OXFORD UNIVERSITY PRESS LTD JOURNAL **00** (0000), 1–11
doi:10.1093/OUP Journal/XXX000

Microservices Architecture to Enable an Open Platform for Realising Zero Defects in Cyber-physical Manufacturing

Rui Pedro Lopes[a,b], Ahmed Ibrahim[a], José Barbosa[a,b] and Paulo Leitao [a,b]

[a] *Research Center in Digitalization and Intelligent Robotics (CeDRI), Instituto Politécnico de Bragança, Campus de Santa Apolónia, 5300-253 Bragança, Portugal Email: {rlopes, ahmed, jbarbosa, pleitao}@ipb.pt*

[b] *Laboratório Associado para a Sustentabilidade e Tecnologia em Regiões de Montanha (SusTEC), Instituto Politécnico de Bragança, Campus de Santa Apolónia, 5300-253 Bragança, Portugal*

Abstract: The market's demand for high-quality products necessitates innovative manufacturing approaches that emphasize flexibility, adaptability, and the reduction of defects. Traditional systems are currently evolving towards embracing I4.0 technologies, including data collection, processing, analytics, and digital twin, aiming for zero-defect manufacturing quality. This paper introduces an open platform, compliant with RAMI4.0 standards, designed to improve manufacturing quality. The platform integrates data using Asset Administration Shells with microservices adaptation for data ingestion and advanced analytics. Additionally, it incorporates Non-Destructive Inspection tools, demonstrating a seamless integration of measurement and quality assurance. This study details the architectural specification and validation of the openZDM platform, employing microservices to ensure flexibility, modularity, and scalability, aligning with the RAMI4.0 model. The platform was validated through deployment in an automotive assembly line, highlighting its effectiveness in integrating inspection scenarios and early defect detection tools. The architecture employs a choreographed approach to manage loosely coupled microservices, enabling efficient data lifecycle management from collection through analytics to visualization.

KEY WORDS Zero Defects Manufacturing, Cyber-physical Systems, RAMI4.0, Microservices

Received July 17, 2025

1 Introduction

In today's globalized market, product quality plays a pivotal role in the success of manufacturing industries to meet customer needs and ensure their satisfaction. However, achieving high-quality products in a rapidly changing landscape requires adaptability and continuous adjustments in production targets. The challenge lies in maintaining product quality, by reducing product defects during the production process while addressing the growing unpredictability of demand. Traditional manufacturing systems, though effective, need to evolve toward innovative production systems that enable extensive data collection and processing, incorporating advanced algorithms that generate knowledge and introduce cutting-edge Digital Twins (DTs)[2]. These emergent technologies aligned with Industry 4.0 (I4.0) enable the implementation of zero-defect manufacturing of technologies towards a vision where defects are eliminated, leading to better products and enhanced competitiveness. For this purpose, innovation and open platforms are required to enable the interoperability of a plethora of advanced data-driven tools aligned with industrial standards, particularly with RAMI4.0[4] and applicable to a multitude of manufacturing necessities. Central to this innovative paradigm is the integration of a microservices architecture, offering a flexible and scalable approach to managing complex systems. By decomposing applications into smaller, independently deployable services, microservices facilitate quick adaptation to new technologies and changing market demands, improving the manufacturing industry's capability to achieve zero-defect production goals.

As a part of the European project openZDM, an open platform, fully compliant with the RAMI4.0 standards and specifications, is being developed being agnostic to any particular industrial sector. This platform seamlessly integrates data sources using Asset Administration Shells (AAS), which are logical sets of available information and the administration interfaces of specific assets within an organizational unit. Each AAS is connected to a particular asset, discoverable and explorable within the I4.0 organizational ecosystem[19]. With dynamic service adaptation for data ingestion, it allows the inclusion of advanced data analytical tools by specifying a standard approach to design and deploy them, and it includes the DT in a native manner, enabling the introduction of complex scenario generation towards Zero Defect Manufacturing (ZDM). Furthermore, the openZDM platform

offers a set of Non-Destructive Inspection (NDI) tools, which are natively integrated into the openZDM platform, demonstrating the custom design of measurement and quality assurance in a seamless manner. NDI tools include sensing devices and methods to check the quality and properties of a material or component without causing any physical damage, thus allowing the detection of defects or issues while keeping the product intact and usable.

Having this in mind, this paper describes the specification of a microservices architecture for implementing ZDM strategies, ensuring modularity, flexibility, scalability and reconfigurability, and aligned with the RAMI4.0 model. This open and modular platform can seamlessly integrate heterogeneous data collection systems and advanced analytical tools, facilitating its design, development, and deployment. The proposed microservices architecture was validated through deployment in an automotive assembly line, integrating data from various inspection scenarios to enable real-time monitoring and early defect detection.

The rest of the paper is organized as follows. Section 2 overviews the existing work related to open and modular platforms to implement industrial Cyber-physical Systems (CPS). Section 3 describes the proposed microservices architecture as an open, modular, scalable and interoperable platform to deploy industrial CPS aligned with the RAMI4.0 model. Section 4 presents the experimental implementation of the proposed approach to an automotive assembly line and discusses the achieved results. Finally, Section 5 rounds up the paper with the conclusions and points out future work.

2 Related Work

The ZDM concern is prevalent and recurrent in both industry and scientific literature. The pursuit of ZDM integration relies on the digitalization of traditional industrial processes to influence every step of the manufacturing value chain, necessitating a specific architecture to design and implement various components in the production process. ZDM depend on the digitalization of physical processes, so that data is acquired and processed, providing valuable insights for decision making and quality control. These processes may be assisted by data analytics algorithms, complemented by DTs for testing and optimizing different scenarios.

Konstantinidis et al. [7] specifies a layered architecture for achieving zero defective products in diary production consisting of four layers: i) Physical layer, which represents the physical environment of dairy production, including dairy farming, dairy manufacturing, and dairy warehouse and logistics, ii) Vision-based layer, with sensors that are used to receive information about the environment of the dairy supply chain; iii) DT layer, models provide analytics and predict future states. They help to identify early disease symptoms, improve cows' nutrition, and improve milk production without impacting the manufacturing process by simulating future states of the processes. Moreover, DTs are used to prevent equipment breakdowns in manufacturing equipment, thereby reducing maintenance costs; iv) ZDM layer, the linking layer between the other layers, using data in order to inspect the product and product quality, and predict potential quality problems, to allow for repair strategy for detected defects and faults and prevention actions responsible for ensuring that no defective product is being made or failure is realized are required. This architecture is a conceptual framework for quality assurance in the dairy industry, without specifying technologies or implementation approaches. Similarly, Magnanini et al. [8], describe a similar layered approach, with the first layer (Multi-source multi-sensor network) to gather data from the shop-floor, second layer (Data management and synchronization) to collect, synchronize, monitor and store data gathered from the first layer, and third layer (System-level Engineering Platform) to optimize the joint quality and production logistics control shop-floor policies that integrates quality, production efficiency and economical aspects.

The track to Industry 5.0 (I5.0) introduces several evolutions in terms of intelligent and softwarized network architectures and services [20]. The focus on human-centered, resilient, and sustainable production presents several technological challenges, including: i) AI-native service-oriented architecture to provide heterogeneous personalized services; ii) high data-transportation and intelligent edge networks; and iii) support for knowledgeable and autonomous human-centric systems. The key term in the authors' findings is "service-oriented," indicating that most functionality is provided from the cloud in various forms. Examples include Navigation/Tracking-as-a-Service (NTaaS), Everything-as-a-Service (XaaS), Network Function-as-a-Service (NFaaS), AI-as-a-Service (AIaaS), and Digital Twin-as-a-Service (DTaaS). This approach allows for flexible and rapid provisioning of tools and analysis, which is critical for the I5.0 vision.

A fundamental component in ZDM is the process of inspecting, testing, or evaluating materials, components, or assemblies for discontinuities, defects, or differences in characteristics without destroying the part's serviceability through NDI [11]. Tasked with digitizing information from the shop floor, the retrieved data must be encapsulated and made available for processing, analysis, visualization, and decision support. The authors describe an architecture composed of a connector, the NDI business logic, and data storage. The connector handles communication with the platform, while the business logic acts as the bridge between the physical asset and its digital representation, using various data processing techniques to transform data into relevant

information. Additionally, the authors consider the RAMI4.0 reference architecture to frame the NDI within the AAS.

In fact, industrial machines involved in production aiming at ZDM are extensively monitored with digital instrumentation and form part of a CPS. In an industry perspective, such assets, together with an AAS, described in the RAMI4.0 documentation, form an Industry4.0 component. Sølvsberg et al. [16] focus on the design and development of a data structure in AAS formatted as a timeline, with small batches based on process or production sequence, common multiple of sampling rates, and early basic data processing, at asset level, and saving to a compressed format. This allows for an analysis-oriented structure communicated through an I4.0 AAS to enable generic multivariate analysis and allow dealing with unforeseen situations.

Martinez et al. [9] describe a case study inspired in a ZDM-CPPS (Zero Defect Manufacturing – Cyber-Physical Production System) hybrid framework, where a cycle between product and machine is followed, including product inspection, data analysis, repair systems, equipment inspection, machine health data analysis and preventive maintenance try to ensure zero defect products. The case study, encompassing steel frame assemblies manufacturing, uses camera to inspect the product, a PC for analysis, and a control panel for managing the actuators. The main objective is to analyze the relationship between product and machine health.

Data acquisition, storage and analysis are fundamental for ZDM. In addition, the visualization of results helps the users understand the information and helps in the decision making process. The diversity of data and analysis algorithms require a flexible architecture for both defining data-processing pipelines and custom visualization controls, around a dashboard. In Reiff et al. [14] work, a web-based platform was developed in Django (back-end) with a front-end in Angular to process data from several sources, enabling intuitive use of data analysis methods. The pipeline is assembled graphically, with the assistance of a custom application similar to the Node-RED concept for assembly blocks. Several blocks, called *Stage*s by the authors, are assembled in a *Pipeline*, making a *Task*, which represents the entirety of an analysis process. *Task*s are represented in a local database and executed on the same memory space, which can hinder scalability.

In the GO0D MAN project, Angione et al. [1] specify a distributed layered architecture that represent the physical Multi-Stage Production and Quality Control processes with a multi-agent system (MAS) (second layer), thus able to adapt to different data schemas and to scale as necessary. The third layer, ZDM Data and Knowledge Management, communicates with the MAS to retrieve the data as well as to control the actuators. Focusing on the vertical flow of responsibility (shop-floor – decision making), there is still lacking a specification for flexible and scalable data storage and processing components.

ZDM is a growing concern in I4.0, specially when the first steps are being visioned towards I5.0. Most of the related work described above, although focusing on ZDM, fall short in terms of technological challenges to deal with flexibility, scalability and data agnosticism. Moreover, the complexity associated with both the production process as well as with the requirements for data privacy, maintenance, instrumentation and others introduces a tangle of concepts and technologies that can easily become an unstructured mess. RAMI4.0 addresses these problems, providing a three-dimensional map showing how to approach the issue of I4.0 in a structured manner. RAMI4.0 is a service-oriented architecture, which combines all elements and information technology (IT) components in a layer and life cycle model.

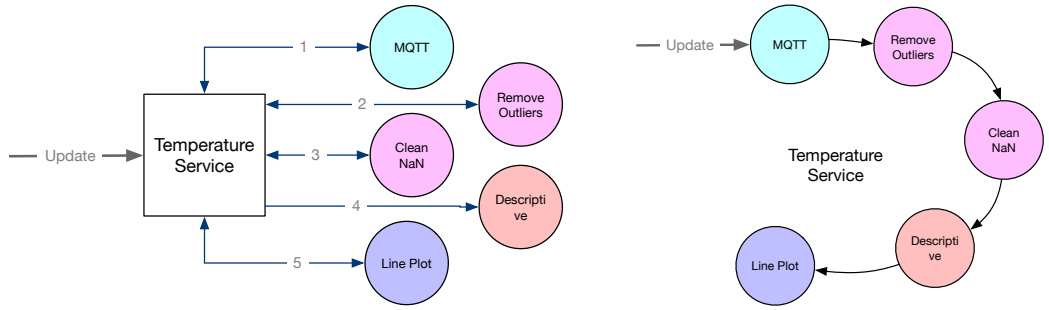
The literature highlights the importance of focusing on a service-based architecture, composed of loosely-coupled, scalable components that can be organized around business capabilities, able for decentralized data management, design to cope with failures and to evolve with time and complexity. This leads to a RAMI4.0 compatible microservices platform which allows to support the development of industrial CPS.

3 Microservices System Architecture

The approach to a microservices architecture is aligned with the necessities of ZDM to improve flexibility, modularity, and scalability within CPS. As illustrated in the GO0D MAN project, ZDM takes advantage of distributed and intelligent infrastructures to monitor, analyze, and react to deviations in real-time, following the "do things right the first time" philosophy to maximize revenue by both eliminating the cost of failure and by increasing customer satisfaction [1]. The microservices architecture alongside its distributed nature complements that philosophy with its rapid autonomy in both local and cloud deployment. It is composed of compact and independent services that communicate with each other, enabling for flexible applications that lead to the decomposition of monolithic systems into independent deployable services. This approach is aligned with the goals of I4.0, fostering enhanced interoperability, agility, and digital integration within manufacturing ecosystems [15, 5].

Microservices represent both an architectural and operational strategy in software development. When containerized, the deployment unit encapsulates the functionality and the necessary dependencies, facilitating the deployment across different environments and improving the system's resilience and manageability [3].

Considering that microservice architecture uses typically small, lightweight, and easy to implement modules and that they enable reusability, efficient use of resources, and ability to scale on demand, it is still necessary to get microservices to work together. For that, two main approaches are considered. In orchestration, a central coordinator manages the interactions between services, allowing for a structured approach (Figure 1a). Choreography, on the other hand, allows for a decentralized, event-driven interaction model where services independently react to events, showing flexibility and resilience (Figure 1b). Within the ZDM context, resilience specifically is highly advantageous as real-time manufacturing variations without a single point of failure is essential. The use of message queues and schema facilitates this as well as it ensures the services' ability to communicate with each other.



(a) Orchestrator (b) Choreography

Figure 1: Microservice coordination

Additionally, the architecture permits the custom, on-premises implementation of different business cases. By assembling different microservices and publishing data schemas along, each business case can be precisely tailored to meet operational needs [17].

3.1 Pipeline of Microservices

The proposed approach to develop an open, modular and scalable platform for realizing ZDM provides a scenario that can benefit from the use of a microservices architecture, assembled to represent the business case by choreographing a set of services (Figure 2).

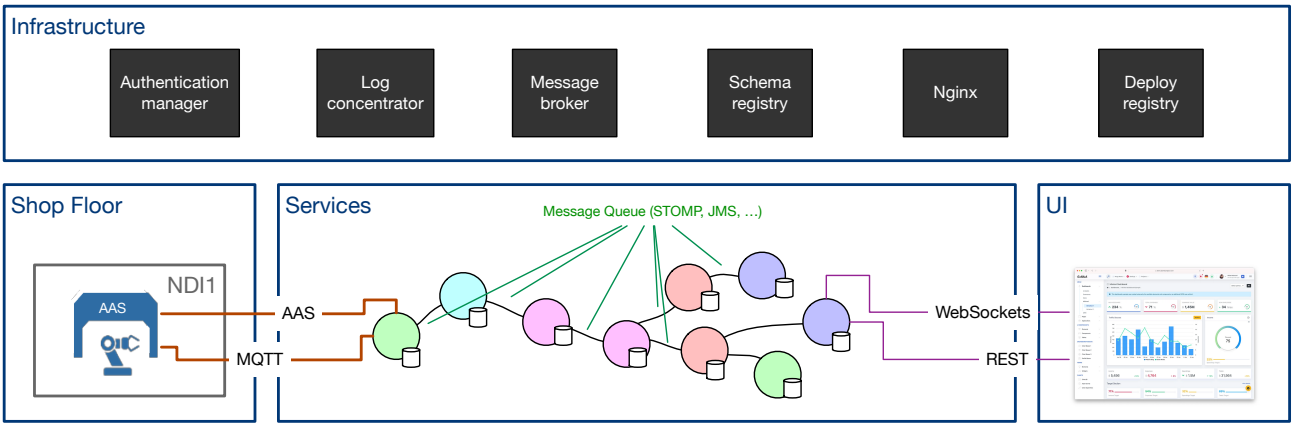


Figure 2: Microservices system architecture.

The architecture is composed of the input data from the shop floor, by AAS representation of assets, by Message Queuing Telemetry Transport (MQTT) or other means, within the RAMI4.0 specification. The entry point is a microservice that receives the data, produces a data schema and forwards the data, in a common format, to the next microservice. The data flows among the different services, which, in turn, read, change and publish data, until it is visualized in a dashboard or sent as a high-priority event (SMS, Telegram or other). The change towards microservices architecture is done within a specific pipeline, which is fundamental for managing the data life-cycle from collection through to analytics and visualization. This pipeline encapsulates a sequence of microservices, created to perform specific tasks in the data life-cycle [10]:

1. Data Collection: initial microservices are responsible for gathering data from the desired sources, ensuring robustness in data acquisition.

2. Data Processing: following collection, the data undergoes processing and, if necessary, they are transformed to different formats that fits the requirements of analytical tools.
3. Data Analytics: analytical tools interpret the processed data, applying analytics to extract patterns and trends.
4. Visualization and Reporting: the final stage in the pipeline involves visualizing these insights and generating reports, making the data easy to understand for end-users.

This sequential pipeline embodies the principles of scalability and modularity being designed to work with different workloads and integrate new technologies or updates. It ensures not only the services' independence and adaptability but also supports isolated changes, updates, and testing, minimizing disruption and maximizing fault tolerance. This design aligns with the agile methodology and continuous integration/deployment (CI/CD) practices, facilitating a dynamic and responsive microservices ecosystem.

3.2 Managing Loosely-Coupled Microservices

The implementation of an "end-to-end" use case involves multiple microservices, going over the boundary of individual services. Managing a set of decoupled microservices has some requirements in terms of communication, data representation and life cycle, requiring a set of specific services, such as message broker, schema registry, deployment registry, authentication controller.

The message broker facilitates the time and space independent communication between different services, and allows services to publish messages to topics without knowing the subscribers, decoupling the message's producer from the consumers. The schema registry maintains a repository of the data schemas to guarantee the dynamic adaptation to different information, structures or business case. This ensures that all the services adhere to a defined set of schemas that are structured according to the requirements which plays a role in maintaining data consistency and compatibility across the system. The deployment registry is essential for storing and distributing container images, as it encapsulates the microservices and their runtime environment. This allows for efficient management of different versions of the services, which contributes to the system's resilience as it allows the services to be quickly deployed or rolled back if needed.

Nginx is used for communicating with the user. It implements the API Gateway between the user interface, running in the users' browser, and the microservices, encapsulated in containers and, as such, directly inaccessible from the browser. The API Gateway supports, thus the microfrontend approach for a modular and flexible visualization. Since all the components are running in containers, it is very difficult and repetitive to check the logs of all the applications and containers. For that reason, a log concentrator is used, which may be embedded in the container orchestration (Docker Swarm or Kubernetes). Lastly, the authentication controller is responsible for managing access control, verifying the identity of the users, enforcing security policies, and preventing unauthorized access. This component is highly important in the context of I4.0, where integrity and confidentiality of data are paramount.

3.3 Deployment of Microservices

Deploying microservices (mostly in cloud environments) is an aspect that benefits from containerization technologies. Containers offer a simple yet efficient method for resource allocation, monitoring, and ensuring the independent deployment of microservices. The process includes a number of activities, including building container images, managing these images, and orchestrating containers to streamline deployment and operations [3]. The starting point of the deployment process is the creation of container images for each microservice. This is done by packaging the microservice with its dependencies into a container image, which is then stored in a repository for deployment.

Container orchestration tools, e.g. Kubernetes or Docker Swarm, are used to manage the deployment of containers in a cluster of machines. These tools automate the scaling, management, and networking of containers, making sure that microservices are deployed and are able to scale and that are resilient to failures by ensuring a minimum number of replicas [18, 12].

3.4 Frontend and API Gateway

As mentioned above, the business logic is implemented in a pipeline of microservices, where the output of one microservice is connected to the next one in a carefully implemented choreography. In addition, the containerization and deployment encapsulate the functionality in isolated containers, that communicate between them within a virtualized platform. Although this is straightforward for ensuring the connectivity and cooperation between them, it also raises some challenges to the development of user interfaces.

Micro-frontends bring the concept of microservices architecture to the frontend, allowing the break up of web applications into smaller deployable units. This approach supports parallel development efforts, making it simpler to manage and deploy updates [13].

The main issue with this approach is the reachability of each individual visualization component. For that purpose, API gateways are used, acting as a bridge between the frontend and the microservices backend. They offer a unified interface to aggregate backend services, manage routing, and address issues such as security and rate limiting. The decision to employ choreography or orchestration highly affects the setup and functionality of API gateways. This choice determines how frontend segments interact with backend services, through centralized orchestration for direct interactions or decentralized choreography for flexible communications [18]. In this work, the API gateway is provided with a reverse proxy based on nginx.

3.5 Alignment with RAMI4.0

The microservices architecture aligns with the RAMI4.0, Especially through the concept of the AAS, which acts as a DT that encapsulates the digital and physical dimensions of manufacturing components, consequently enabling interactions and data exchanges within I4.0 frameworks. By promoting flexibility and scalability, this architecture resonates with the RAMI4.0's structure, including Layers, Life Cycle & Value Stream, and Hierarchy Levels, to allow for interoperability among various systems and different components. The integration of microservices with AAS paves the way for smart manufacturing and digital integration, embodying the principles of I4.0 and bolstering a robust framework for implementing ZDM strategies in industrial CPS [6].

4 Experimental Implementation

This section details the deployment of the proposed system architecture in an industrial study, illustrating a microservices-based approach with choreography for business logic implementation align with RAMI4.0. The case study include a pipeline implementation of three business cases in the automotive assembly line, focused on the body area that comprises three inspection stations: the first measures a set of points (x,y and z coordinates each) in the car frame, the second measures the gap and flush of the fitted doors, and the last measures the gap and flush of the trunk and hood. Each station produces approximately 300 measurements (D points), each one with the specification (US and LS), rejection (UR and LR) and tolerance (UT and LT). In total, each system will output between 2000 and 3000 measurements for each car. The experimental implementation focuses, for simplicity, on the first inspection station, where the car frame is measured before attaching the doors, hood, and trunk. The microservice architecture is generic enough to be applied, without changes, to the rest of the inspection stations.

The pipeline implementation involves the data flow through three stages for each inspection station: data collection, data processing, and data utilization. In the first stage, data is collected from the inspection station and retrieved through an AAS interface. In the second stage, the data is cleaned and checked for correctness. Finally, in the third stage, three business cases are applied: storing the data in a datalake for long-term analysis, real-time monitoring of the measurements for immediate decision-making, and rule-based analysis for quality control and defect detection (Figure 3).

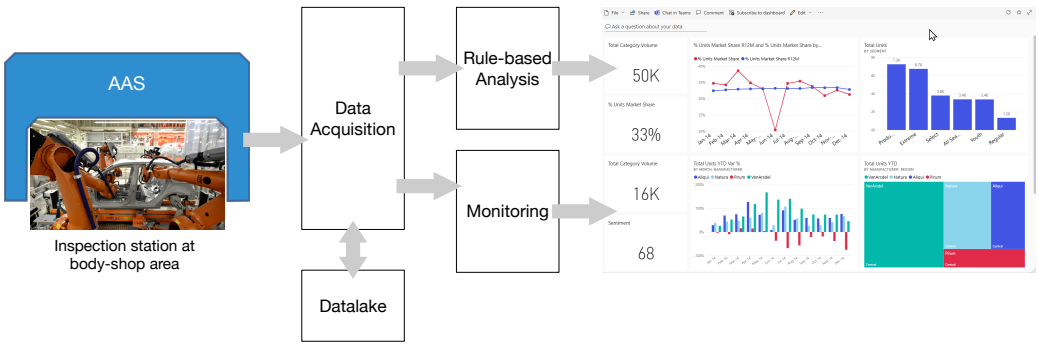


Figure 3: Overview of the experimental implementation

The three business cases capitalize on the integration of AAS to improve the manufacturing processes integrating storage of data in a Datalake, application of rule-based analysis for outlier detection and quality control and Real-time Data Visualization to assist in the immediate decision making.

4.1 Infrastructure

The instantiation of the proposed microservices architecture started with the definition of the services that contribute to manage the decoupled microservices, in terms of message broker, schema registry, deployment registry, and API gateway. The message broker was built around the NATS Messaging Service, due to its support for multiple programming languages, real time data streaming, resilient data storage and flexible data retrieval, providing a set of fast, reliable queues which microservices can subscribe and publish to (Figure 4). The NATS managed communication channels provide the connectivity between the modules, creating a pipeline that can exist in a local set of computers or in a cloud environment.

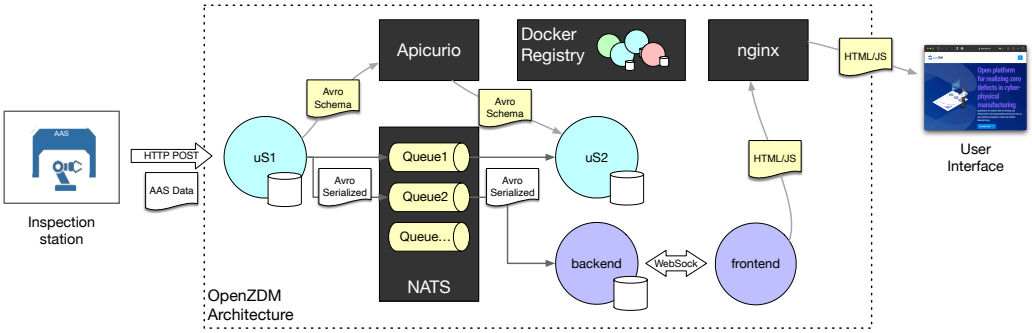


Figure 4: Overview of the microservices management services

The exchange of messages between microservices requires that they recognize the structure of data and adjust their operation accordingly. For that, it is necessary that each one of them publishes the produced data schema, so that it can be dynamically consumed by the destination. In the scope of this experimental implementation, the Apache Avro serialization was selected, together with the associated schema. The schemas are published in the Apicurio Registry, a high performance, runtime registry for API design and schemas.

The user interface is created as a three tier web application, visualized in a browser. Since both the backend and frontend are encapsulated in Docker, an nginx reverse proxy is configured as API gateway. Finally, the image management is performed in the docker registry, which are deployed to a Docker Swarm infrastructure.

After setting the infrastructure and defining the supporting services, three business logic scenarios were implemented, namely Datalake Integration, Rule-Based Analysis and Real-Time Data Visualization.

4.2 Datalake Integration

One important functionality of a data processing pipeline is the possibility to store, for a long period, a large amount of raw data. In a microservices architecture, each module is responsible for holding only the data they need, discarding what is not necessary. Nevertheless, for historical purposes or for advanced processing, it is convenient to hold data for an undefined amount of time. For this reason, a datalake integration pipeline was implemented, as illustrated in Figure 5.

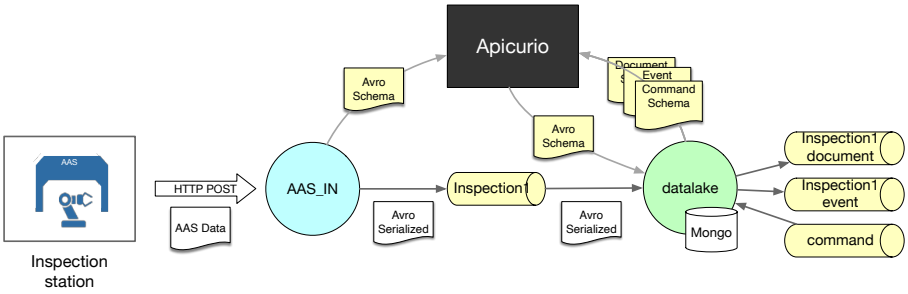


Figure 5: Datalake integration pipeline

The data is stored in MongoDB which provides simple yet efficient functionalities, enabling ongoing data snapshots, and data backup and restoration. The data stored is structured according to the Avro schema that encapsulates the data models defined by the AAS. Avro is language-agnostic, enabling data serialization across programming languages and frameworks.

The datalake service works as both data sink and data source. From the sink side, the NATS queues are subscribed to receive and store data. From the data source side, it publishes to two queues (*document* and *event*)

and subscribes to one (*command*), to be able to receive commands and respond to them. Whenever a new data item is stored, the datalake publishes an event in the *event* queue and forwards the data to the *document* queue. The datalake receives `delete`, `replay` and `search` commands, to remove data or to resend specific data units to both *event* and *document* queues.

It is important to highlight that, according to the microservices architecture and nature, each microservice has the responsibility to store the data it needs, meaning that they will not depend or need the datalake for operating. Nevertheless, data is safely stored in there, for future-proof.

4.3 Rule-Based Analysis

The purpose of the rule-based analysis within the context of zero-defect manufacturing is to apply a set of predefined rules to process and analyze data collected from the body shop stations (Figure 6). The objective is to identify deviations, anomalies, or defects in real-time or near-real-time, enabling the immediate execution of corrective actions or adjustments. Specifically, this analysis considers Nelson rules, a set of statistical rules used for detecting out-of-control signals in process control. The information processed by the rule-based analysis is stored locally in a SQL Lite database, allowing for independence and quick access. The backend validates the data against a predefined schema to show if rules were triggered. If so, a notification is sent to the frontend, showing an alert to the user indicating the rule and points triggered.

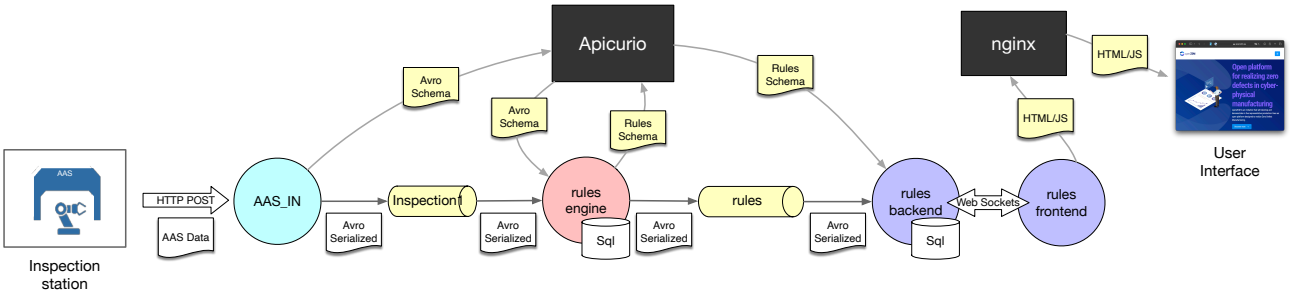


Figure 6: Rule-Based analysis pipeline

4.4 Real-time Data Visualization

Real-time monitoring within the ZDM framework is crucial for peaking into the manufacturing process, enabling the recognition of patterns and the detection of anomalies, deviations, and defects as they occur. This is important for maintaining high-quality standards, thereby contributing to the overall efficiency and effectiveness of the manufacturing operations. Just like the rule-based analysis service, the real-time monitoring service stores its data locally in an SQL database in the backend, since previously mentioned, it allows for quick access and independence. The frontend of the real-time monitoring service is built using React, a JavaScript library for building user interfaces. The React's component-based architecture allows for the development of a responsive and dynamic frontend that can update in real-time as new data becomes available. The communication between the frontend and the backend is facilitated through WebSocket connections, which allows for full-duplex communication between the client and the server, enabling the server to push updates to the client in real-time. This is useful for real-time monitoring applications where data needs to be updated continuously without requiring the client to make repeated requests.

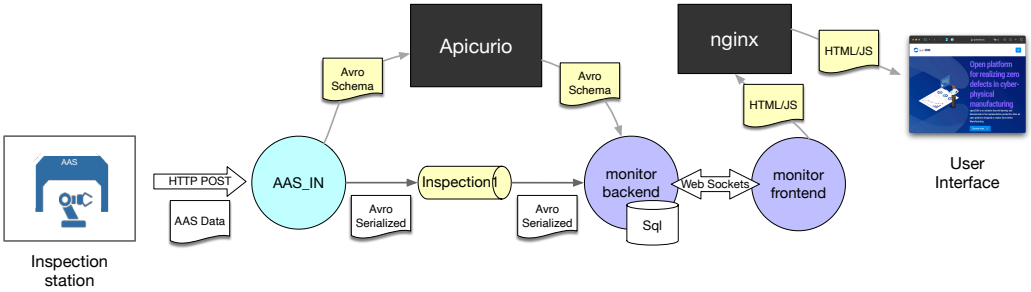


Figure 7: Real-time monitoring pipeline

5 Analysis of Results

The individual description of the previous business cases does not prevent from looking at all of them collectively (Figure 8). Properly choreographed microservices ensures that each acts and proceed with the functionality when the previous one finished its processing. The communication infrastructure, together with the other microservices management services ensures loose coupled interaction and flexible combination of functionality.

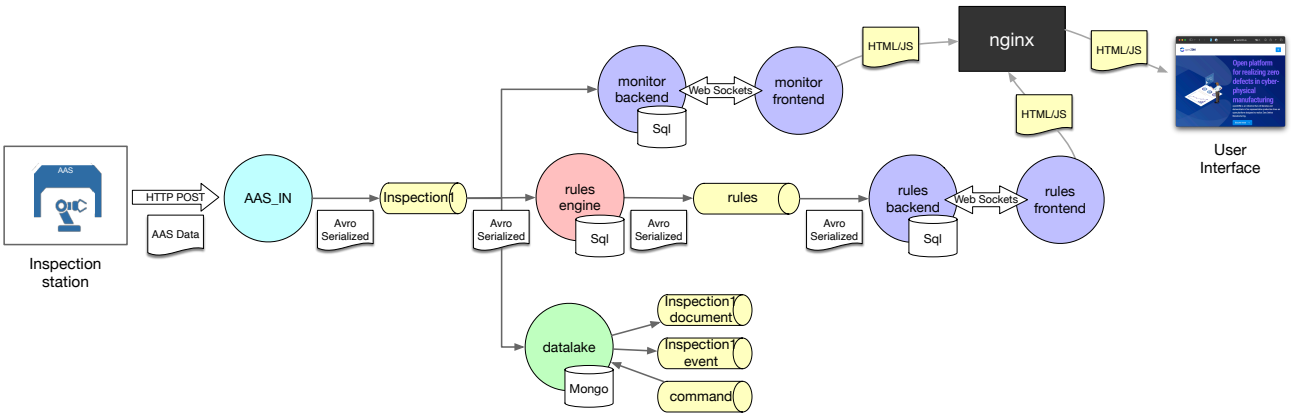


Figure 8: Datalake, rule-based analysis and real-time monitoring pipelines

The implementation of a microservices-based system architecture for realizing zero defects in cyber-physical manufacturing project demonstrates significant qualitative benefits in terms of flexibility, modularity, robustness, and scalability. This architecture allows the custom, on-premises implementation of multiple different business cases, allowing the services to be tailored to their specific operational needs. By taking advantage of microservices, the system can adapt to changes, ensuring a dynamic and responsive manufacturing environment.

In terms of flexibility and modularity, the system's design allows for the smooth integration and disintegration of services depending on operational needs, enabling a reconfigurable and modular approach to application development. This is beneficial in a manufacturing context where operational requirements can be very different from one situation to another. The ability to easily assemble and reconfigure services according to specific business cases without disrupting the system functionality shows the architecture's agility.

The robust nature of the microservices architecture ensures that the system can hold out against failures within individual services without compromising the integrity of the entire application, which is critical in maintaining the production processes. Moreover, the architecture's scalable design means it can accommodate different loads, from gradual increases in data processing demands to the integration of new manufacturing lines, with minimal adjustments. The business cases' implementation shows the microservices architecture's scalability and independence. Each microservice is created for specific tasks such as data inspection and transformation, analytics, and visualization, they operate independently and collaboratively. This modularity allows for scaling and improves the system's agility, adapting to evolving requirements. The architecture's decoupled nature simplifies maintenance and ensures system integrity and operational efficiency, consequently setting a new standard for operational excellence in cyber-physical manufacturing environments.

Moreover, in a demonstration of the system's capability to manage high volumes of data efficiently, a detailed operational benchmark was achieved during the deployment, spanning across different amounts of data load. The system was tested with 10, 100, 1000, and 10000 messages, revealing processing times of 7, 12, 64, and 635 seconds respectively. The tests underline the system's proficiency in scaling up to handle larger datasets efficiently, confirming a consistent and scalable performance. It is important to note that this metric is not solely indicative of raw throughput; it encapsulates a comprehensive data handling process. Initially, data is extracted from CSV files and structured following the AAS format. This structured data is then dispatched via a POST request to the NDI entry point. Upon reaching the NDI, the data undergoes a transformation into Avro format, an operation crucial for achieving uniform data representation. The data in Avro format is then sent to a message queue that carries the name of the corresponding schema. This process, from the data extraction to final queuing, underscores not only the system's ability to handle complex data workflows but also its alignment with the agile and dynamic manufacturing environment envisioned by I4.0 and the RAMI4.0 standards.

6 Conclusions and future works

In light of the imperatives imposed by the global market landscape, this paper put forward a refined architectural strategy that is anchored in the microservices paradigm to address the challenges of maintaining

and improving product quality in the face of dynamic market demands. This approach aims to set a benchmark in achieving ZDM goals across diversified industrial domains while aligning with I4.0's technologies, including RAMI4.0 standards. The framework introduced advocates for a structured integration of scalable and adaptable microservices that support data analytics, digital twins, and non-destructive inspection modalities. The goal of this integration is to simplify taking a proactive approach to defect reduction, which will improve product quality and provide businesses with a competitive advantage.

This architectural proposition was validated within the scope of the openZDM project, focusing on a pragmatic deployment in an automotive assembly line. This deployment underscored the architecture's robustness in coordinating data assimilation through AAS, dynamic adaptation of services for data ingestion, and the deployment of data analytical tools, thereby underlining its transformative potential in manufacturing processes. The designed system architecture, by its inherent flexibility, modularity, and scalability, sets forth a foundational framework for the development of open, modular, and interoperable platforms poised for the deployment of CPS that align with the RAMI4.0 model. The deployment outcomes demonstrated the architecture's resilience, adaptability, and capability to fulfill distinct business imperatives, thus pointing out a new era of operational excellence.

Future work will be devoted to integrating a more advanced configurator engine to support the easy and fast re-usability and re-configuration of the microservices workflow, enhancing the system adaptation and evolution to face condition changes (e.g., application to new domains, and adding or removing components), as well as to enrich the platform with cyber-security mechanisms to ensure data privacy and security in such distributed system.

Acknowledgements

This work was partially supported by the HORIZON-CL4-2021-TWIN-TRANSITION-01 openZDM project, under Grant Agreement No. 101058673. The authors are grateful to the Foundation for Science and Technology (FCT, Portugal) for financial support through national funds FCT/MCTES (PIDDAC) to CeDRI (UIDB/05757/2020 and UIDP/05757/2020) and SusTEC (LA/P/0007/2021).

References

[1] G. Angione, C. Cristalli, J. Barbosa, and P. Leitao. Integration challenges for the deployment of a multi-stage zero-defect manufacturing architecture. In *IEEE International Conference on Industrial Informatics (INDIN)*, pages 1615–1620, Helsinki-Espoo, Finland, 2019. ISSN: 1935-4576.

[2] M. Attaran and B.G. Celik. Digital twin: Benefits, use cases, challenges, and opportunities. 6.

[3] J. Cao. Design on deployment of microservices on container-based cloud platform. In *Journal of Physics: Conference Series*, volume 1624, 2020. ISSN: 1742-6588 Issue: 6.

[4] DIN Deutsches Institut für Normung e. V. DIN SPEC 91345: Referenzarchitekturmodell Industrie 4.0 (RAMI4.0). `https://www.din.de/de/forschung-und-innovation/themen/industrie4-0/din-veroeffentlicht-din-spec-zu-rami4-0-158570`, 2016. Accessed: 11 March 2024.

[5] Nicola Dragoni, Saverio Giallorenzo, Alberto Lluch Lafuente, Manuel Mazzara, Fabrizio Montesi, Ruslan Mustafin, and Larisa Safina. Microservices: Yesterday, Today, and Tomorrow. In Manuel Mazzara and Bertrand Meyer, editors, *Present and Ulterior Software Engineering*, pages 195–216. Springer International Publishing, Cham, 2017.

[6] Josef Frysak, Claudia Kaar, and Christian Stary. Benefits and pitfalls applying RAMI4.0. In *IEEE Industrial Cyber-Physical Systems (ICPS)*, pages 32–37, St. Petersburg, May 2018.

[7] F.K. Konstantinidis, V. Balaska, S. Symeonidis, F. Psarommatis, A. Psomoulis, G. Giakos, S.G. Mouroutsos, and A. Gasteratos. Achieving Zero Defected Products in Diary 4.0 using Digital Twin and Machine Vision. In *ACM International Conference Proceeding Series*, pages 528–534, 2023.

[8] M.C. Magnanini, M. Colledani, and D. Caputo. Reference architecture for the industrial implementation of zero-defect manufacturing strategies. In *Procedia CIRP*, volume 93, pages 646–651, 2020. ISSN: 2212-8271.

[9] P. Martinez, M. Al-Hussein, and R. Ahmad. A cyber-physical system approach to zero-defect manufacturing in light-gauge steel frame assemblies. In *Procedia Computer Science*, volume 200, pages 924–933, 2022. ISSN: 1877-0509.

[10] Genc Mazlami, Jurgen Cito, and Philipp Leitner. Extraction of Microservices from Monolithic Software Architectures. In *IEEE International Conference on Web Services (ICWS)*, pages 524–531, Honolulu, HI, USA, June 2017.

[11] V. Medici, M. Martarelli, N. Paone, G. Pandarese, W. Van De Kamp, B. Verhoef, K. Sipsas, R. Broechler, L.B. Besada, K. Alexopoulos, and N. Nikolakis. Integration of Non-Destructive Inspection (NDI) systems for Zero-Defect Manufacturing in the Industry 4.0 era. In *IEEE International Workshop on Metrology for Industry 4.0 and IoT*, pages 439–444, 2023.

[12] Yipei Niu, Fangming Liu, and Zongpeng Li. Load Balancing Across Microservices. In *IEEE Conference on Computer Communications (INFOCOM)*, pages 198–206, Honolulu, HI, April 2018.

[13] A. Pavlenko, N. Askarbekuly, S. Megha, and M. Mazzara. Micro-frontends: Application of microservices to web front-ends. *Journal of Internet Services and Information Security*, 10(2):49–66, 2020.

[14] C. Reiff, S. Oechsle, F. Eger, and A. Verl. Web-based Platform for Data Analysis and Monitoring. In *Procedia CIRP*, volume 86, pages 31–36, 2020. ISSN: 2212-8271.

[15] A. Rojko. Industry 4.0 concept: Background and overview. *International Journal of Interactive Mobile Technologies*, 11(5):77–90, 2017.

[16] E. Solvsberg, C.D. Oien, S. Dransfeld, R.J. Eleftheriadis, and O. Myklebust. Analysis-oriented structure for runtime data in Industry 4.0 asset administration shells. In *Procedia Manufacturing*, volume 51, pages 1106–1110, 2020. ISSN: 2351-9789.

[17] Davide Taibi, Valentina Lenarduzzi, and Claus Pahl. Processes, Motivations, and Issues for Migrating to Microservices Architectures: An Empirical Investigation. *IEEE Cloud Computing*, 4(5):22–32, September 2017.

[18] Mario Villamizar, Oscar Garces, Harold Castro, Mauricio Verano, Lorena Salamanca, Rubby Casallas, and Santiago Gil. Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud. In *2015 10th Computing Colombian Conference (10CCC)*, pages 583–590, Bogota, Colombia, September 2015. IEEE.

[19] C. Wagner, J. Grothoff, U. Epple, R. Drath, S. Malakuti, S. Grüner, M. Hoffmeister, and P. Zimermann. The role of the industry 4.0 asset administration shell and the digital twin during the life cycle of a plant. pages 1–8. ISSN: 1946-0740.

[20] S. Zeb, A. Mahmood, S.A. Khowaja, K. Dev, S.A. Hassan, M. Gidlund, and P. Bellavista. Towards defining industry 5.0 vision with intelligent and softwarized wireless network architectures and services: A survey. *Journal of Network and Computer Applications*, 223, 2024.