStringEasy()` that assumes every item has up to 2 attributes: the normal "price" attribute plus optionally the "price_negated" attribute. This method allows rules to be printed without references to attributes at all, which helps in databases coming from common CSV data files.
- 20201105: modified the product and division that takes place in the computation of the confidence of a rule, as numerical round-off errors may make a rule seem non-dominated while it is (it was previously possible to get a confidence of 100.000000001% for a rule, which would make it appear non-dominated to another rule that gives 100% confidence)
- 20201120: made package method `getDBId()` public.
- 20201120: bug fix in `dominatesGeneralCase()` method: there was a problem when this rule specifies a value that is above the lowest but less than or equal to the 2nd lowest in the dataset and the other rule specifies the lowest value in the dataset. In this case, the two variables consAttrAntItemLevelsm and r_consAttrAntItemLevelsrpos will be equal to each other, but this rule clearly does NOT dominate and false should be returned immediately.

See Also:

Serialized Form

Constructor Summary

Constructors

Constructor and Description
`QuantRule(long consid, long consattrid, long[] antids)` public constructor.
`QuantRule(long consid, long consattrid, long[] antids, boolean isConsAttrLevelExact)` public constructor is like the 3-arg constructor but also specifies whether the quantitative part of the RHS of the rule is to interpreted as an equality(=) rather than an inequality(≥).
`QuantRule(RuleBase rb)` public copy constructor.
`QuantRule(gr.ait.tlop.ThreadLocalObjectPool<RuleBase> pool)` public constructor needed when using object pooling via the tlop library.

Method Summary

All Methods | Instance Methods | Concrete Methods | Deprecated Methods

Modifier and Type	Method and Description
int	compareTo(java.lang.Object other) QuantRule a is smaller than QuantRule b if: a-confidence < b-confidence else if a-confidence==b-confidence then if a-support < b-support else if a-support==b-support then if a-cons-attr-id < b-cons-attr-id else if a-cons-attr-id==b-cons-attr-id then if a-cons-attr-value < b-cons-attr-value else if a-cons-attr-value==b-cons-attr-value then if a-cons-item-id < b-cons-item-id else if a-cons-item-id==b-cons-item-id then if a-num-ant-items < b-num-ant-items else if a-num-ant-items==b-num-ant-items then compare _antItemIds, then _consAttrAntItemLevels, then _itemAttrLevels of a and b to decide.
boolean	dominates(QuantRule r, boolean ignore_sup_conf) check if this rule dominates the argument rule passed in.
boolean	dominatesGeneralCase(QuantRule r, boolean ignore_sup_conf) check in the general case, if this rule dominates the argument rule passed in.
boolean	dominatesGeneralCaseOld(QuantRule r, boolean ignore_sup_conf) **Deprecated.** 20180606
boolean	dominatesOld(QuantRule r, boolean ignore_sup_conf) **Deprecated.** 20180606
boolean	equals(java.lang.Object o) performs field-by-field check for equality.
double	getConfidencePercLvl() **Deprecated.**
double	getConfidencePercLvlFast() faster version of the deprecated getConfidencePercLvl().
double	getConfidencePercLvlFastBV() faster version of the deprecated getConfidencePercLvl().
double	getConsAttrLevel() get the target-attr level of the consequent item.
double	getConvictionPercLvlFastBV(BooleanVector bv) return the conviction of this rule as a percentage.
long	getDBId() get the id by which this rule is indexed in the db.
double	getExistConfidencePercLvlFast() same as getExistConfidencePercLvlFastBV(BooleanVector) but doesn't require BV implementation functionality.
double	getExistConfidencePercLvlFastBV(BooleanVector sup_users) same as getConfidencePercLvlFastBV(bv) but the denominator is required to have only those user-histories that actually contain a "purchase" for the consequent item at any value.
double	getItemAttrLevel(long itemid, long attrid) return the level of the specified attribute for the specified item.
double	getIthAntItemConsAttrLevel(int i) return the target-attr level of the i-th antecedent item.
long	getLeastPreferredAntItemIdAttrQuantified(long attrid, java.util.Map<java.lang.Long,java.lang.Integer> posOrderMap) get the least-preferred item-id among the antecedent items of this rule that have the particular attribute-id quantified, where preferences are determined according to the second argument.
long	getLeastPreferredAntItemIdQuantified(java.util.Map<java.lang.Long,java.lang.Integer> posOrderMap) get the item-id of the least preferred item among the antecedent items of this rule that have any of their attributes quantified.
double	getLiftLvlFastBV(BooleanVector bv) return the lift of this rule (NOT as percentage, as the lift is a simple ratio of two percentages).
long	getMaxAntItemIdAttrQuantified(long attrid) get the maximum item-id among the antecedent items of this rule that have the particular attribute-id quantified.
long	getMaxAntItemIdQuantified() get the maximum item-id among the antecedent items of this rule that have any of their attributes quantified.
double	getSupportPercLvlFast() faster version of the deprecated getSupportPercLvl().
double	getSupportPercLvlFastBV() faster version of the deprecated getSupportPercLvl().
double	getSupportPercLvlFastBV(BooleanVector sup_users) faster version of the deprecated getSupportPercLvl().
int	hashCode() computes the (integer-truncated) sum of all item-ids in the rule.
void	invalidateSuppConfCaches() used to force clean computation of support and confidence levels of this rule.

Methods inherited from class gr.ait.mytv.qarma.RuleBase

antItemsAreSubsetOf, existsItemInAntecedents, getAntecedentItemIds, getAntItemIndexByItemId, getConsItemAttrId, getConsItemId, getIthAntItemId, getNumAntecedentItems, setConsequentItemAttrId

Methods inherited from class gr.ait.tlop.PoolableObject

isManaged, isUsed, newInstance, release, safeRelease, setIsUsed

Methods inherited from class java.lang.Object

clone, finalize, getClass, notify, notifyAll, wait, wait, wait

DataMgr:

gr.ait.mytv.qarma

Class DataMgr

java.lang.Object
    gr.ait.mytv.qarma.DataMgr

```
public class DataMgr
extends java.lang.Object
```

contains static methods for managing I/O in the QARMA method, plus it holds the global rule-set to be created as QARMA output.

Notes:

- 20190408: Notice that the method resetGlobalQuantRulesInDb_Unsynch() now accepts the target_attr_id param that makes sure that only those rules for the particular target_attr_id are deleted.
- 20190602: added new method, uploadMoreCSVFormattedRowsIntoDB() to allow for "augmenting" dataset with more "test" rows, after a QARMA run (to show up in the GUI classes).
- 20190920: introduced method fillItemConsAttrValuesCacheFastFromDB(long taid) that works much faster than the non-fast method by executing a single SQL query in the RDBMS. Also added method fillItemAttrValuesCacheFastFromDB(long taid) that works as the above mentioned method (ie goes to the DB and performs one or two queries the most).
- 20190921: introduced method fillItemAttrLevelsSupportCacheBVFaster(long taid) that has the same functionality as fillItemAttrLevelsSupportCacheBV(long) but is hopefully faster by not creating unneeded ItemAttr and ItemAttrLevel objs.
- 20190923: introduced methods fillAttrItemLevelsSupportCacheBVFaster(long taid), getAllUserIdsWithItemAttrAboveLevelBV2(long iid,long aid,double lvl,long tid) , getAllUserIdsWithItemAttrAboveLevelBV2_Unsynch(...) methods for speeding up the item-attr-level support caches creation when BV implementation is enabled.
- 20200823: added debug info to print rule loading information when asked.

static boolean	**deleteQuantRuleFromDB_Unsynch**(long ruleid) delete the rule with db-id equal to the given argument from the DB.
static double[]	**getAllAttrValues4ItemSortedAsc_Unsynch**(long itemid, long attrid, long target_attr_id) this method will only work reliably if in the main program thread, the _itemAttrValues lookup table is populated (via the fillItemAttrValuesCache() method), before any other threads ever invoke this method.
static double[]	**getAllAttrValues4ItemSortedAsc**(long itemid, long attrid, long target_attr_id) would be better implemented as an appropriate SQL query.
static double[]	**getAllConsequentAttrValues4ItemSortedAsc_Unsynch**(long itemid, long target_attr_id) this method will only work reliably if in the main program thread, the _itemConsAttrValues lookup table is populated (via the fillItemConsAttrValuesCache() method), before any other threads ever invoke this method.
static double[]	**getAllConsequentAttrValues4ItemSortedAsc**(long itemid, long target_attr_id) would be better implemented as an appropriate SQL query.
static long[]	**getAllItemIds**(long target_attr_id) would be better implemented as an appropriate SQL query.
static java.util.List<java.lang.Long>	**getAllItemIdsFromDB_Unsynch**() method used by GUI application, is not synchronized.
static double[]	**getAllMultipleAttrValues4Item**(long itemid, long attrid, long target_attr_id) would be better implemented as an appropriate SQL query.
static double[]	**getAllMultipleConsequentAttrValues4Item**(long itemid, long target_attr_id) would be better implemented as an appropriate SQL query.
static java.util.Set<java.lang.Long>	**getAllUserIdsWithItemAttrAboveLevel_Unsynch**(long itemid, long attrid, double level, long target_attr_id) this method will only work reliably only if the fillItemAttrLevelsSupportCache() method has been called from the main thread, before any other spawned threads call this method.
static java.util.Set<java.lang.Long>	**getAllUserIdsWithItemAttrAboveLevel**(long itemid, long attrid, double level, long target_attr_id) get the users with at least one consumed item with attr value above specified threshold.

static void	**uploadCSVFormattedDataIntoDB**(java.lang.String filename, boolean add_negative_price_attr, boolean exists_header, int[] items_to_exclude, int start_row, int end_row) populates the cpace2 schema on a MySQL RDBMS with data contained in a CSV file.
static void	**uploadExtraCSVFormattedDataIntoDB**(java.lang.String filename, boolean add_negative_price_attr, boolean exists_header, int[] items_to_exclude, int start_row, int end_row) adds more data (columns only) to the cpace2 schema on a MySQL RDBMS contained in a CSV file.
static void	**uploadMoreCSVFormattedRowsIntoDB**(java.lang.String filename, boolean add_negative_price_attr, boolean exists_header, int[] items_to_exclude, int start_row, int end_row) uploads the rows described in the specified file, without deleting any prior data in the DB.

static void	**uploadMovieLensExtFormattedDataIntoDB**(java.lang.String filename, boolean add_negative_price_attr) populates the cpace2 schema on a MySQL RDBMS with data contained in a file containing item ratings in an extension of the Movie-Lens format to accomodate multiple attributes besides the ratings attribute.
static void	**uploadMovieLensFormattedDataIntoDB**(java.lang.String filename, boolean add_negative_price_attr) populates the cpace2 schema on a MySQL RDBMS with data contained in a file containing item ratings in Movie-Lens format.

ConsumedItem:

gr.ait.mytv.qarma

Class ConsumedItem

java.lang.Object
  gr.ait.mytv.qarma.ConsumedItem

All Implemented Interfaces:

java.io.Serializable

```
public final class ConsumedItem
extends java.lang.Object
implements java.io.Serializable
```

encapsulates the notion of an Item consumed by a user in the QARMA method. Revision Notes:

2020-11-19: made package method getTotalNumUsers() public as many applications in other packages can benefit.

2020-01-31: added method reloadUserHistoriesFromCSVFile() that reads chunks of user-histories from a CSV file that may have been garbage-collected -them being soft refs in the 1st place.

2019-05-02: added support for comment lines (starting with the "hash" character '#') when loading user histories from CSV files only.

2018-07-20: added loadAllUserHistoriesFromMovieLensExtFile(String) and loadUserHistoryFromMovieLensExtFile(String) methods to allow loading test data from MovieLens-Extended format, primarily to allow testing and applications such as PdMSeriesTester in the apps package to work. This made the existence of the new field _mlextFile necessary.

2018-06-16: added loadAllUserHistoriesFromDB(long) method to support producing rules whose consequent item attribute is not the standard price attribute. This necessitated the need to modify the getUserConsumedItemsByUserId(long uid) method to accept a 2nd parameter, long target_attr_id, and also, to add a getAttrPosByAttrId(long) method.

2016-02-08: changed the _userCIs data structure to hold as values, instead of a SoftReference<ConsumedItem[]>, an SRHolder object, which is exactly a value-holder for a SoftReference<ConsumedItem[]>, thus allowing the re-loading of soft references into the _userCIs hash-map without running any risks of modifying the hash-map while it is being iterated via a ReadOnlyIterator.

2016-02-10: modified loadAllUserHistoriesFromCSVFile() and loadUserHistoryFromCSVFile() so that the BufferedReader that is opened to read the CSV file, is properly closed before the method exits.

See Also:

Serialized Form

Constructor Summary

Constructors

Constructor and Description
ConsumedItem(long userId, long itemId, long whenConsumed, long[] attrids, double[] attrvals, double price) sole public constructor.

Method Summary

All Methods | **Static Methods** | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method and Description
int	getAttrPosByAttrId(long attrid) return the index in the _attrIds array of the attribute whose id is specified in the argument.
double	getAttrValByAttrId(long attrid) return the value of the attribute whose id is specified in the argument.
double	getConsequentAttrValue() return the consequent (target) attribute value for this ConsumedItem.
long	getItemId() return the item-id for this ConsumedItem.
long	getIthAttrId(int i) return the attr-id of the i-th attr specified in the constructor.
double	getIthAttrVal(int i) return the attr-value of the i-th attr specified in the constructor.
int	getNumAttrs() return the number of attributes specified for this ConsumedItem in its constructor.
static int	getTotalNumUsers() return the total number of user-histories in the DB.

static ConsumedItem[]	getUserConsumedItemsByUserId(long userid, long target_attr_id) get the consumed items of the user with the id given in the argument.
long	getUserId() return this ConsumedItem's user-id.
long	getWhenConsumed() return the time (unix-time milliseconds) when this ConsumedItem was consumed.
static int	loadAllUserHistoriesFromCSVFile(java.lang.String filename, boolean add_negative_price_attr, boolean exists_header, int[] items_to_exclude, int start_row, int end_row, long target_attr_id) loads all user-histories found in a CSV file, reseting first the _userCIs, _seqUIDs hash-table and _userids array.
static void	loadAllUserHistoriesFromDB() loads all user-histories in the ConsumedItem class cache.
static void	loadAllUserHistoriesFromDB(long target_attr_id) loads all user-histories in the ConsumedItem class cache, and swaps each ConsumedItem's price with the appropriate value for the attribute specified in the parameter, if the parameter is other than -1, so that QARMA can be used to infer rules whose consequent items specify attribute values for attributes other than the standard price attribute.
static void	loadAllUserHistoriesFromMovieLensExtFile(java.lang.String filename) loads all user-histories found in a MovieLensExt-formatted file, reseting first the _userCIs, _seqUIDs hash-table and _userids array.
java.lang.String	toString() returns a String representation of this ConsumedItem containing only the _itemId, _consValue, _attrIds, and _attrValues information.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

4.3.4.5 QARMA UIs

QARMA comes with a set of 2 GUIs that allow to visualize various aspects of the produced rules. The first GUI allows the user to see all the rules in the DB and upon selection of a rule, to see the "user histories" (data instances) that satisfy the antecedents of the selected rule, along with the value of the target item/attribute. The GUI is shown in Figure 42.

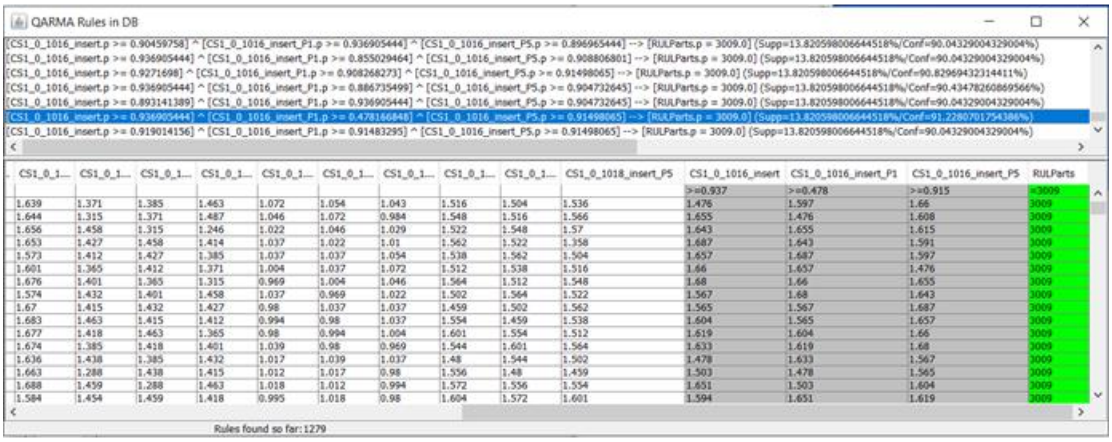

Figure 42: QARMA Rule Visualization Showing All Instances in DB Matching Rule Antecedents. The column with the green background is the target variable.

The second GUI allows not only to inspect and rule, but to modify the rule on-the-fly so as to see how the rule's support and confidence metrics change as the intervals in which the antecedents are required to be in change. Figure 43 is a screenshot of this tool. The pie-chart in the middle of the bottom panel of In Figure 43 shows the confidence of the rule, the straight blue line immediately below shows the selected item's position with respect to the extreme values the item/attribute takes on the training dataset, and the numbers in green below show respectively how many data instances in the dataset satisfy both the antecedents and the consequent of the rule, how many satisfy only the antecedents, and how many instances are the database in total.

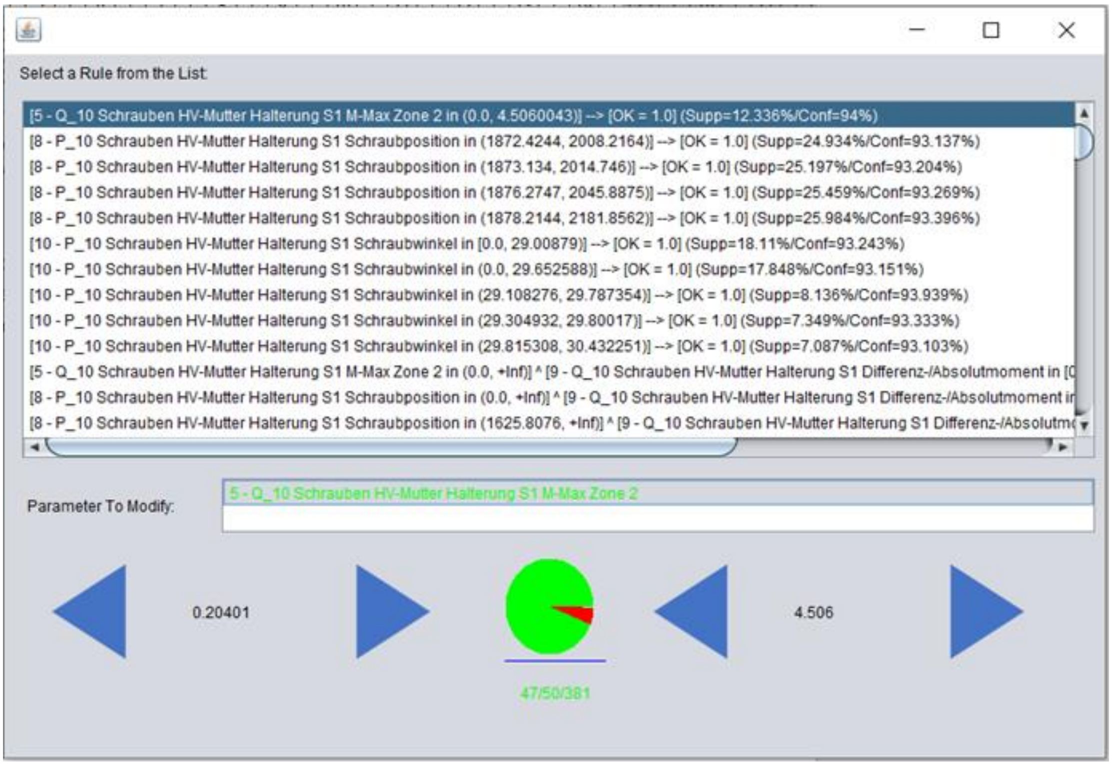

Figure 43: QARMA QuantRule Inspection and Modification Tool

4.3.5 Use in QU4LITY Pilots

4.3.5.1 Parameter settings combinations to avoid at THYSSENKRUPP

QARMA was able to determine that setting only 4 variables (out of about 90) to certain values in the intervals specified in any one of 67 combinations, leads to at least 3 products in the run that do not pass the acoustic tests. These are parameter combinations to avoid. More information is provided in WP7 deliverables (D7.5/D7.6).

4.3.5.2 Associations between input and output variables at RIASTONE

Even with large datasets (comprising about half a million rows of data), QARMA is able to extract statistically significant rules connecting input variable values to output variable values.

With sensors attached to the machines participating in the production process and with computer vision-based quality assurance of the manufactured products, estimation of RUL is expected to improve in performance. More information is provided in WP7 deliverables (D7.5/D7.6).

4.3.6 Use in QU4LITY Open Calls

QARMA4Industry has been deployed and used by QU4LITY Open Call Winners (e.g., IDEAL) as part of their experiments. For example, QARMA4Industry has already been deployed and used in the fashion industry, in the development of a system that could predict the degree of fit of a particular morphotype to a set of body measurements. This additional use case has proven the generality of the digital enabler, while confirming its performance advantages over conventional machine learning techniques like Support Vector Machines (SVN), Linear Regression, and Artificial Neural Networks (ANN) (with a single hidden layer). In the scope of the Open Call experiments, QARMA4Industry has been enhanced with support for categorical attributes, as well as with a special GUI (Graphical User Interface) for experimenters. These additional developments are detailed in the experimenters' support tasks/deliverables of WP6 of the project.

QARMA4Industry, including its enhancements realized in the 1st round of Open Call experiments will be also made available to the proposers and winners of the 2nd QU4LITY Open Call.

4.4 Anomaly Detection for Quality Control

4.4.1 Digital Enabler Overview

Reliably detecting anomalies in a given set of images is a task of high practical relevance for visual quality control in industrial manufacturing processes. For this purpose, Neural Networks, such as for example Auto Encoder Networks, that consist of an encoder and a decoder network can learn to reconstruct images of normal products, while they will learn to reconstruct infrequent defects only very poorly. Hence such a trained neural network can classify images as anomalies, when the reconstruction error exceeds some threshold. At the same time the defect can be localized from the distribution of the reconstruction error over the product image. A striking advantage of such an unsupervised method is that it does not require time-costly selection and labeling of a training set as is the case for supervised deep learning method and more traditional computer vision techniques. Furthermore, encoder network maps the product images is to a low dimensional space. This mapping can reveal information about product properties and corresponding variation in the production process which might not be easily visible in the original product images.

4.4.2 The Business and ZDM Perspective

Industrial production processes are nowadays extremely efficient in the sense that generally production speeds are very high and defects rates are very low. Hence images acquired with inline camera systems for defect detection automatically provide an enormous dataset that is representative for normal process capabilities. Thus, without any prior knowledge on product specifications and possible defects an Auto Encoder trained on such a dataset may provide a classification into normal products and deviating products. This even holds when the product portfolio is not completely uniform. On the other hand, when the product portfolio becomes too differentiated or defects rates become too high the anomaly classification will deteriorate unless additional labelling is provided for the training set.

As long as defects' rates do not increase significantly, time ordered datasets may reflect changes in the product portfolio, potential drifts in process capabilities and thereby drifts in process stability and installation status. Application of a repetitively trained auto encoder will keep distinguishing the most deviating products from normal products even when the target products or the production process changes. As supervised retraining of defect classification algorithms with labeled data and following implementation in production systems can be very costly the use of an intrinsically unsupervised method based on reconstruction errors in raw production data can be very advantageous to catch any previously unseen defects.

In addition to the reconstruction errors, analysis of the encoder mapping to latent space may reveal structures that are not easily visible in the variation of the product images, and thereby assist in tracing the source of processes distortion before these lead to actual defects.

4.4.3 The Technological Perspective

The enabler consists of Deep Learning network that can be applied on images of a fixed format. The Deep Learning network maps these images via a number of (convolutional) layers to a latent space of a fixed dimension. Next the decoder in the network reconstructs an image in a reverse version. By training the network on a set of images the network learns to reconstruct images constituting the bulk of the dataset. The reconstruction error can be used as a measure of the deviation between the actual and the desired product. When the reconstruction error exceeds a certain threshold value the product is considered as anomalous.

4.4.4 Use in QU4LITY Pilots

The anomaly detection system designed by TNO is being evaluated on image data originating from the Philips Pilot line. As soon as it becomes clear that the applied autoencoder can detect product defects that are relevant to the control of the pad printing process the algorithm can be deployed in the pilot line for online evaluation.

4.5 Image Analyzer for Surface Inspection

4.5.1 Digital Enabler Overview

In laser-based additive manufacturing (AM) of metal parts from powder bed, information about actual part quality obtained during build is essential for cost-efficient production and high product quality. In addition to that, the overall system is nowadays required to have a standardized API and communication interface that allows easy and sufficient integration into existing quality management infrastructure. To address this demand, a novel analysis approach using high dynamic range (HDR) optical imaging in combination with convolutional neural networks (CNN) is proposed for spatially resolved and layer-wise prediction of the surface roughness and defective surface areas of LPBF (Laser Powder Bed Fusion) parts. In a further step, the predicted surface roughness maps are used as a feedback signal for a reinforcement learning technique that employs a dynamics model to subsequently identify optimal process parameters under varying and uncertain conditions. The proposed approach ultimately combines the estimation of the local surface roughness based on image texture and model-based reinforcement learning (MBRL) to an in-situ optimization framework for LPBF processes and provides standardized communication interfaces for a fast integration into existing manufacturing data flows. The preliminary results underline that the system provides first step towards highly adaptive and self-optimizing machines in the field of automated laser powder bed fusion with the primary goals of reducing production costs and improving the environmental fingerprint as well as print quality.

4.5.2 The Business and ZDM Perspective

As a metal-based 3D printing technology, the laser powder bed fusion process faces several challenges when it comes to quality assurance and process optimization.

For example, the lack of process reproducibility and the resulting quality differences between work pieces hinder the transition of the technology to mass production. Hence, a reliable and cost-effective approach for in-situ quality monitoring and process optimization is highly demanded [Tapia2014].

A significant quality parameter in LPBF is the **increased roughness** of the as-built surfaces, which potentially leads to **reduced fatigue life** of the final part due to the concentration of residual stresses on the surfaces [Whip2019]. Additionally, high surface roughness generally leads to **poor surface quality** and therefore requires long and **expensive post-finishing operations**. The final part surface is often specified to be in range of the roughness defined by the current application which can require a surface roughness of 0.8 µm or better to prevent mechanical failure of the part due to cracks initiating on its surface [Olakanmi2015].

In comparison to the overall part surface roughness, that is particularly difficult to measure during the build process, the local top surface roughness can be estimated for each layer after its processing. Further, the layer-wise surface roughness is a key feature to evaluate the results of laser-material interaction during a build process and provides a useful indication of the part quality [Alrbaey2014].

The roughness of LPBF surfaces mainly results from the layer-wise build process using overlapping laser tracks (e.g., multiple single laser welds), the applied process parameters, and incomplete melted material [Whip2019].

The following parameters have been investigated in literature for different materials and indicate a significant correlation to the surface roughness of LPBF parts:

1) Laser power --
2) Scan velocity +
3) Build orientation ++
4) Layer thickness ++
5) Hatch distance +
6) Scan strategy +

From the physical point of view, as the laser power increases, the size of the melt pool also increases [Gockel2014]. During the layer-wise build process, larger melt pools increase the intersection area between different tracks and therefore lead to a smoother surface. However, if the laser power exceeds a certain limit, the increased energy intensity may result in the formation of a high fluctuant keyhole that can introduce additional defects such as subsurface pores or spatters. In addition to increasing surface roughness, the appearance of **defects such as pores and lack of fusion can critically reduce the final density** of the part [Yasa2011]. Due to heat accumulation effects, the process remains highly dynamical and which leads to further shifts of the process window that is related to a sufficient build quality.

In industrial production lines, thus, a trade-off between increased defect probabilities due to physical process limits and further aspects of the part quality (e.g., surface roughness, part density) as well as **time for production** (scanning speed) must be found by the quality monitoring and optimization framework.

Furthermore, an **early detection of critical defects** in layer-wise manufacturing saves time as the build can be aborted which **saves machine time and resources**.

As a result, the digital enabler developed in this work can be used to tackle two important problems in today's additive manufacturing domain. First, the system enables the **layer-wise assessment of the part's surface quality** in terms of surface roughness and defective surface areas. Secondly, this information can be used in a further layer-wise **process optimization** step using model-based reinforcement learning, which allows to **operate** the **manufacturing process** close to its current **physical limits** and ensuring a certain **part quality** in a dynamical environment at the same time.

4.5.3 The Technological Perspective

Initially, the roughness estimation module described in Figure 44 is engaged in the acquisition of HDR image data of the surface of the currently processed layer.

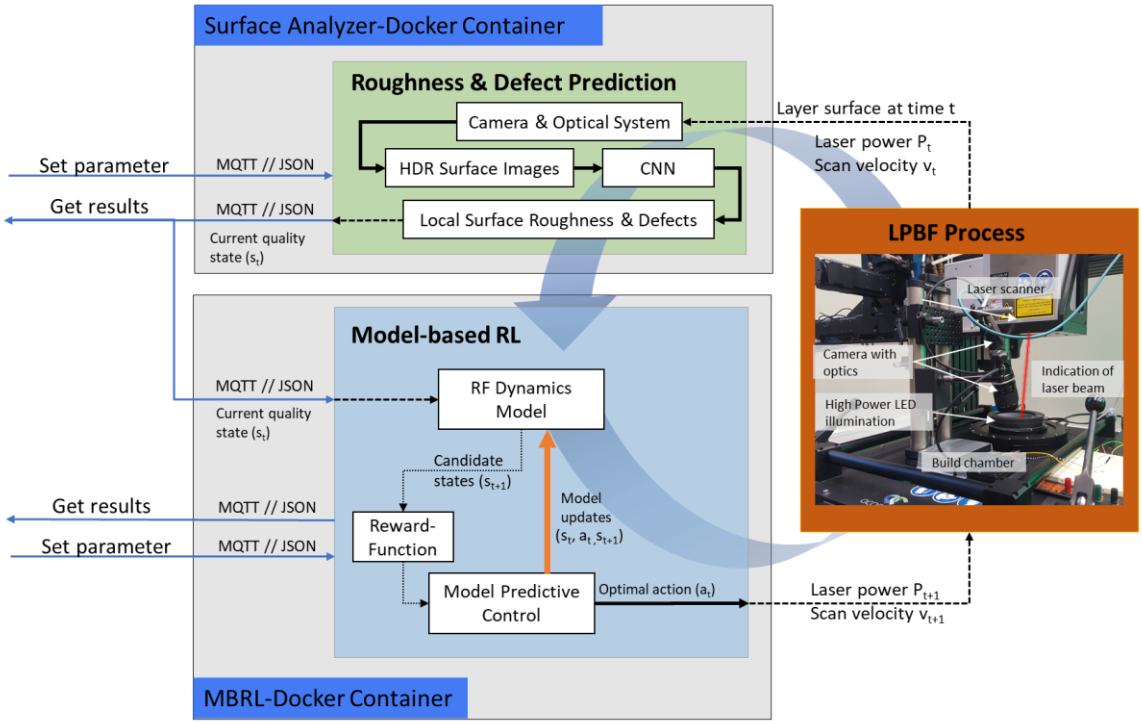

Figure 44: Overall framework for layer-wise monitoring and optimization of LPBF processes.

The HDR technique allows to increase the range of luminosity that can be represented within a single image. This is particularly important for scenes that contain high brightness gradients that cannot be adequately represented by a camera's standard dynamic range. Due to the high reflectivity of the metallic surfaces in metal-based AM and its strong dependency on the direction of light incidence, some surface areas only lead to a low intensity in the acquired image. Other areas, however, appear highly reflective and show saturation of the affected pixel values.

After image capturing, the images are divided in small image patches, which are annotated based on the measured surface roughness and subsequentially used to train a convolutional neural network that assesses the surface roughness based on image patches. Although the origin of CNNs lays back in the 1980s, huge attention was recently given where high performance GPU implementation enabled to train complex CNNs with a high number of parameters that outperformed many other methods in the most important image recognition contests [He2015, Zhang1990]. CNNs can not only be used for image data, but they bring certain advantages to these applications, such as translation invariance through weight sharing, and local connectivity that takes the spatial structure of images into account [Cireşan2011].

The architecture used for this work is depicted in Figure 45 and consists of three types of layers, which are connected consecutively to create a deep neural network model: Convolutional layer, fully connected layer and pooling layer. In the convolutional layer, small filter kernels convolve over the input array to produce class specific filter responses as layer output.

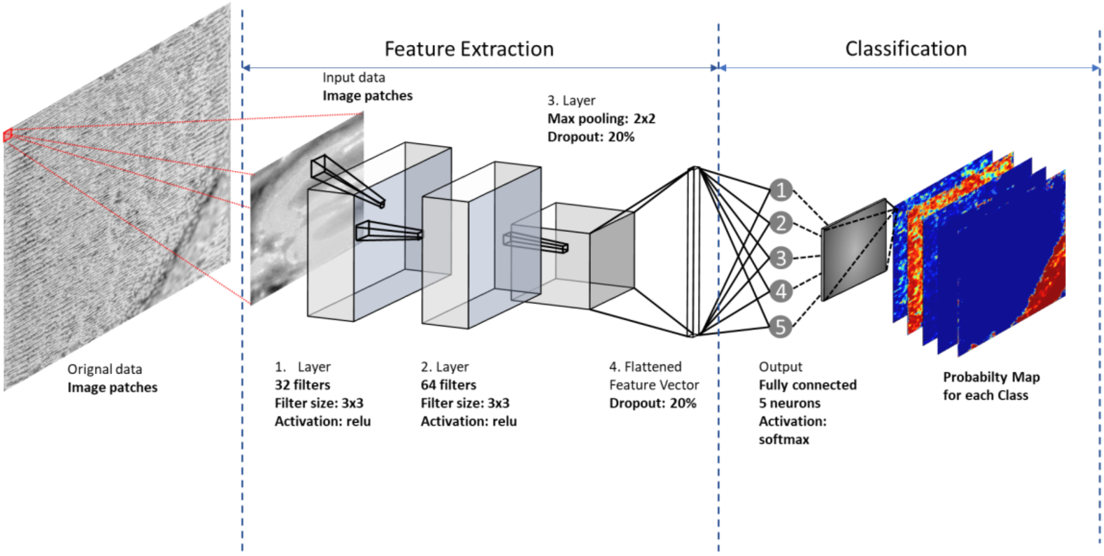

Figure 45: Architecture of the proposed CNN model used to predict surface roughness and surface distortion

After concatenating the output of the third layer, the flattened feature vector is used as input for the output layer, using fully connected nodes with softmax activation to classify the image patches into five different surface classes.

To generate spatially resolved roughness maps, overlapping image patches are extracted by a sliding window approach that resamples the original image using a step size of 16 pixels. The process of extracting image patches from the original layer surface image of a specimen is parallelized and optimized to run on GPU. The roughness class probabilities for the output layer are obtained by feeding 1024 patches per batch as input for the trained CNN-model. The calculation of the roughness map for a single LPBF part surface image (i.e., 3661x3617 pixels relating to 49,728 image patches) takes approximately 4.5 seconds on a NVIDIA GeForce 1080 Ti GPU, which is below the recoating time for each layer (i.e., time the slider needs to apply new powder on the processed surface).

The surface roughness analyzer module including camera and CUDA driver is embedded in a docker container allowing to train and run inference on different machines and independent of other components and modules. The parameters and classification results can be communicated via JSON messages over MQTT protocol.

An example of a JSON message the provides results from the surface roughness analyzer module combined with further process, machine and measurement system related information is shown in Figure 46.

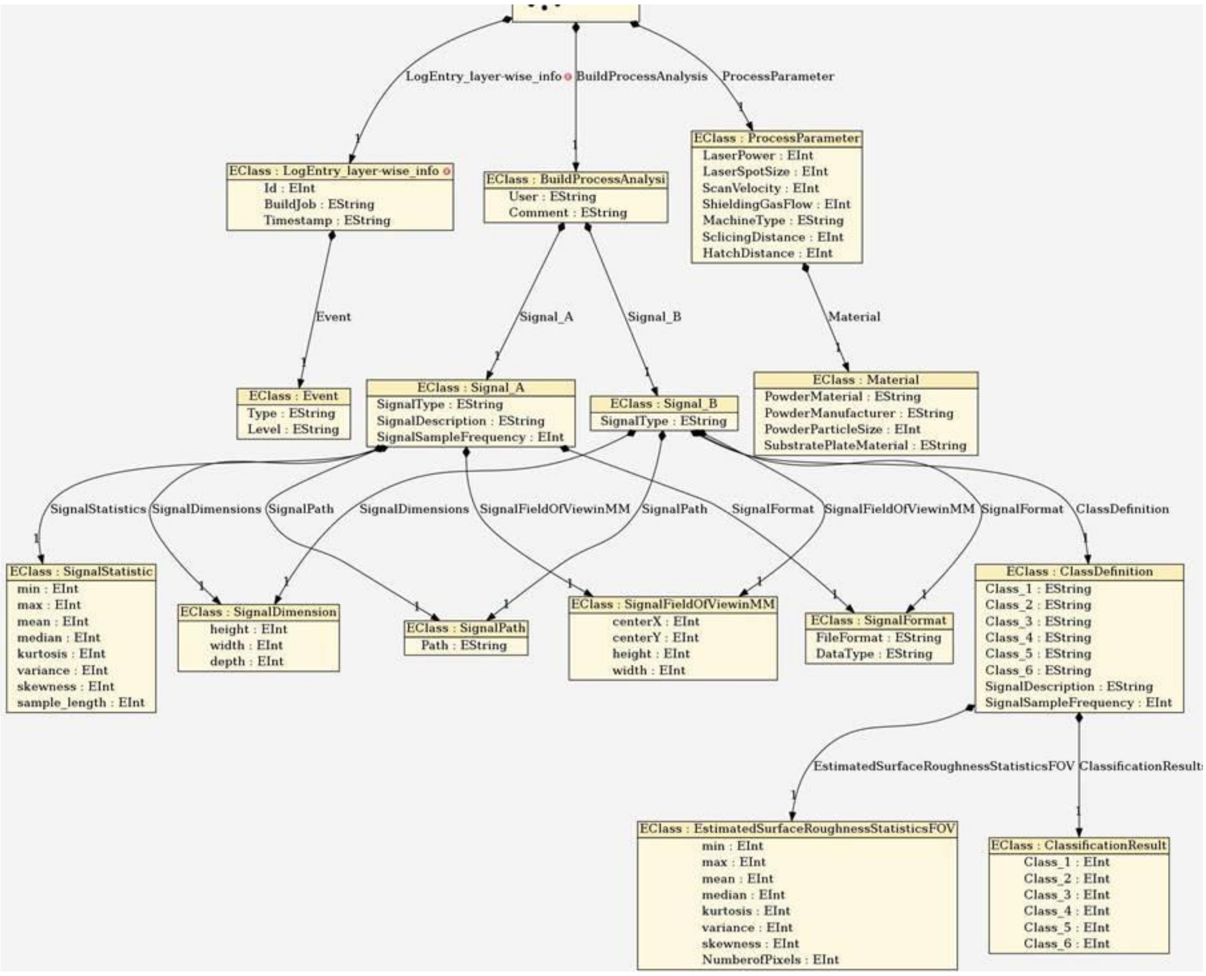

Figure 46: Structure JSON message to communicate surface classification results and further process related information

An example for the section "BuildProcessAnalysis", which represents results obtained from the image processing, is given in Figure 46. The classification results are reported in section "BuildProcessAnalysis/Signal_B/ClassificationResults", while the classes are defined in section "BuildProcessAnalysis/Signal_B/ClassDefinition", which might change for different machines, processes or martials. Further information regarding the surface properties is reported in section "BuildProcessAnalysis/Signal_B/EstimatedSurfaceroughnessStatisticsFOV" which can be used to quickly assess the minimum, maximum, mean and further statistics for the predicted roughness values within the given field of view (FOV) of the camera system. The machine specific field of view of the camera system is defined in section "BuildProcessAnalysis/Signal_B/SignalFieldofViewinMM" by its center position and height and width in millimeters. In addition, section "BuildProcessAnalysis/Signal_A/", contains information about the original image including the path where it is stored. This allows other modules to access the image data for further processing (e.g., visualization).

```
"BuildProcessAnalysis":{

    "Signal_A":{
        "SignalType": "2DSurfaceImageExtIll",
        "SignalDimensions": {"height":4156, "width": 3922, "depth": 1},
        "SignalFieldOfViewinMM": {"centerX": 0, "centerY": 0, "height": 120, "width": 120},
        "SignalFormat": {"FileFormat":"hdf5", "DataType": "uint8"},
        "SignalPath": {"Path":"deviceName://experiemnt_xy_2DSurfaceImageExtIll/layer_00001.h5"},
        "SignalStatistics":
                {"min":36, "max":3216,
                 "mean":1666.3, "median":1701.56,
                 "kurtosis":-3.1, "variance":15,
                 "skewness": 1.5, "sample_length": 560},


        "SignalDescription": "Raw surface image using external illumination (unprocessed) | one image per layer",
        "SignalSampleFrequency" : 1
        },
    "Signal_B":{
        "SignalType": "SurfaceRoughnessMap",
        "SignalDimensions": {"height":4156, "width": 3922, "depth": 1},
        "SignalFieldOfViewinMM": {"centerX": 0, "centerY": 0, "height": 120, "width": 120},
        "SignalFormat": {"FileFormat":"hdf5", "DataType": "uint8"},
        "SignalPath": {"Path":"deviceName://experiemnt_xy_SurfaceRoughnessMap/layer_00001.h5"},

        "ClassDefinition": {"Class_1": "Percentage of surface area whithin field of view with roughness Sa 0-2.9µm",
                            "Class_2": "Percentage of surface area whithin field of view with roughness Sa 3-4.9µm",
                            "Class_3": "Percentage of surface area whithin field of view with roughness Sa 5-7.9µm",
                            "Class_4": "Percentage of surface area whithin field of view with roughness Sa 8-15µm",
                            "Class_5": "Percentage of surface area whithin field of view with surface distortion",
                            "Class_6": "Percentage of surface area whithin field of view with powder coverage",

        "ClassificationResults":
                {"Class_1": 0.68,
                 "Class_2":0.20,
                 "Class_3":0.06,
                 "Class_4":0.01,
                 "Class_5":0.01,
                 "Class_6":0.04},

        "EstimatedSurfaceRoughnessStatisticsFOV":
                 {"min":1.56, "max":12.63,
                  "mean":4.23, "median":5.32,
                  "kurtosis":-3.1, "variance":3.64,
                  "skewness": 1.5, "NumberofPixels": 16299832},

        "SignalDescription": "Predicted surface roughness map for defined field of view (FOV)",
        "SignalSampleFrequency" : 1
        }
    },
    "User": "A. Peters",
    "Comment": "Example of log entry that provides integrated singal information for each layer"
  }
}
```


In a further step, the results from image processing system can are used for layer-wise process optimization. While several optimization technologies can be used to this task, a reinforcement learning approach, which allows ongoing learning and adaptation in highly dynamic environments was developed and employed.

For this use case, the agent denotes a software implementation of a RL method that choses LPBF process parameter based on a given state and a learnt policy. For that, the agent's current state s_t is defined as a tuple $s_t = (P_t, v_t, Sa_{mean,t}, \delta_t)$ comprised of the applied laser power P_t and scan velocity v_t as well as mean surface roughness $Sa_{mean,t}$ and percentage of defective areas δ_t for a given part surface at time t. The quality metrics $Sa_{mean,t}$ and δ_t are estimated by the CNN-model explained above. In each step, the agent can choose from a defined set of action describes as follows:

$$\mathcal{A} = \begin{Bmatrix} (P_{none}, v_{down}); (P_{none}, v_{up}); (P_{down}, v_{none}) \\ (P_{down}, v_{down}); (P_{down}, v_{none}); (P_{up}, v_{none}) \\ (P_{up}, v_{down}); (P_{up}, v_{up}) \end{Bmatrix} \quad (4)$$

where each element in the list represents an action tuple that consists of a possible action value for laser power P_{t+1} and scan velocity v_{t+1} to be applied in the next layer. Each process parameter can have the value up, down, or none which represent the action of increasing, decreasing, or applying no change at all with respect to the given process parameter.

Different RL algorithms can be used to estimate the optimal policy for choosing an action in a given state. The policy can be learnt with or without having a model that approximates the environment, referring to model-based and model-free RL respectively. Both approaches face their own challenges, but each offers unique advantages. Generally, model-based approaches show higher data efficiency and faster learning compared to model-free methods, which on the contrary can be used for a variety of applications and avoid the incorporation of model errors into the learnt policy [Nagabandi2018, Polydoros2017].

Since model-free learners require a larger number of training examples, their attractiveness for real-world applications such as laser materials processing is reduced. Without a corresponding simulation in which the agent can interact with the modeled environment to derive an effective control strategy, model-free learning appears to be disadvantageous.

In MBRL, the system dynamics are modeled as a function $\hat{f}_\theta(a_t, s_t)$ to estimate the next state s_{t+1}. The function is parametrized by θ and can be formulated as a regression task within the scope of supervised learning that maps a given state s_t and action a_t at time t to predict the subsequent state s_{t+1}at time t+1.

The agent used in this scope receives a negative numeric reward for the current state $s_t = (P_t, v_t, Sa_{mean,t}, \delta_t)$ if the percentage of surface defects δ_t detected by the CNN in the current surface image is higher than 10 %. At this point, a human expert would have to decide whether the build process should be stopped and modified or aborted. From the agent's perspective, the optimization episode terminates with a negative reward of -10,000 for the recent action applied.

A positive reward of 2,000 is given for a predicted roughness value $Sa_{mean,t}$ below 4 µm in combination with defective surface areas smaller than 10 %. At this point, the optimization episode is terminated, as the current state is considered optimal.

The optimization routine is applied after each layer based on the results obtained from the image-based surface analyzation system and the current process parameter applied on the machine.

4.5.4 Use in QU4LITY Pilots

The new approach based on HDR imaging combined with CNNs and model-based RL for inter-layer quality optimization of LPBF processes was evaluated on laboratory trials using LPBF machines at Fraunhofer ILT. The targeted quality metrics in this scope (and for the PRIMA pilot) are the current layer's surface roughness as well as the percentage of distortion on the layer's surface. Although this is a preliminary study, the intermediate results indicate that the framework has the potential to be successfully applied in industrial LPBF processes. The following experimental results are encouraging to continue and improve the demonstrated concept:

1) Surface roughness classification based on optical imaging and deep neural networks outperforms a classical ML approach using statistical texture features under the same image resolution and dynamic range conditions by more than 20 % in F1-Score.

2) HDR imaging increases the classification performance by more than 9 % compared to its low dynamic range counterpart. Experiments indicate that an image resolution of at least 5.66 μm/pixel is required for roughness classification accuracies greater than 80 %.

3) Based on measured surface roughness data, the negative correlation between surface roughness and volumetric energy density could be reproduced using the image-based roughness predictions (see Figure 47).

4) Moreover, the experimental evaluation also supports the assumption that a low surface roughness correlates with high component densities (see Figure 47).

5) For 21 unknown LPBF surfaces, the proposed MBRL approach finds optimal process parameters resulting in high rewards and a low surface roughness of 3.38 μm (average for 21 parts) obtained faster than with the Q-Learning reference implementation. At the same time, the MBRL effectively avoids actions that would result in a high percentage (>10 %) of predicted defective surfaces and thus be penalized by negative rewards.

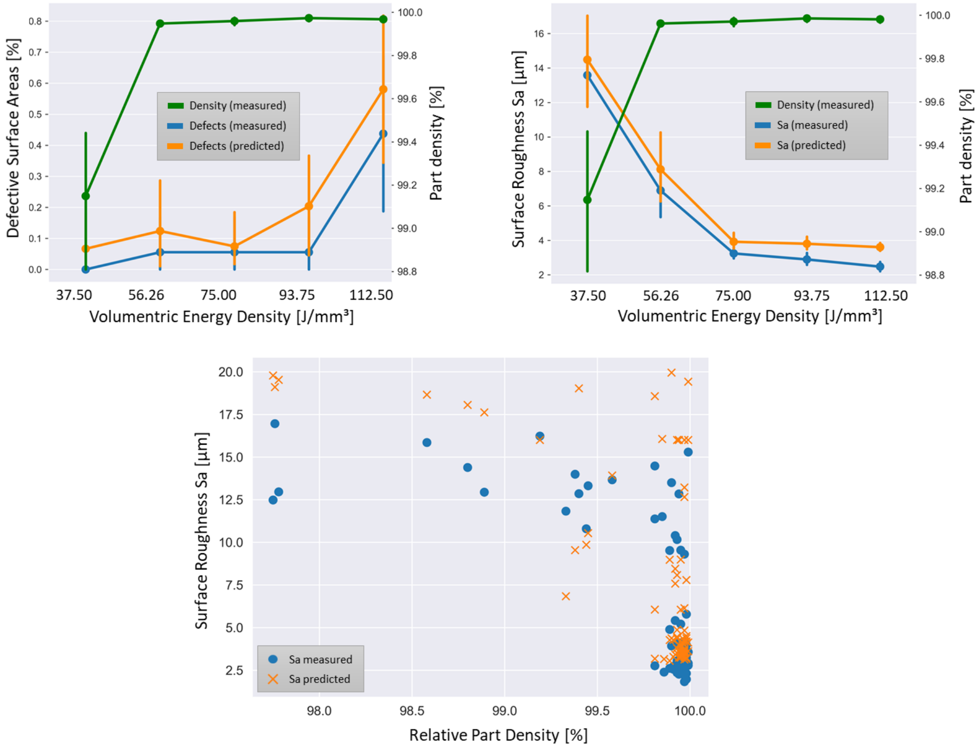

Figure 47: Correlation between (prediction) surface roughness, (predicted) surface defects, part density and energy input based on 88 LPBF cube surfaces.

In Figure 48, examples of spatially resolved roughness predictions for different 3D printed surfaces are depicted. In part 5, a large defective surface area has been predicted (>10% of all pixels within the image are predicted as defective), which would subsequently flag a warning or immediate process stop based on the final implementation and decision of the process owner (PRIMA or one their customers).

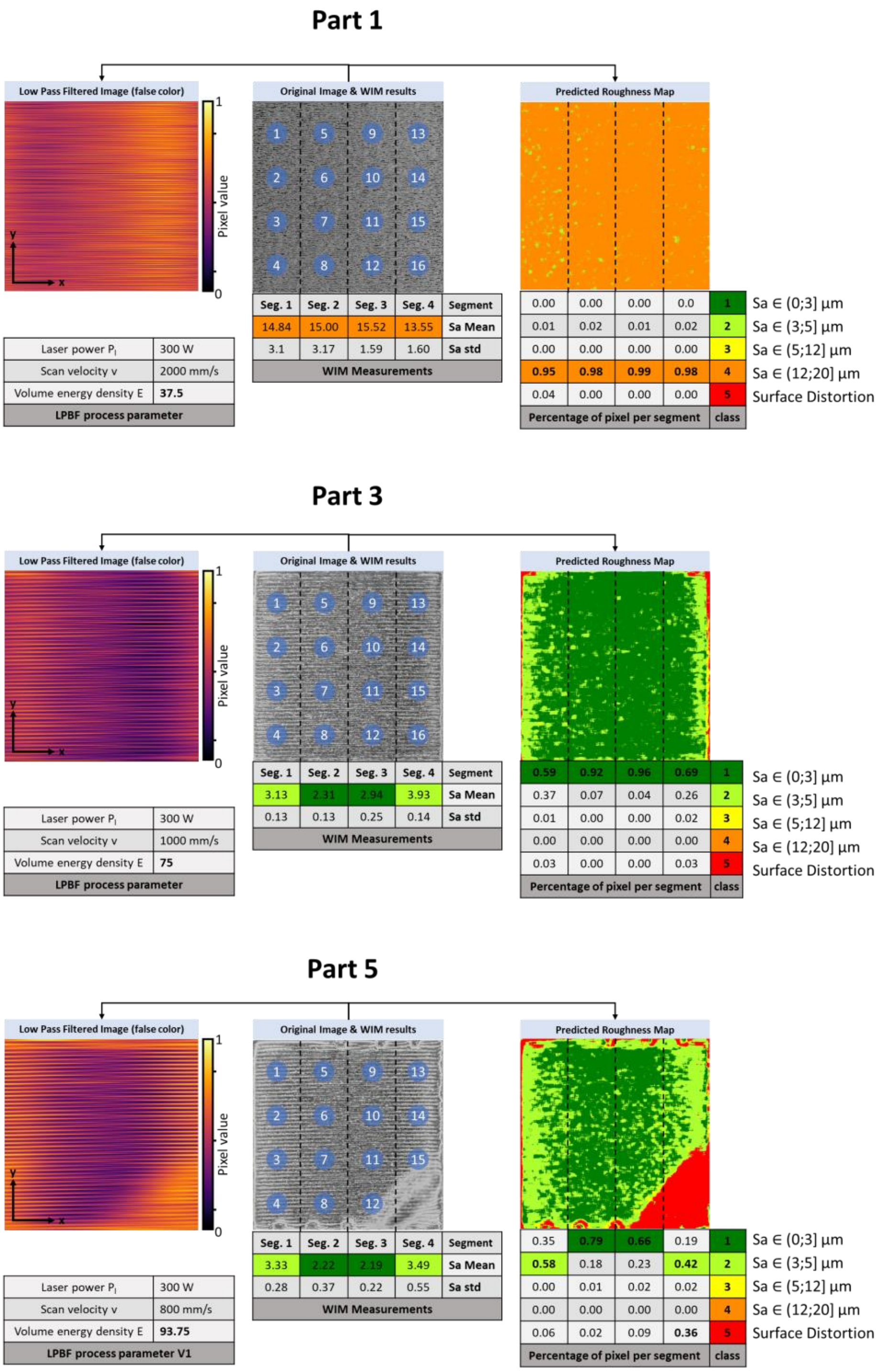

Figure 48: Results of the surface roughness and defective areas prediction module

Using model-based reinforcement learning, the predicted surface roughness and the predicted amount of defective surface areas are leveraged to optimize part quality. For that, 50% of the experimental data (44 part surfaces) is used to train the MBRL-agent to choose optimal actions (i.e., process parameter for the next layer).

After the training procedure, the MBRL approach is evaluated on 21 unknown top surfaces of LPBF cubic samples. After the mean surface roughness $Sa_{mean,t}$ and the percentage of defective surface areas δ_t are predicted by the CNN, the agent must choose an optimal action or action sequence to achieve the optimization goal. An example for an optimizations sequence is given in Figure 49.

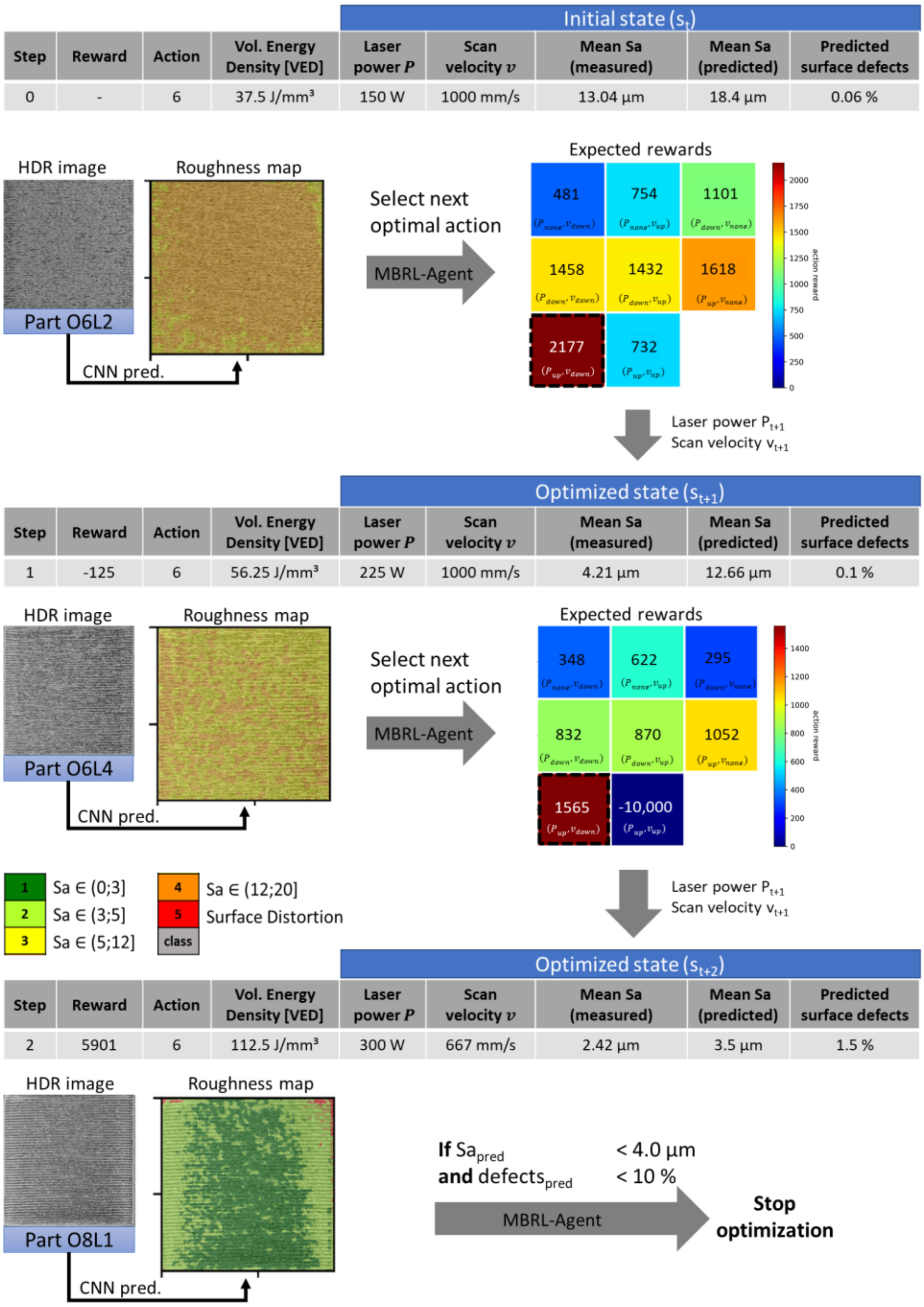

Figure 49: Example of an inter-layer roughness optimization sequence based on MBRL and CNN-based image processing.

The trained MBRL agent starts with a state representing low laser power (150 W), medium scan velocity (1000 mm/s) and a measured mean surface roughness of

13.04 µm. After assessing the current state $s_t = (P_t, v_t, Sa_{mean,t}, \delta_t)$, the agent choses the action with highest expected rewards, estimated by using the learned dynamics function in combination with a defined reward function.

The MBRL approach is able to reduce the average surface roughness during two optimization steps to 2.42 µm by increasing the laser power to 300 W and decreasing the scan velocity to 667 mm/s. The roughness prediction suggests slightly higher roughness values in both cases. The predicted amount of defective surface areas increases from 0.06 % to 1.5 %, which is probably due to the uncertainty of the CNN-model regarding the prediction at the surface edge.

An overall evaluation of the 21 LPBF surfaces optimized by the approach is given in Table 5. The mean surface roughness Sa of components before the optimization was 10.40 +/- 5.44 µm.

Table 5: Overall results for 21 cubic surfaces evaluated using the trained CNN in combination with the MBRL agent.

Criteria	Average (21 parts)	Standard deviation (21 parts)
Initial surface roughness $Sa_{mean,ini}$	10.40 µm	5.44 µm
Post optimization surface roughness $Sa_{mean,opt}$	3.38 µm	0.28 µm
Post optimization surface defects δ_{opt}	2.9 %	1.2 %
Required optimization steps	1.97	0.89
Reward after optimization	5851	823

While the outcome of this work appears promising, future work should address the real-time implementation of the proposed framework at PRIMA facilities that enables the quality assessment and process optimization during the build of more complex components on different machines. Additionally, the system's ability to reliably communicate with other modules and tools within the PRIMA pilot needs to be improved and tested. Nevertheless, the demonstrated system can potentially be used with small adaptations for quality assurance and optimization of many different 3D printing processes and machines. While in its current state, the image-based roughness estimation module requires approx. 4.5 seconds to infer a roughness map from a high-resolution image, further inference optimization (e.g., using TensorRT or other CNN architectures) could enable surface quality prediction in real-time. This would allow the system to be applied not only for inter-layer quality assessment and optimization but also for faster production processes represented by other pilots.

4.6 Improved Failure Classification Enabler

4.6.1 Digital Enabler Overview

The IPS (Institute of Production Systems) of the TU Dortmund University proves a vast experience in the field of data analytics, machine learning, quality assurance and machine documentation. During several research and industrial projects, the IPS has proved its expertise in every field itself as well as in the combination of the mentioned topics. By combining quality analysis with machine documentation and data analytics the IPS already proved the application of a holistic optimization approach in several projects to ensure an efficient quality management and production planning as well as to empower companies in the field of machine learning and Predictive Maintenance.

4.6.2 The Business and ZDM Perspective

At Amberg, Siemens' Digital Factory Division is manufacturing its 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.

As typical in electronics production, a high effort is put into non-value adding processes for testing the manufactured products to guarantee high product quality. As of today, all products are tested in all testing steps (marked orange in Figure 50).

Figure 50: Circuit board manufacturing line with test stations

At Solder Paste Inspection (SPI), the volume and geometry of the solder paste are measured and compared with predefined threshold values provided by the machine vendor. If the measurement of any single solder pad is not within the tolerance, the location of the pad is registered, and the panel consisting of four circuit boards is classified as "DEFECT". It is then automatically extracted from the actual manufacturing flow and is moved to a buffer, and an operator is notified to check the measurement results. After manually checking the effected soldering pad, the operator classifies it as "GOOD," leading to the result that the inspection is "PASSED." Classifying as "DEFECT" results in removing the panel from the line to perform rework. The current inspection process is illustrated in Figure 51.

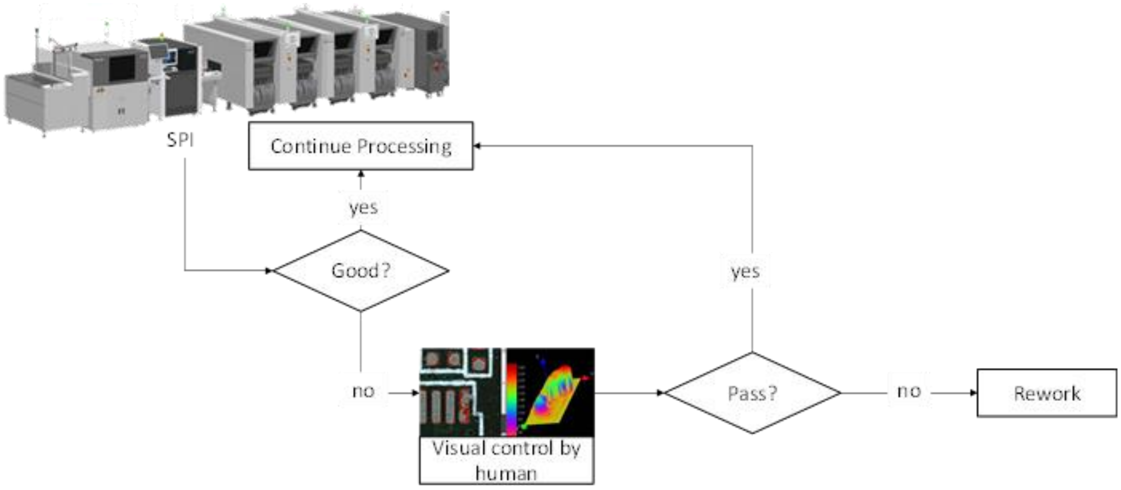

Figure 51: Inspection Process

As illustrated in Table 6, the "PASSED" result is treated as a pseudo error.

At every test station a similar procedure is in place today, leading to a comparable situation: the pads are tested compared with predefined threshold values provided by the machine vendor, being classified as "DEFECT" and moved to a manual re-testing sub-line/buffer.

Overall, this results in a lower as possible efficiency of the line. Since pseudo errors are a significant percentage of the overall test results and require additional work for the operator, gaining deeper understanding of (pseudo) failures is a necessary basis for further optimization of the entire process chain.

Machine Test Result	Operator Test Result	Overall Result	Interpretation
GOOD	---	GOOD	No Error
DEFECT	GOOD	PASSED	Pseudo Error
DEFECT	DEFECT	FAILED	Rework

Table 6: Solder paste inspection station – Error table

These pseudo errors are a significant percentage of the overall test results. Since each pseudo error requires an operator to perform additional evaluation of the test results, the aim of the business scenario is the reduction of pseudo errors to reduce unnecessary evaluation steps. In this context, the propagation of defects or product properties throughout the entire process chain is of particular importance.

4.6.3 The Technological Perspective

The module uses automatically recorded process data from SPI. These include geometric measurement data of the applied solder paste at defined pin positions. Figure 52 represents the data flow of the module.

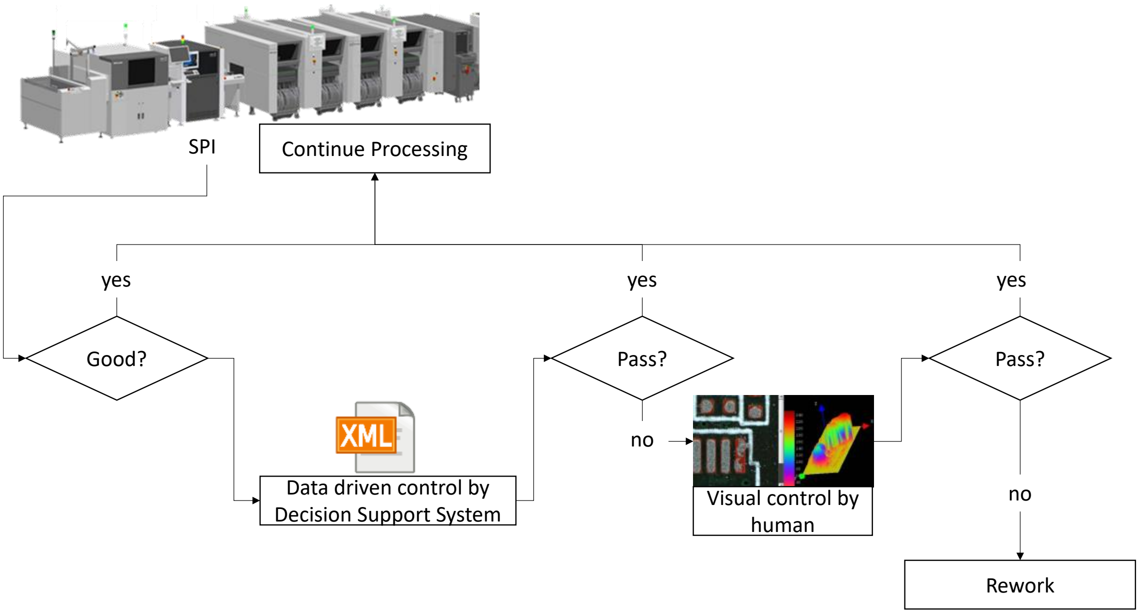

Figure 52: Integration of the Module in the Inspection Process

The objective of the module (mentioned as Decision Support System (DSS) in Figure 52) is an objective identification of defective pins. For this purpose, supervised and unsupervised machine learning algorithms are used: Machine Learning Algorithms are used to build up models which can differentiate between GOOD and DEFECT on the basis of recorded measurement values. The decision is made within the SPI: If a product is classified as DEFECT, this product is rejected and reworked. In case of a classification as GOOD, the tested product is further processed according to Figure 50.

In general, the Module can use automatic measurement results to classify DEFECT and GOOD products. By visualizing the decision made by the module and comparing them with the real expert labels, the module can be validated to ensure long-term use in the product line.

As described above, products rated as DEFECT are verified by humans. It is planned to integrate the module into the SPI, so that if a DEFECT product occurs, the module is used to distinguish between GOOD and DEFECT instead of humans. In case the module evaluates a product as DEFECT as well, only these products will be verified by humans (as illustrated in Figure 52).

The Module is established by different "function-blocks" within Python. These "function-blocks" are connected to each other as illustrated in Figure 53.

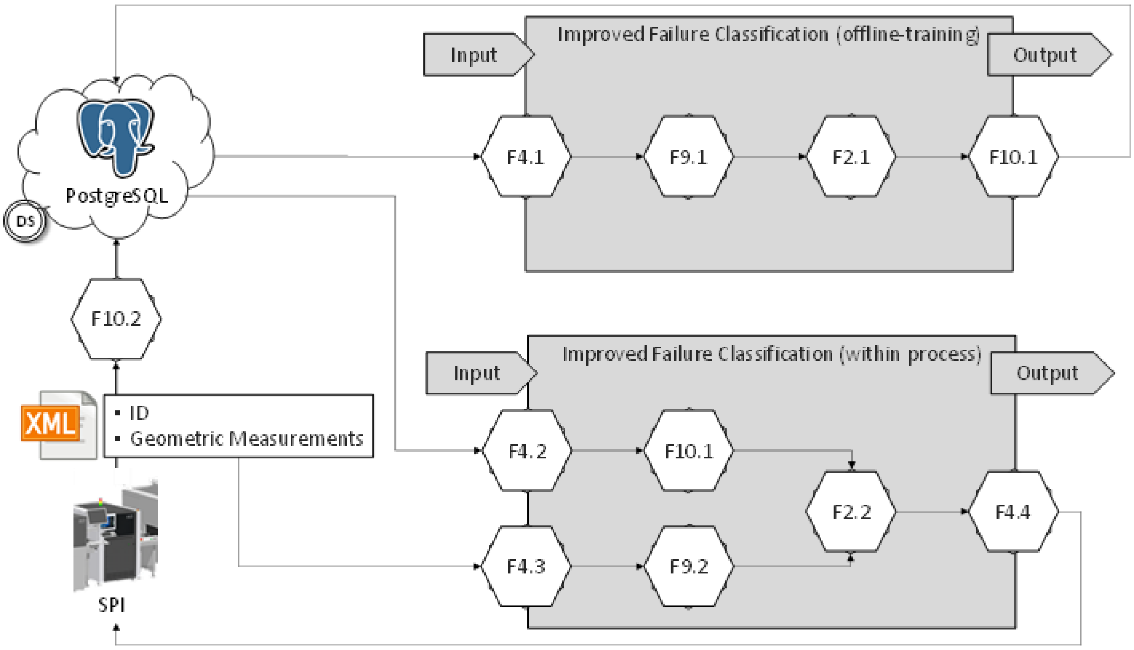

Figure 53: Insight in the Module

A description of the presented "function-blocks" in Figure 53 is given in Table 7.

ID	Function	Description
F2.1	Analytics	Train and optimize Machine Learning Models
F2.2	Analytics	Use Machine Learning Models to determine good and defect products
F4.1	Communication	Retrieve data from database
F4.2	Communication	Retrieve stored model
F4.3	Communication	Retrieve data via XML
F4.4	Communication	Send results to SPI / humans
F9.1	Preprocessing	Data cleaning / selection of features / feature generation
F9.2	Preprocessing	Data cleaning / selection of features / feature generation (identical to F9.1)
F10.1	Storage	Store trained model in cloud
F10.2	Storage	Store data in Database

Table 7: Used Function Blocks within the Module

Within these "function-blocks", different libraries are necessary, as listed in Table 8.

Package name	(short) Description	Further Information
pandas	data analysis and manipulation tool for DataFrames	Link
numpy	mathematical functions	Link
psycopg2	PostgreSQL database adapter	Link
pandas.io.sql	Read SQL query or database table into a DataFrame	Link

Package name	(short) Description	Further Information
feather	portable file format for storing Arrow tables or data frames	Link
scikit-learn	Library of ML-Algorithms	Link
xml.etree.ElementTree	API for parsing and creating XML data	Link

Table 8: Python Libraries Used for the Development and Operation of the Enabler

4.6.4 Use in QU4LITY Pilots

The general objective of the "Pseudo Error Reduction" scenario / use case is to improve the overall product quality rate as well as to increase the testing efficiency. In the chosen scenario, the vision is to reduce the pseudo error rate significantly by changing from predefined static testing parameters to a refined quality assessment procedure based on processing parameters and prior testing results. It is furthermore intended to gain experience and improve the transparency of the manufacturing process by applying data mining and machine learning.

The developed module was applied within a SPI in Surface Mount Technology (SMT) assembly in electronics manufacturing. SPI is conducted in automatic inspection systems, which measure the solder paste position and geometry on printed circuit boards (PCBs) and capture images of potentially defective solder pads. While the evaluation of the measured values with the corresponding specifications is fully automated, the images are interpreted manually by an operator.

For the application of the proposed method, the scope was limited to one SPI facility and the defect pattern of an insufficient amount of solder paste. For the modelling a historic data set of 34,278 inspected parts, where only 92 parts (0.27%) were defective, was extracted, transformed, and processed accordingly.

Based on the seven SPI measurements a OCSVM model (RBF kernel, nu = gamma = 0.001) was trained and 0.27% of the population were re-labeled as defects. Based on the re-labeled data set a DT model was trained, optimized, and validated in a 10-fold cross-validation. Table 9 shows the confusion matrix of the predicted label compared to the true expert label for the evaluation of the achieved classification performance.

Accuracy 99.91%		**True expert label**		
		Defect	**No defect**	**Class precision**
Predicted label	**Defect**	27	8	77.14%
	No defect	1	10,248	0.01%
	Class recall	96.43%	99.02%	

Table 9: Confusion matrix of the final classification result (DT, maximal depth = 4, split criteria = gini index, minimal leaf size = 1)

As the results show, 99.91% of the predicted labels comply with the expert assessment. Therefore, it can be concluded that the SPI verification result cannot only be received by image-based manual assessment but likewise by using the proposed method which evaluates the recorded measurements in a multivariate way.

However, since it cannot be ensured that new, previously unknown defect patterns are detected by the model with sufficient certainty, its deployment as a decision support system for the operator is of great interest.

4.7 IKCloud+ anomaly detection

4.7.1 Digital Enabler Overview

ikCLoud+ is a MLOps based solution, which has defined workflows that can evaluate different prediction models from a datasource, selecting the best performing one for production deployment. Furthermore, all steps of the workflow are tracked, and the user can check the results of each evaluated model at any point in time. Current enabler focuses on anomaly detection, providing an Autoencoder based prediction model solution.

The workflow consists of an Apache Airflow workflow definition that schedules a periodic ETL executed by Spark storing processed data in an HDFS system. In addition, candidate models are defined in Jupyter notebooks, and different pipelines are defined in Kubeflow, who then runs training and testing of the models using the datasets generated in the ETL phase.

This way, the enabler provides an out of the box solution for anomaly detection in addition to the necessary tooling to define new prediction models to extend the enabler to different ZDM aspects, that are trained and tested with minimal configuration.

4.7.2 The Business and ZDM Perspective

The Zero Defects concept should be viewed as a quest for perfection to improve quality in the manufacturing process. True perfection might not be achievable but at least the quest will push quality and improvements to a point that is acceptable.

In this line strong manufacturers have evolved their workflows and tooling in order to minimize the defects within their production chain. But stepping forward to the next level in quality improvement goes through augmenting the availability, performance and quality of their installations, getting an optimum cost per part ratio. To achieve this, manufacturers like FAGOR ARRASATE need to apply ZDM concepts and improve the digital platform and interoperability of the digital platforms between them and its customers.

FAGOR ARRASATE has a long experience in delivering press machines as well as providing the building blocks of such lines. A press machine is the product par excellence of FAGOR ARRASATE. A typical press machine is composed by two rigid platforms (head and base), a bed, a ram, and a mechanism as well as all the other surrounding components that guarantee the full automation and process control (temperature, pressure sensors, etc.).

Historically, machine tool manufacturers have not had any information of the machine behaviour once they were working at the customer facilities. Maintenance actions by the machine tool supplier, where mainly started by a customer's call and where mainly related to corrective actions once the failure had already happened.

Currently many condition issues on the machine are detected afterwards, they appear when a quality matter is detected on the forming parts or when a machine component is damaged, causing even machine stoppage. These problems are usually fixed by machine adjustment or changing programs or forming process parameters.

Consequently, the only way to avoid future problems is by preventive maintenance or machine adjustment actions. These are carried out either by the machine owner itself or external services which are sometimes delivered by FAGOR ARRASATE.

In QU4LITY project, FAGOR ARRASATE equips a press machine with a SMART CONNECT technology that provides data from the machine, to the owner and to the machine supplier. Within the context of Zero-Defect Manufacturing, ikCloud+ provides data analytics capabilities to the pilot focusing on the anticipation and avoidance of failures. This predictions aid reducing downtimes, thus assuring quality of the end products.

4.7.3 The Technological Perspective

ikCloud+ is composed by a set of top edged machine learning tooling, chained together to enhance existing systems providing them data analysis capabilities with a minimum effort. ikCloud+ includes the necessary tooling to develop and test new models or datasets. The enabler includes within its workflows a prediction model based on autoencoders, that has been developed and provided as a core inference system to aid predictions.

4.7.3.1 Autoencoders

The domain of a problem determines the type of data to be handled by a solution. As previously mentioned, current pilot's domain focuses on manufacturing presses that delivers data in time series form, being this periodic, that can be grouped in cycles.

Autoencoders have demonstrated their suitability to model time series in an unsupervised manner.

Autoencoders (AE) are neural networks trained in an unsupervised way in order to reconstruct the input data. The structure of the AEs consists of two symmetric neural networks: the encoder and the decoder. The type of the Neural network can be of any type, either multilayer perceptrons, convolutional neural networks, LSTM, etc.

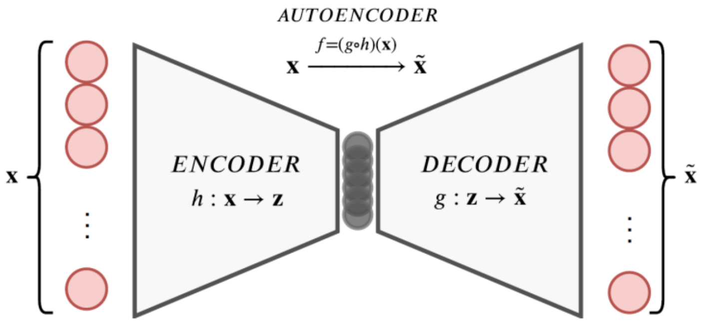

Figure 54: General structure of an AE

Figure 54 depicts the structure of an AE, representing the left part the encoder and the right one the decoder. In one hand the encoder can be defined as a function $x \in \mathbb{R}^n \rightarrow z \in \mathbb{R}^d$, where $d < n$ to compress the input data into a lower dimensional latent vector z, containing the most significant features of it. On the other hand, the decoder is defined as a function $z \in \mathbb{R}^d \rightarrow \tilde{x} \in \mathbb{R}^d$ that uses the compressed output from the encoder and decompresses into features that match the original input data x, trying to minimize the difference. This difference is known as loss function $\mathcal{L}(x, \tilde{x})$, called reconstruction loss, being the Mean Square Error (MSE) and Mean Average Error (MAE) the most used ones.

Bi-LSTM

As mentioned in previous sections, LSTMs are Recurrent Neural Networks including some cells in their internals providing some control over the temporality of the data they process, making them suitable for processing data inputs of long periods, thus avoiding the vanishing gradient and exploding gradient problems.

However, even if LSTMs are aware of the previous information of the data in a certain point in time, they lack information about the future that usually is important for accurate predictions. Bi-LSTMs address this problem, running inputs in both directions, from past to future and from future to past, called forward pass and backward pass respectively.

Attention

When input time series used in LSTMs are long sequences, LSTMs do not correctly encode the input, as they miss some significant features. For this, attention comes into the stage to allow the decoder to identify which parts of the input sequence are relevant to the output, and what parts of the encoded vector are relevant to select the appropriate output. This is done by adding a context vector that contains the weights of each element in the input-output, representing these weights the significance of each piece of the input. Using this significance weights, a more accurate prediction is made by the AE.

Bi-LSTM Autoencoder with Attention

The proposed inference solution for ikCloud+ component is a Bi-LSTM Autoencoder with Attention. This model has been chosen because of the nature of the data, being this cyclic long term time series. The setup consists of 2 hidden layers with the addition of attention module, being this setup the best suited among the evaluated models. The autoencoder provides feature extraction from the input data that can be used for data analysis, being the anomaly detection the one to focus on.

Having the lack of tagged validation data, the used approach is to evaluate the autoencoder with a previously known dataset. This dataset has been already used in IKERLAN for other model development and consists also of cyclic time series with anomalies within it. This type of anomalies has 4 different categories, but for the current use case the only classification being made is whether a stroke is an anomaly or not.

Once the model has proven its performance on the well-known dataset, the model has been adapted for the pilot input data structure and evaluated.

Data pre-processing (ETL)

In order to provide consistent data that the model can use in the train phase, the input data is pre-processed on 2 steps that are performed in the ETL phase:

- Valid data filtering through the removal of invalid strokes.
- Feature normalization through a MinMax scaler setting each feature individually such that it is in the [0-1] range.

Model Training (TRAIN)

When the data is ready to be processed, the model is trained making use of the 80% of the input data, leaving the rest to validate the model. This training phase is made through a variable number of epochs defined in the model. Current setup stablishes an adaptative number of epochs, that starts from 1 epoch and stops whenever in 20 epochs there is no loss reduction. In addition, a window is set, which determines the number of values included in each cycle. Once the autoencoder is trained, an evaluation phase is performed with the test data, which will infer the new feature vector *z*.

Anomaly detection

The way to identify anomalies with the AE is done by clustering encoder’s output (z) in 2 different groups that identify anomalies and correct behavior. This clustering is done applying OCSVM clustering algorithm. By applying this, we assume that the AE’s predictions in a correct functioning of the press will be all close to each other, having similar feature vectors. In case of any anomaly, this prediction is predicted with a higher error than normal, classifying it outside the correct functioning cluster, thus being an anomalous prediction.

4.7.4 Use in QU4LITY Pilots

The ikCloud+ platform was integrated into the FAGOR ARRASATE use case to provide a ZDM solution based on anomaly detection in order to be able to identify press malfunctioning on early stages. FAGOR ARRASATE's production press machine with FALINK system was selected as the data source. This machine provided data for 21 different dies, and each die had measurements grouped by degrees of the press axis. Enabler's output detects anomalies by clustering the predictions in 2 different groups, correct functioning and error. However, if future data contains different labels classifying different anomalies, this can be fine-tuned in order to predict the type of anomaly.

4.7.4.1 The dataset

The data consists on 13 different variables, and separated among 22 different dies. For each die a different autoencoder has been trained, this is due to the fact that the range of the data varies significantly depending on the die being used, leading to non-stationary data series. So a single autoencoder would be unable to adapt to each die. Then, for each die, cycles were identified based on the axis degrees, being the start of a cycle the point where axis degree is 0, leaving each cycle with a total of 378 timesteps. In cases where there are missing values, data has been completed with zeros.

4.7.4.2 Autoencoder definition

The autoencoder was defined with 2 hidden layers, with an input shape of 189x26 instead of the original sized 378x13 cycle. This is because each cycle has been merged into a double sized vector, in order to improve speed in the LSTM encoder training phase. The number of hidden units used in the autoencoder are 128 for the first layer, 64 for the second layer and 64 for the decoder layer. The loss used is the MSE and for updating the weights the ADAM optimizer has been used.

4.7.4.3 Training and evaluation

Once the autoencoder is trained with the 80% of the data, the rest of the data is used to validate the model. To evaluate the model and make predictions, only the feature vector extracted by the encoding phase is used, discarding the decoding part of the AE. This way, the full encoded data can be depicted by applying a PCA over the feature vector reducing the dimensionality of the result. Below a 2D representation of the encoded feature vectors is shown.

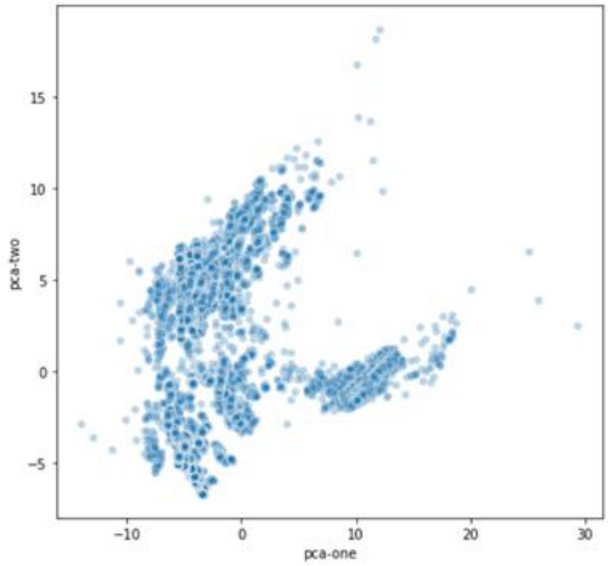

Figure 55: Encoded feature vectors´ 2D representation after applying PCA to the encoder´s output

However, as there are no tagged data to check the AE's accuracy, visualizing AE's predictions over different variables is also a good indicator of the behavior of the model. Below AE's reconstruction vs real input data is shown for the variables press_vel and mm_power.

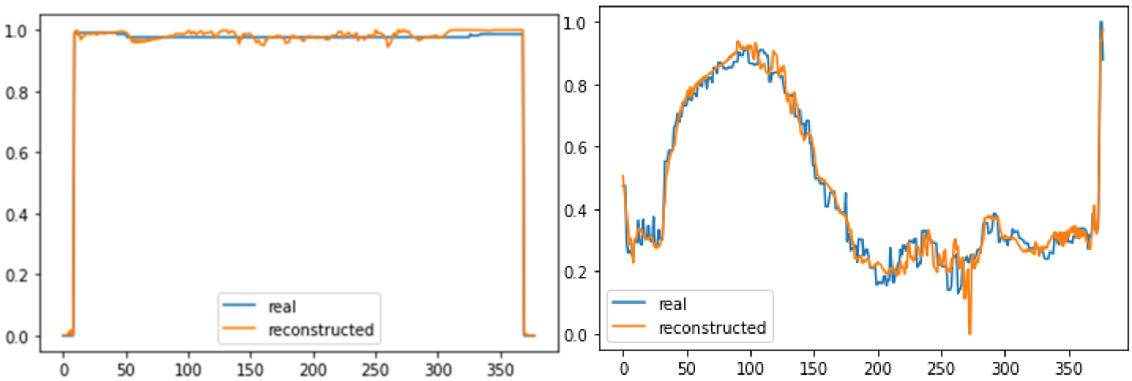

Figure 56: Real input data versus decoders output for variable press_vel(left) and mm_power(right)

4.7.4.4 Anomaly detection

The way to identify anomalies with the AE is done by clustering the encoder's output (z) in 2 different groups that identify anomalies and correct behavior. This clustering is done applying OCSVM with a gamma parameter of 0,001, meaning that at least 0,1% of train data of the clustering model is an anomaly. The resulting space by clustering encoder´s output is shown in Figure 57 and

Figure 58.

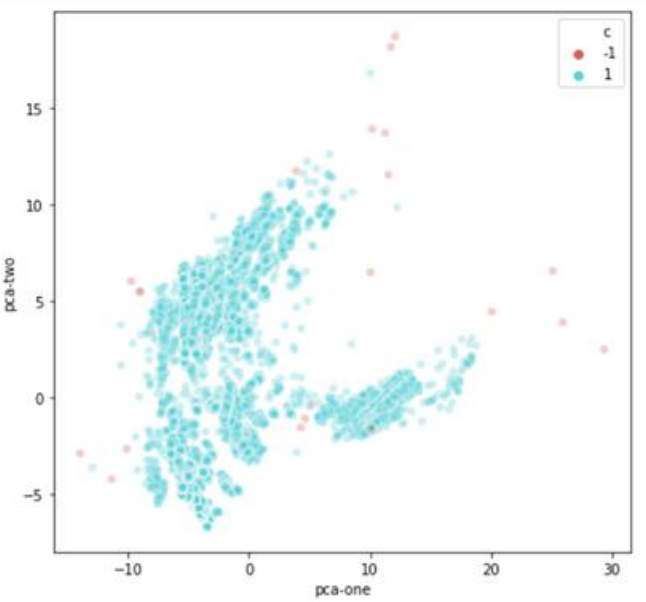

Figure 57: 2D representation after clustering Encoded feature vectors

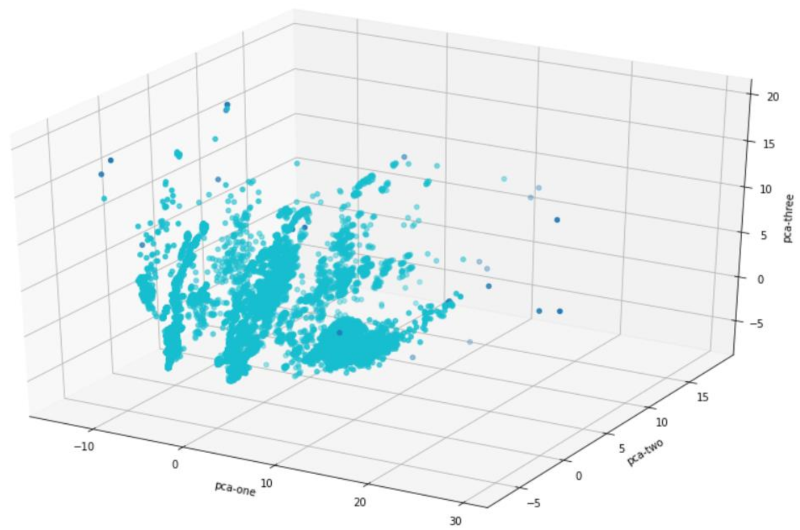

Figure 58: 3D representation after clustering Encoded feature vectors

It is usual that sensor data in industrial environment is not tagged. Current pilots' data is not a different scenario, and the available data has no malfunction tags. At IKERLAN this is a common scenario and the way to overcome this situation consists of testing the candidate models against two well-known in-house datasets. These are frequently used by IKERLAN and contain similar type of sensorial labeled data. The results of testing the candidate AE were promising and the solution need to wait until real tagged data is available, in order to validate the preliminary results.

Final results obtained by current solution will be made available in FALINK platform providing a set of REST interfaces focused on anomaly detection and Zero Defect Manufacturing. These interfaces are being implemented within WP7.

5. Conclusions

This deliverable has presented a collection of reusable digital components and libraries, which can be used for the development, deployment and operation of digital systems for quality management and ZDM in production lines. The presented set of reusable components are characterized as digital enablers and fall in two broad categories:

- **BigData and IoT Platforms** for quality management i.e., platforms that facilitate the management, storage and analytis of large data volumes that feature the four Vs (Volume, Velocity, Variety, Velocity). These platforms are also destined to support the automated collection of product quality related data in production environments.
- **Machine Learning Techniques** for quality management use cases i.e., data mining techniques that can extract product quality and ZDM insights from historical datasets about products and quality processeses. These techniques facilitate the production of knowledge for quality control and quality management processes leveraging on large amounts of production data. They include deep learning techniques.

The pertinence of these enablers to quality management stems from the fact that modern ZDM processes are data-driven and data intensive. This is also evident in the QU4LITY Reference Architecture, which specifies various BigData building blocks as core elements of ZDM systems and use cases. In practice, the development of the digital enablers that are presented in this deliverable has been driven by the need to support a variety of ZDM use cases spanning maintainance, product testing, and processes control. Specifically, the BigData and ML enablers of QU4LITY support the following set of use cases (see Section 4): (i) RUL Calculation/Estimation, towards supporting the intelligent management of assets, including practices like predictive maintenance that improve Overall Equipment Efficiency (OEE) and boost productiuon quality ; (ii) Fault Detection/Identification, which facilitate the timely identification of defects, faults, anomalies and other quality issues ; (iii) Determination of associations between production variables that lead to specific levels of quality towards either pursuing or avoiding production configurations that feature these associations ; (iv) Determination of process parameter settings that must be avoided; (v) Anomaly Detection of products product that deviate from high quality products; (vi) Product testing towards identifying whether products lie within specific thresholds specified by their manufacturer.

The deliverable outlines the functionalities of the prototype implementations of the presented digital enablers. Furthermore, there are demostrators for each one of the enablers. All the presented enablers have been deployed and used in at least one QU4LITY pilots in WP7. Moreover, some of the enablers have been used to support the QU4LITY open calls as well, which is indicative of their resuability and generality.

Also, some of the presented assets are included in the QU4LITY market platform in WP8 i.e., made available to the Industry 4.0 community. In this direction, the organizations in charge of the digital enablers have also organized a series of dissemination activities (e.g., webinars and digital talks) where the prototypes of this deliverable have been presented and relevant feedback has been received. This feedback will be considered as part of the future development and fine-tuning of the enablers. Likewise, all the enablers are exploitable assets of the QU4LITY project. The partners will pursue their sustainable development and wider use following the end of the project.

Overall, the present deliverable has presented a pool of reusable and exploitatable assets for digital quality management, notably assets that fall in the realm of BigData platforms and ML/DL algorithms. These assets have provided tangible value to most of the QU4LITY pilots and the to the open call experiments of the project. At the same they have enriched the products and research prototypes portofolios of the partners and have boosted their competitiveness.Therefore, the deliverables has essentially contributed to the project's and the partners' objectives as outlined in the QU4LITY DoA (Description of the Action).

References

[Akhila18] Akhila, C. V. (2018). A Survey on Collaborative Learning Approach for Speech and Speaker Recognition. In 3rd National Conference on Image Processing, Computing, Communication, Networking and Data Analytics (p. 199)

[Alrbaey2014] K. Alrbaey, D. Wimpenny, R. Tosi, W. Manning, and A. Moroz, "On Optimization of Surface Roughness of Selective Laser Melted Stainless Steel Parts: A Statistical Study," J. of Materi Eng and Perform, vol. 23, no. 6, pp. 2139–2148, 2014, doi: 10.1007/s11665-014-0993-9.

[Altenhofen18] Altenhofen C., Luu T.H., Grasser T., Dennstädt M., Mueller-Roemer J.S., Weber D. and Stork A.: Continuous property gradation for multi-material 3D-printed objects, Proceedings of the 29th Annual International Solid Freeform Fabrication Symposium – An Additive Manufacturing Conference. Austin, Tex.: University of Texas 2018, pp. 1675-1685

[Barrett74] Barrett, James P. "The coefficient of determination—some limitations." The American Statistician 28, no. 1 (1974): 19-20.

[Chakravarti67] Chakravarti, Laha, and Roy, (1967). Handbook of Methods of Applied Statistics, Volume I, John Wiley and Sons, pp. 392-394.

[Christou18] I.T. Christou et al. "A Parallel/Distributed Algorithmic Framework for Mining All Quantitative Association Rules", arXiv preprint arXiv:1804.06764, 2018

[Christou19] I.T. Christou, "Avoiding the Hay for the Needle in the Stack: Online Rule Pruning in Rare Events Detection", Proc. IEEE ISWCS 2019, Oulu, Finland, Aug. 27-29, 2019.

[Cireşan2011] D. C. Cireşan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Flexible, High Performance Convolutional Neural Networks for Image Classification," in Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two, 2011, pp. 1237–1242.

[D'Agostino86] D'Agostino, Ralph B. Goodness-of-fit-techniques. Vol. 68. CRC press, 1986.

[Deuse14] J. Deuse, M. Wiegand, O. Erohin, D. Lieber und R. Klinkenberg. "Big Data Analytics in Produktion und Instandhaltung". In: Instandhaltung im Wandel. Herausforderungen und Lösungen im Zeitalter von Industrie 4.0. Hrsg. von H. Biedermann. 2014, S. 33–48.

[Deuse17] J. Deuse, J. Schmitt, M. Stolpe, M. Wiegand und K. Morik. "Qualitätsprognosen zur Engpassentlastung in der Injektorfertigung unter Einsatz von Data Mining". In: Schriftenreihe der Wissenschaftlichen Gesellschaft für Arbeits- und Betriebsorganisation (WGAB) e.V. (2017).

[Gelman03] Gelman, Andrew, John B. Carlin, Hal S. Stern, and Donald B. Rubin. "Bayesian Data Analysis. Chapman & Hall/CRC." In :. 2003.

[Gockel2014] J. Gockel, J. Beuth, and K. Taminger, "Integrated control of solidification microstructure and melt pool dimensions in electron beam wire feed additive manufacturing of Ti-6Al-4V," Additive Manufacturing, 1-4, pp. 119–126, 2014, doi: 10.1016/j.addma.2014.09.004.

[Harper16] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. <i>ACM Trans. Interact. Intell. Syst.</i> 5, 4, Article 19 (January 2016), 19 pages. DOI:https://doi.org/10.1145/2827872

[He2015] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," Oct. 2015. [Online]. Available: http://arxiv.org/pdf/1512.03385v1

[Lu97] J. C. Lu, J. Park, Q. Yang, "Statistical inference of a time-to-failure distribution derived from linear degradation data", Technometrics, vol. 39, pp. 391-400, 1997.

[Luong17] M.-T. Luong et al. "Effective Approaches to Attention-based Neural Machine Translation", https://nlp.stanford.edu › pubs › emnlp15_attn, 2017.

[Motulsky87] Motulsky, Harvey J., and Lennart A. Ransnas. "Fitting curves to data using nonlinear regression: a practical and nonmathematical review." The FASEB journal 1, no. 5 (1987): 365-374.

[Müller 07] Müller, M. (2007). Dynamic time warping. Information retrieval for music and motion, 69-84.

[Myung03] Myung, In Jae. "Tutorial on maximum likelihood estimation." Journal of mathematical Psychology 47, no. 1 (2003): 90-100.

[Nagabandi2018] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning," in 2018 IEEE International Conference on Robotics and Automation (ICRA), 2018, pp. 7559–7566.

[Olakanmi2015] E. O. Olakanmi, R. F. Cochrane, and K. W. Dalgarno, "A review on selective laser sintering/melting (SLS/SLM) of aluminium alloy powders: Processing, microstructure, and properties," Progress in Materials Science, vol. 74, pp. 401–477, 2015, doi: 10.1016/j.pmatsci.2015.03.002.

[Peng09] Peng, Chien-Yu, and Sheng-Tsaing Tseng. "Mis-specification analysis of linear degradation models." IEEE Transactions on Reliability 58, no. 3 (2009): 444-455.

[Polydoros2017] A. Polydoros and L. Nalpantidis, "Survey of Model-Based Reinforcement Learning: Applications on Robotics," Journal of Intelligent & Robotic Systems, vol. 86, 153-, 2017, doi: 10.1007/s10846-017-0468-y.

[Sakamoto86] Sakamoto, Yosiyuki, Makio Ishiguro, and Genshiro Kitagawa. "Akaike information criterion statistics." Dordrecht, The Netherlands: D. Reidel 81 (1986).

[Schulte16] L. Schulte, J. Schmitt, F. Meierhofer, J. Deuse: Optimizing Inspection Process Severity by Machine Learning under Label Uncertainty. In: Isabel L. Nunes

(Hg.): Advances in Human Factors and Systems Interaction. Proceedings of the AHFE 2020 Virtual Conference on Human Factors and Systems Interaction. International Conference on Applied Human Factors and Ergonomics (AHFE). USA (Virtual Conference), (2020), 16. - 20.07.2020. Cham, Switzerland: Springer.

[Sherstinksi18] A. Sherstinksi, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network", arXiv:1808.03314 [cs.LG], 2018.

[Si11] Si, Xiao-Sheng, Wenbin Wang, Chang-Hua Hu, and Dong-Hua Zhou. "Remaining useful life estimation–a review on the statistical data driven approaches." European journal of operational research 213, no. 1, 1-14, 2011.

[Tapia2014] G. Tapia and A. Elwany, A Review on Process Monitoring and Control in Metal-Based Additive Manufacturing," Journal of Manufacturing Science and Engineering, vol. 136, no. 6, 2014, doi: 10.1115/1.4028540.

[Whip2019] B. Whip, L. Sheridan, and J. Gockel, "The effect of primary processing parameters on surface roughness in laser powder bed additive manufacturing," Int J Adv Manuf Technol, vol. 103, 9-12, pp. 4411–4422, 2019, doi: 10.1007/s00170-019-03716-z.

[Wiegand16] M. Wiegand, M. Stolpe, J. Deuse und K. Morik. "Prädiktive Prozessüberwachung auf Basis verteilt erfasster Sensordaten". In: at-Automatisierungstechnik 64.7 (2016), S. 521–533.

[Wöstmann17] R. Wöstmann, P. Strauß und J. Deuse. "Predictive Maintenance in der Produktion: Anwendungsfälle und Einführungsvoraussetzungen zur Erschließung ungenutzter Potentiale". In: Werkstatttechnik online 107.7/8 (2017), S. 524–529.

[Yasa2011] E. Yasa, J. Deckers, and J.-P. Kruth, "The investigation of the influence of laser re-melting on density, surface quality and microstructure of selective laser melting parts," Rapid Prototyping Journal, vol. 17, no. 5, pp. 312–327, 2011, doi: 10.1108/13552541111156450.

[Zhang1990], W., Itoh, K., Tanida, J., Ichioka, Y., 1990. Parallel distributed processing model with local space-invariant interconnections and its optical architecture. Appl Opt 29, p. 4790.

List of figures

List of tables

List of Abbreviations

Acronym	Abbreviation
AI	Artificial Intelligence
AM	Additive Manufacturing
ANN	Artificial Neural Networks
API	Application Programming Interface
CNN	Convolutional Neural Network
CAD	Computer Aided Design
CMMS	Computerized Maintenance Management System
CPD	Change Point Detection
CPPS	Cyber Physical Production Systems
CSC	Cyber System Connector
CTQ	Critical To Quality
DAG	Directed Acyclic Graph
DDA	Distributed Data Analytics
DL	Deep Learning
DPE	Distributor Processor Engine
DSS	Decision Support System
DTW	Dynamic Time Warping
EPE	Edge Processor Engine
ETL	Extract Transform Load
GPU	Graph Processing Unit
GUI	Graphical User Interface
HDFS	Hadoop Distributed File System
HDR	High Dynamic Range
HMMs	Hidden Markov Models
IoT	Internet of Things
JAR	Java ARchive
LPBF	Laser Powder Bed Fusion
LSTM	Long Short-Term Memory
MASS	Mueen's Algorithm for Similarity Search
MBRL	Model Based Reinforcement Learning
ML	Machine Learning
MLOPs	Machine Learning Operations
MQTT	Message Queue Telemetry Transport
PBF	Powder Bed Fusion
QARMA	Quantitative Association Rule Mining
QMM	Quality and Manufacturing Management
RA	Reference Architecture
RDBMS	Relational Database Management System
RDD	Resilient Distributed Dataset
RNN	Recurrent Neural Networks
RUL	Remaining Useful Life
SMT	Surface Mount Technology
SPI	Solder Paste Inspection
SVM	Support Vector Machines
UI	User Interface
ZDM	Zero Defect Manufacturing

Partners: