Available online at www.sciencedirect.com

ScienceDirect

Procedia Manufacturing 51 (2020) 1091–1097

Procedia MANUFACTURING

www.elsevier.com/locate/procedia

30th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM2021)
15-18 June 2021, Athens, Greece.

A microservice architecture for predictive analytics in manufacturing

N. Nikolakis[a], A. Marguglio[b], G. Veneziano[b], P. Greco[b], S. Panicucci[c], T. Cerquitelli[d], E. Macii[e], S. Andolina[f], K. Alexopoulos[a]*

[a]*Laboratory for Manufacturing Systems and Automation, Department of Mechanical Engineering and Aeronautics, University of Patras, Patras, Greece*
[b]*Engineering Ingegneria Informatica S.p.A., Palermo, Italy*
[c]*COMAU S.p.A., Turin, Italy*
[d]*Department of Control and Computer Engineering, Politecnico di Torino, Turin, Italy*
[e]*Interuniversity Department of Regional and Urban Studies and Planning, Politecnico di Torino, Turin, Italy*
[f]*SynArea Consultants S.r.l., Turin, Italy*

* Corresponding author. Tel.: 30-2610-910160; fax: +30-2610-997314. *E-mail address:* alexokos@lms.mech.upatras.gr

Abstract

This paper discusses on the design, development and deployment of a flexible and modular platform supporting smart predictive maintenance operations, enabled by microservices architecture and virtualization technologies. Virtualization allows the platform to be deployed in a multi-tenant environment, while facilitating resource isolation and independency from specific technologies or services. Moreover, the proposed platform supports scalable data storage supporting an effective and efficient management of large volume of Industry 4.0 data. Methodologies of data-driven predictive maintenance are provided to the user as-a-service, facilitating offline training and online execution of pre-trained analytics models, while the connection of the raw data to contextual information support their understanding and interpretation, while guaranteeing interoperability across heterogeneous systems. A use case related to the predictive maintenance operations of a robotic manipulator is examined to demonstrate the effectiveness and the efficiency of the proposed platform.

Keywords: Machine learning; Service-oriented platform; Plug-and-play approach; Microservice architecture; Robotics industry.

1. Introduction

Several research efforts have been conducted concerning the design and development of distributed architectures to effectively support Industry 4.0 needs with respect to recent advances in the field of information and communication technology (ICT). Existing solutions include among others: (*i*) the integration of the existing production system with legacy systems and advanced Internet of Things (IoT) technologies as discussed in [1]. To this aim, the key ICT technologies are based on virtualization and a cloud-based service architecture. (*ii*) A distributed system architecture to enable predictive maintenance [2]. (*iii*) Fog Computing, a new architecture enabling on-demand computation to offload the computation in the cloud architecture, reducing unnecessary network overhead, by properly selecting the most effective edge devices as computation delegates[3][4].

This paper discusses on a lightweight, resilient, and scalable architecture based on microservices for enabling predictive analytics at the edge level of a cloud system. Considering the security aspect of the proposed architecture a multi-level authentication mechanism and its implementation is discussed in section 4. As a case study, the proposed architecture implementation coupled with the one required for testing applications has been deployed and connected to a robotic industry testbed.

10.1016/j.promfg.2020.10.153

2. Literature review

Current production systems are in their majority based on monolithic software solutions. This causes a lack of flexibility adaptability scalability and makes their maintenance complex. Service Oriented Architectures (SOA) were introduced to overcome these drawbacks. However, the increased complexity of modern manufacturing systems requires increased flexibility and further distribution of underlying functionalities. In fact, with the implementation of cyber physical production systems isolated services have resulted on utilizing versatile hardware and software components [5]. Consequently, flexibility as a result of increased automation levels are challenging. In this context micro-services are gaining momentum. In particular, a micro service can be defined as an independent part of a software interacting with via lightweight mechanisms, as mentioned in [6], where a classification framework for evaluating architectures is provided. A process consisting of four discrete steps is presented in [7] to decompose monolithic applications to microservices. Microservices provide mainly modelling and design principles to implement lightweight, scalable, and most important distributed applications. This kind of granularity facilitates the modelling, development as well as maintenance of applications. Nevertheless, the operation of microservices require more runtime environment environments to be distributed increasing their computational and network resources consumption [8].

IoT devices generate large volumes of data which are usually processed in a cloud system to enable on-demand services as well as scalability to future needs. One of the key challenges is the allocation of resources as well as the security aspect. The first give birth in recent years to a decentralized computing paradigm, the fog computing [9]. A review of the recent fog computing trends and applications is provided in [10], where among other challenges the application service and resources management as well as communication between layers along with security and privacy are highlighted.

A design methodology to embed analytics capabilities in modular microservices is presented in [11]. In this context, a monitoring system enabled by fog network, functionally and geographically decentralized is discussed in [12]. The proposed system is used for data monitoring in an industrial environment enabling predictive maintenance of a flexible assembly line. Moreover, IoT features have been used to enable real time condition monitoring and fault detection in the UK water industry [13]. The edge device collects and transmits data from the edge while analysis is performed at a network node, concluding that the proposed system potentially provides a low cost and flexible industrial solution.

To improve the scalability and flexibility of application development microservices are coupled with Docker containers [14]. According to [15] container virtualization is a lightweight solution to guarantee quality of service in IoT devices. A resource allocation algorithm is discussed in [16] to minimize the deployment cost. Moreover, a two-level orchestration approach based on microservices and containerization technologies is proposed in [17] for adaptive planning and control towards enabling reconfigurable cyber physical production systems. Moreover, Kiss et al. in [18] propose an automated orchestration for cloud applications using microservices. In addition, the time aspect of microservices is investigate in [19] with regards to the development and deployment of time critical microservices.

3. Platform architecture

The proposed architecture is comprised of several logical components, which collectively provide its predictive maintenance functionality. These logical components reside on a cloud host, but the functionality extends to the edge gateways and sensors on the factory floor. Additionally, the platform is intended to be flexible enough to be hosted on either an on-premises environment, a cloud service provider (CSP) environment, or a hybrid of the two.

In order to address these requirements, the proposed system has been designed on a micro-services architecture pattern [20], where each logical component is a service. Each service communicates and collaborates with other services to perform its specific function, and extends to other components, such as the edge gateways. The described platform has been designed with the purpose of being flexible in terms of deployment and use of technologies. As such, virtualisation techniques have been adopted to wrap microservices hosted in lightweight servers. The proposed architecture adopts a certain set of technologies, briefly mentioned below, without excluding their replacement with newer or different ones, depending on changing needs and requirements. Furthermore, other functionalities are included such as a) adaptive maintenance scheduling, driven from the prediction results; b) predictive analytics; c) visualisations tools.

A high-level architecture view of the proposed platform architecture is presented in Figure 1.

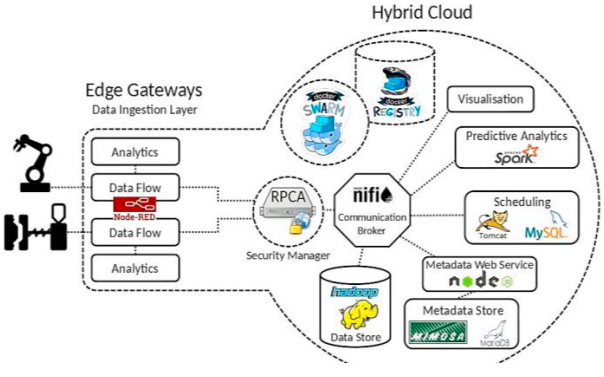

Figure 1: Architectural view

The data ingestion layer, composed of both hardware and software assets, collects data from several plant-level sources.

In particular, the layer has been designed to enable the gathering of sensor data even from legacy factory equipment that otherwise could not have provided data to any kind of cloud facility.

Before being sent to the cloud, where they will be stored into the platform repository, sensor data are pre-processed in order to transmit only the required data for the predictive analytics component. The data ingestion layer supports transmission of batch data collection as well as stream data in real-time, depending on the sources available at the deployment site.

Next, incoming data flows go through a security management layer involving authentication, authorization, encryption, and data lineage features. Moreover, the security layer supports a mutual authentication mechanism while facilitating role-based access control policies. The security manager is detailed in subsection 3.1.

After the security manager, the secured data messages go through the communication broker, i.e. NiFi. The communication broker coordinates the retention of the sensor data in the Data Store and of their metadata in the Metadata Store. Interaction with the Metadata Repository is achieved through the Metadata Web Service based on a REpresentational State Transfer (REST) Application Interface (API).

The communication capabilities of the proposed platform have been designed in accordance with the emerging trend of developing standard-based maintenance solutions. The proposed platform adopts the MIMOSA schema [21], which defines a set of specifications and hierarchy that complies with existing maintenance and interoperability standards. The MIMOSA schema enables physical asset lifecycle management for manufacturing, providing a hierarchical context for sensor-related data and analytics.

At the top of the MIMOSA hierarchy is the “Enterprise” which represents an organization or company. Under the **“**Enterprise” there are one or more **“**Sites”, which represent factories or even mobile manufacturing locations. “Sites” contain “Segments”, which are parts of a manufacturing process, such as a production line made up of “Assets”. An “Asset” is typically a piece of machinery or equipment that would appear in a bill of materials and have a manufacturers serial number. Finally, “Segments” are collections of “Assets” that have been put together to implement a manufacturing process. The proposed platform supports predictions made against both “Segments” and “Assets”.

A JSON-LD message format has been implemented as a data transfer model creating linkages to the MIMOSA data model. JSON-LD enables context aware metadata. The ingest service extracts the smart data and/or raw sensor data from the incoming message. Both types of data are converted into sets of comma delimited strings and stored as JSON files in the data repository. Due to the size of the sensor data, it is stored in individual files, based on the collection window metadata. “Smart” data are obtained by processing raw data through some form of aggregation function. They may only consist of a single string, and thus is typically appended to an existing file, whose collection window is defined in the metadata repository. For example, the standard deviation of a set of sensor data, or a maintenance prediction performed on the edge gateway are considered as smart data in the context of this work. The same JSON-LD message also contains extracted metadata which contain contextual information for the sensor data, such as the manufacturing operation being performed by a robot over the data collection window. The metadata allow the analytics engine to retrieve data based on the specific operation or configuration of the plant. The data file reference is included in the metadata, so that the platform’s services, or external authorized systems, can retrieve the required data. Thus, one of the functions of the metadata repository is as a smart index into the Data Store, i.e. HDFS.

The metadata configuration definitions have been populated into the metadata repository. While raw data, including sensor measurements, alarms, assets, etc. are stored in the platform repository, the elements required to support the collection and management of those data are extracted and stored in the metadata database. Additional metadata have been used for provisioning, monitoring, and measuring of quality of service and resilience of the cloud through responsive dashboards.

The platform’s repositories or storage layer are responsible for the persistence of raw data collected by the ingestion process and data which is the result of the distributed processes and analysis, including the predictions. Moreover, the publication layer, consisting of the visualization services, provides the results of the predictive maintenance algorithms processing data collected from the field by the batch processes described above.

Security manager, communication broker, data, and metadata stores, the components connected to the platform, as well the services needed to interact with them, are orchestrated using Docker Swarm. A scheduler addresses the orchestration of jobs through a unified workflow of different actions. Considering a platform consisting of more than one clusters with different hosts and services, the cluster manager facilitates their management while the resource manager allocates the jobs to available nodes, considering their hardware resources.

3.1. RPCA: Reverse Proxy Certification Authority

Regarding security, the proposed cloud platform authenticates every a) hardware device, i.e. plant-level equipment capable of sending data to the cloud, b) user or c) Docker host, which is a computer machine on which a Docker daemon runs, interacting with it. Moreover, their access to the platform resources is regulated. This is achieved through the security manager, named Reverse Proxy Certification Authority (RPCA).It is composed of two modules: CA (Certification Authority) and RP (Reverse Proxy).

The RPCA component is implemented on top of OpenSSL libraries. In particular, the CA module provides to each of the aforementioned entities a Transport Layer Security (TLS) certificate. The RP module determines which of the incoming

requests can be accepted. This component, acting as main cloud entry point, guarantees that the requests made by uncertified entities to access protected resources are refused. Four types of certificate are generated and used to implement a two-way authentication: *Hardware_Certificate*, *Browser_Certificate*, *Host_Certificate* (Client role) and *Service_Certificate* (Server role).

More specifically, when a hardware device provides its *Hardware_Certificate* to the Cloud endpoint (Reverse Proxy), the reached service (e.g. NiFi) answers back with its *Service_Certificate so that the device can verify the service identity*. *Hardware_Certificates* can be generated and downloaded through a web interface provided by the RPCA. Since the channel between the user's browser and RPCA web server has to be secured, *Browser_Certificates* are employed.

An operator - in charge of configuring the gateway at the Edge - who wants to obtain a *Hardware_Certificate*, installs on his/her browser a *Browser_Certificate* generated by the Cloud Administrator and received via a trusted channel.

Host_Certificates are used in the Cloud to secure the channels between each Docker Host and the platform's local Docker Registry containing the required Docker images so they can be safely downloaded.

Furthermore, the RPCA implements a multi-level authorization mechanism through a set of rules, cross-checking the identity information contained inside the certificates with the requested Uniform Resource Identifiers (URIs). Therefore, the authorization logic does not have to be implemented in each service connected to the platform. The RPCA logic diagram is presented below in Figure 2.

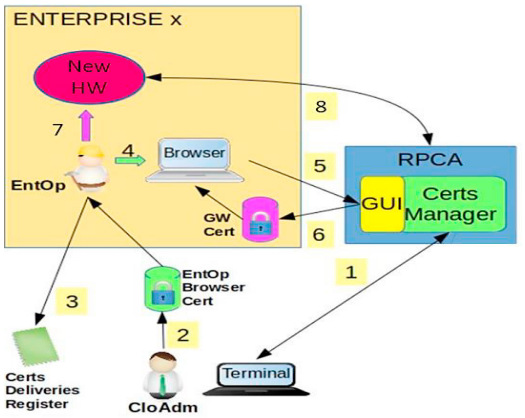

Figure 2: RPCA logic diagram, where the numbers indicate 1) the cloud admin accessing the RPCA through a shell, 2) cloud admin generating an certificate and delivering to the entity operator, 3) update the deliveries register keeping track of the deliveries, 4) entity operator installs the certificate in the browser of the device, 5) Entity operator accesses the RPCA certificate manager web application to submit the information required to generate a device specific certificate, 6) the certificate is returned to the device as an encrypted package containing the certificate authority (CA) certificate, 7-8) authentication procedure on runtime

3.2. Services

In this work and mostly for testing purposes two services are integrated into the platform to test its main functionalities. Specifically, the first service, the predictive analytics, is responsible to estimate the remaining useful life (RUL) of the machine/robot under analysis. The derived methodology might support proactive strategies to make a shift in traditional maintenance approaches to more effective optimizing strategies. The service, as detailed in [22], relies on data-driven methodologies to transform raw data collected from the industrial field into feature engineering to correctly estimate the machine degradation over time mainly affected by the operational response of the machine under daily usage. The Predictive Analytics module implements the main processing layer facilitating both stream and batch data processing. Stream data are processed, aggregated, and analysed event by event so that real-time results are available. The batch service on the other hand deals with cleaning tasks as well as analysis through custom algorithms for predictive maintenance objective. The batch jobs are scheduled and executed evenly across the cluster every day.

The visualisation service allows a strong interaction with the platform, through a modular and intuitive interface, designed to be exploited by different users, including maintenance engineers and operators as well as system administrators. Through the interface an operator can request/perform maintenance activities, filtered by his role and experience, supported, and enriched by available 3D models integrated in VR/AR applications, which provide step-by-step interactive guides. The scope of this service is to evolve the typical text-based maintenance manual into an interactive 3D maintenance visualization, to better guide the operator through complex procedures.

4. Platform Implementation

The SERENA cloud platform is built on a lightweight micro-services architecture, that allows the core cloud services, its applications as well as the edge components to be managed as a single domain.

The principal infrastructure technology to implement SERENA's services is Docker containers. Docker provides an open source implementation of a process containerization system, which encapsulates the service, and abstracts it from the underlying infrastructure that hosts the containers. Unlike virtual machines, which are a mechanism for logically dividing physical machines, Docker containers are principally a mechanism to wrap an application, and the resources required to run it, into a simple executable unit. Therefore, Docker containers are isolated from the underlying host infrastructure, and can easily be distributed between the available hosts to improve the flexibility and agility of the system. Docker containers share many similar features to virtual machines, but

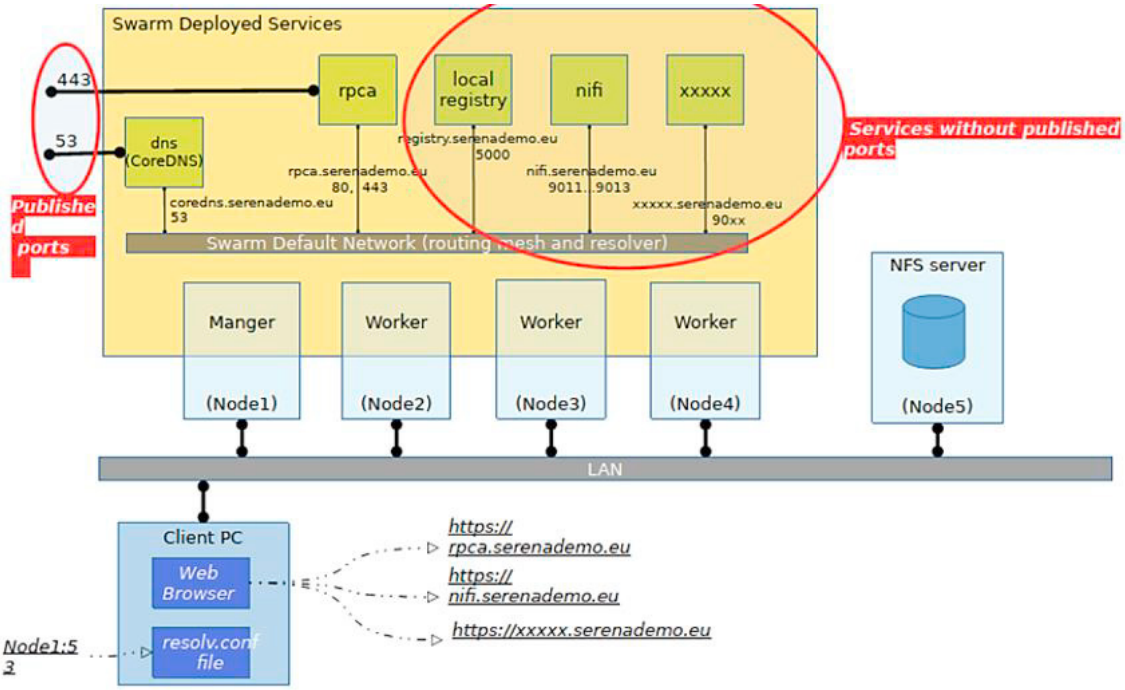

Figure 3: Proposed platform implementation

they perform distinctly different function within the infrastructure system. In fact, virtual machines and containers can be complimentary technologies, and used in conjunction with each other, the containers being layered on top of the virtual machines. One of the goals of the SERENA project, is for the SERENA system to have the flexibility to run on a wide variety of host infrastructures, from physical servers or virtual machines, to on-premises clouds and CSP (Cloud Service Provider) hosted environments, Docker containers provide the mechanism to achieve this goal.

A first implementation of the reference architecture is illustrated in Figure 3.

The platform implementation adopts the Cloudera distribution of the Hadoop ecosystem. Regarding the deployment Docker swarm has been selected since it provides native load balancing, service continuity and ability to easily scale services, easing at the same time the deploying activity. The raw and smart sensor data repository has been implemented in HDFS, while for the ingestion layer Apache NiFi has been used.

The main services are implemented using the following technologies:

- The cloud management and orchestration services, based on Docker and Docker Swarm
- The central communications broker implemented in NiFi
- The data, metadata, and document repositories rely upon HDFS
- The predictive analytics services, impended using
- Apache Spark
- Stream and batch processing using Spark libraries

All the services communicate using an overlay network, which allows each service to transparently communicate with other services, without having to know the other services location.

The ingest service exposes a REST API endpoint over an SSL connection, which uses the SERENA security service to mutually authenticate the entities and validate that the gateways are authorized to send data to the service. All the tasks in the service share a common service port number, and incoming traffic is load balanced between the available tasks in the server. The actual process flow is imbedded within the engine and wrapped as a single container image running as a Docker task. As the implementation and operation of the data process flow engines are stateless, any task in the service can handle incoming traffic from any gateway.

The data flow coming from the edge gateway must be saved both on HDFS and on a relational database using a Mimosa data model subset. To this end Apache NiFi will receive incoming data, and thanks to dedicate flows, aimed at verifying the goodness and adequacy of the data received, data flow will be saved on the Hadoop file system in a raw format and simultaneously on database containing the metadata describing the raw data received. On top of this data, the analytics component will be able to start processing both data at rest and data in motion.

5. Case study

The implemented platform was deployed at the premises of COMAU and connected to an IoT edge device named robot

box. The robot box intention is to replicate a controlled robotic manipulator testbed for simulating robot failures in an easy and controlled approach for enabling predictive analytics.

One of the activities involved in robot maintenance, is the replacement of belts which are the motion transmitters. The robot box (Figure 4) consists of an electric motor connected to a gearbox using a rubber made belt. The grey slider on the right is used to simulate different belt tensioning states where each status is made repeatable thanks to a centesimal indicator.

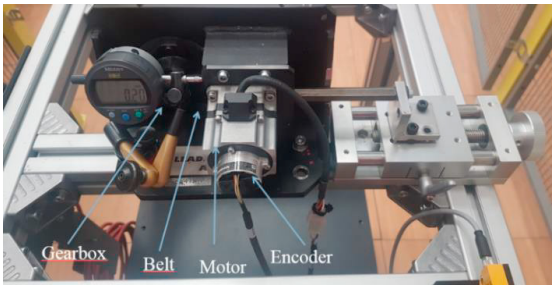

Figure 4: Robot box testbed

The platform deployment at the premises of COMAU involves five virtual machines with a total hardware demand of 16 CPUs, 32 GB of RAM and 500 GB of storage disk. In order to guarantee container state persistence an NFS has been installed on a dedicated VM. It should be noted that at operating speed, the fully deployed SERENA system is expected to be more hardware demanding, this is just the first implementation. A screenshot of the Swarm visualization is presented in Figure 5.

Figure 5: Swarm visualisation of the platform deployed containers.

As discussed in Section 3.2 to test and evaluate the proposed platform two services were integrated into it. As an application case, we considered the belt tensioning level since it is subject to loosening over time by leading to a reduction of robot precision. For this reason, high quality manufacturing processes might require a high number of user inspections, associated with a high cost of inspection and maintenance. To overcome this issue, we proposed to exploit machine learning algorithms for enabling predictive analytics. First, a predictive model was built to model the status of the belt in base of the current and the position signals provided directly by the robot. To this aim each cycle, in terms of current and position data were collected from the robot box, and modelled as feature engineering (e.g. mean, rms, min, max, skewness, kurtosis and so on), representing the input data of the machine learning algorithms. The built model can classify robot cycles in real-time on the analysis of their features computed and estimate the remaining useful life of the robot belt.

The visualization service integrated into SERENA provides a human interface towards the core functionalities of the deployed system in a modular and intuitive way. It can be easily exploited by different users, e.g., maintenance engineers and operators as well as system and gateway administrators, to effective deal with Serena functionalities.

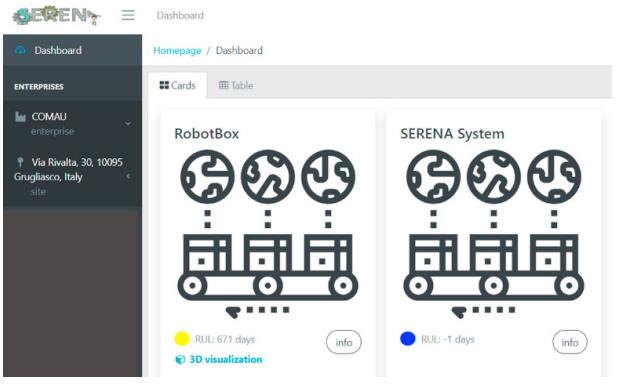

Figure 6: Platform user interface

Next and through the SERENA HMI the outcomes of the predictive analytics service are shows to the users. As an example, the RUL value of the robot box, is shown to the user as presented in Figure 7.

Figure 7: Robot box visualisation and RUL display through the SERENA dashboard

In conclusion, a first deployment of the platform has been installed at the COMAU premises by exploiting existing network and infrastructure. It last 2 working days approximately and the presence of computer experts was required. Nevertheless, the configuration and replacement of components was relatively straightforward with no additional complexities added.

6. Conclusion

This study aims to present the design and implementation of a lightweight microservice based architecture for enabling predictive analytics along with other functionalities and facilitating the maintenance activities of the operators and engineers in manufacturing domains. Moreover, the use of virtualization solutions enables certain independency for specific technologies. The proposed architecture is supported to be scalable and resilient.

A first setup of the platform was performed in an industrial environment with the first results supporting that much time is needed by comparison to a commercial solution but seems promising considering the nature of the development.

Next steps include the further automation of the deployment process as well as the connection of additional services.

Acknowledgements

The research leading to these results has been partially funded by the European Commission under the H2020-IND-CE-2016-17 program, FOF-09-2017, Grant agreement no. 767561 “SERENA” project, VerSatilE plug-and-play platform enabling REmote predictive mainteNAnce.

References

[1] Babiceanu RF, Seker R. Trustworthiness Requirements for Manufacturing Cyber-physical Systems. Procedia Manuf 2017;11:973–81. https://doi.org/10.1016/j.promfg.2017.07.202.

[2] Apiletti D, Baralis E, Cerquitelli T, Garza P, Venturini L. SaFe-NeC: A scalable and flexible system for network data characterization. Proc. NOMS 2016 - 2016 IEEE/IFIP Netw. Oper. Manag. Symp., IEEE; 2016, p. 812–6. https://doi.org/10.1109/NOMS.2016.7502905.

[3] Nikolakis N, Senington R, Sipsas K, Syberfeldt A, Makris S. On a containerized approach for the dynamic planning and control of a cyber - physical production system. Robot Comput Integr Manuf 2020;64:101919. https://doi.org/10.1016/j.rcim.2019.101919.

[4] Cerquitelli T, Bowden D, Marguglio A, Morabito L, Napione C, Panicucci S, et al. A fog computing approach for predictive maintenance. Lect. Notes Bus. Inf. Process., vol. 349, Springer, Cham; 2019, p. 139–47. https://doi.org/10.1007/978-3-030-20948-3_13.

[5] Fuchs J, Oks SJ, Franke J. Platform-based service composition for manufacturing: A conceptualization. Procedia CIRP 2019;81:541–6. https://doi.org/10.1016/j.procir.2019.03.152.

[6] Di Francesco P, Lago P, Malavolta I. Architecting with microservices: A systematic mapping study. J Syst Softw 2019;150:77–97. https://doi.org/10.1016/j.jss.2019.01.001.

[7] Li S, Zhang H, Jia Z, Li Z, Zhang C, Li J, et al. A dataflow-driven approach to identifying microservices from monolithic applications. J Syst Softw 2019;157:110380. https://doi.org/10.1016/j.jss.2019.07.008.

[8] Soldani J, Tamburri DA, Van Den Heuvel WJ. The pains and gains of microservices: A Systematic grey literature review. J Syst Softw 2018;146:215–32. https://doi.org/10.1016/j.jss.2018.09.082.

[9] Bonomi F, Milito R, Zhu J, Addepalli S. Fog computing and its role in the internet of things. MCC’12 - Proc. 1st ACM Mob. Cloud Comput. Work., New York, New York, USA: ACM Press; 2012, p. 13–5. https://doi.org/10.1145/2342509.2342513.

[10] Naha RK, Garg S, Georgakopoulos D, Jayaraman PP, Gao L, Xiang Y, et al. Fog computing: Survey of trends, architectures, requirements, and research directions. IEEE Access 2018;6:47980–8009. https://doi.org/10.1109/ACCESS.2018.2866491.

[11] Ali S, Jarwar MA, Chong I. Design methodology of microservices to support predictive analytics for IoT applications. Sensors 2018;18:4226. https://doi.org/10.3390/s18124226.

[12] Mihai V, Popescu D, Ichim L, Drăgana C. Fog computing monitoring system for a flexible assembly line. Stud. Comput. Intell., vol. 853, 2020, p. 197–209. https://doi.org/10.1007/978-3-030-27477-1_15.

[13] Short M, Twiddle J. An industrial digitalization platform for condition monitoring and predictive maintenance of pumping equipment. Sensors 2019;19:3781. https://doi.org/10.3390/s19173781.

[14] Docker Inc. Empowering App Development for Developers | Docker n.d. https://www.docker.com/ (accessed February 18, 2020).

[15] Celesti A, Mulfari D, Galletta A, Fazio M, Carnevale L, Villari M. A study on container virtualization for guarantee quality of service in Cloud-of-Things. Futur Gener Comput Syst 2019;99:356–64. https://doi.org/10.1016/j.future.2019.03.055.

[16] Wan X, Guan X, Wang T, Bai G, Choi BY. Application deployment using Microservice and Docker containers: Framework and optimization. J Netw Comput Appl 2018;119:97–109. https://doi.org/10.1016/j.jnca.2018.07.003.

[17] Nikolakis N, Senington R, Sipsas K, Syberfeldt A, Makris S. On a containerized approach for the dynamic planning and control of a cyber - physical production system. Robot Comput Integr Manuf 2020;64:101919. https://doi.org/10.1016/j.rcim.2019.101919.

[18] Kiss T, Kacsuk P, Kovacs J, Rakoczi B, Hajnal A, Farkas A, et al. MiCADO—Microservice-based Cloud Application-level Dynamic Orchestrator. Futur Gener Comput Syst 2019;94:937–46. https://doi.org/10.1016/j.future.2017.09.050.

[19] Štefanič P, Cigale M, Jones AC, Knight L, Taylor I, Istrate C, et al. SWITCH workbench: A novel approach for the development and deployment of time-critical microservice-based cloud-native applications. Futur Gener Comput Syst 2019;99:197–212. https://doi.org/10.1016/j.future.2019.04.008.

[20] Butzin B, Golatowski F, Timmermann D. Microservices approach for the internet of things. IEEE Int. Conf. Emerg. Technol. Fact. Autom. ETFA, vol. 2016- November, IEEE; 2016, p. 1–6. https://doi.org/10.1109/ETFA.2016.7733707.

[21] Mimosa.org. MIMOSA Open Standards for Physical Asset Management. MIMOSA Consort Website n.d. https://www.mimosa.org/ (accessed February 18, 2020).

[22] Bowden D, Marguglio A, Morabito L, Napione C, Panicucci S, Nikolakis N, et al. A cloud-to-edge architecture for predictive analytics. vol. 2322. 2019