Available online at www.sciencedirect.com

ScienceDirect

Procedia Manufacturing 11 (2017) 931 – 938

Procedia MANUFACTURING

27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy

A microservice-based middleware for the digital factory

Michele Ciavotta[a,]*, Marino Alge[a], Silvia Menato[a], Diego Rovere[b], Paolo Pedrazzoli[b]

[a]*University of Applied Sciences of Southern Switzerland, Via Galleria 2, Manno, CH-6928 Switzerland*
[b]*Technology Transfer System S.r.l, Via Francesco d'Ovidio, 3, Milano, 20131, Italy*

Abstract

In recent years a considerable effort has been spent by research and industrial communities in the digitalization of production environments with the main objective of achieving a new automation paradigm, more flexible, responsive to changes, and safe. This paper presents the architecture, and discusses the benefits, of a distributed middleware prototype supporting a new generation of smart-factory-enabled applications with special attention paid to simulation tools. Devised within the scope of MAYA EU project, the proposed platform aims at being the first solution capable of empowering shop-floor Cyber-Physical-Systems (CPSs), providing an environment for their Digital Twin along the whole plant life-cycle. The platform implements a microservice IoT-Big Data architecture supporting the distributed publication of multidisciplinary simulation models, managing in an optimized way streams of data coming from the shop-floor for real-digital synchronization, ensuring security and confidentiality of sensible data.

Peer-review under responsibility of the scientific committee of the 27th International Conference on Flexible Automation and Intelligent Manufacturing

Keywords: Microservices; Real-digital synchronization; Cyber Physical Systems; Data Stream Analysis; Smart Factory; Industrie 4.0

* Corresponding author.
E-mail address: michele.ciavotta@supsi.ch

Peer-review under responsibility of the scientific committee of the 27th International Conference on Flexible Automation and Intelligent Manufacturing
doi:10.1016/j.promfg.2017.07.197

1. Introduction

Manufacturing has always been an extremely competitive field wherein the players strive to build increasingly more efficient and flexible solutions in order to take on challenges dictated by a global economy. In particular, the worldwide competition carries on the necessity for mass-customization to meet capricious customers' trends, and consequent unpredictable workloads. Such scenario calls for scalable, fast reconfigurable automation systems, as much as possible integrated in the Enterprise Information Systems (EIS). The Industrial Internet, that is the convergence of industrial manufacturing and Information and Communication Technologies (ICT), is generally considered the core of the forth industrial revolution (*Industrie 4.0*). ICT trends as Cloud Computing, Big Data, Internet of Things (IoT)[1] and CPS (Cyber-Physical Systems) [2] are the innovation drivers towards this paradigm shift - Smart Factory [3] - that merges at various levels automation and computation.

In recent years an ever-growing number of industrial devices came with embedded computational capacity; they are usually referred to as Cyber-physical systems. Gartner reports[1] that in 2016 around 1.2 billion CPSs are active, estimating a steady growth up to 2.9 billion devices by 2020. It is noteworthy that those numbers, accounting for an annual turnover of $991 billion, do not include consumer (tablets, smartphone and computers) or cross-industry (as light bulbs or sensors) devices. In the production context, CPSs are increasingly replacing classical PLCs since atop their flexible ubiquitous paradigm more intelligent, automated manufacturing processes can be built. Furthermore, CPSs are able to communicate (via closed industrial networks but are often also over the Internet) with other CPSs and with enterprise software (like ERP, SCADA, MES and Simulation engines) introducing that modularity, service orientation, and decentralized automation, theorized by Industrie 4.0 and predicted to flatten the automation pyramid and lead to large scale distributed and decentralized automation solutions.

The integration between automation and information systems entails the creation of a heterogeneous ecosystem where industrial CPSs, software middleware, and enterprise applications seamlessly interact using the protocols of the Internet of Things (IoT). In this context, CPSs assume a digital (virtual) nature in addition to the Cyber-Physical one. Specifically, if on the one hand a CPS is equipped with computational on-board capabilities (Cyber nature), able to sense, control and react to changes in the shop-floor (Physical nature), on the other it also provides a digital interface to allow the integration within the EIS (Virtual nature). This digital interface is often referred to as CPS alter ego or *Digital Twin* (DT), borrowing the expression from the IoT world. Fundamentally, a Digital Twin is a digital avatar encompassing CPS data and intelligence; it represents structure, semantics and behavior of the associated CPS, providing services to mesh the virtual and physical worlds.

Despite the success of IoT witnessed in the last decade, the adoption of IoT/CPS deployments in manufacturing is still moving the first steps for a number of reasons including lack of suitable standards and recognized architectural references to achieve interoperability and digital continuity. Furthermore, industrial CPSs feature security and real-time management issues that make them fundamentally different from other IoT devices; for this reason, current deployments are implemented on an ad-hoc fashion, over focusing on unidirectional data collection for the shop-floor for monitoring purpose withal [4].

In this work we present the implementation of a distributed middleware developed within the frame of MAYA European project[2], tailored to enable scalable interoperability between enterprise applications and CPSs with especial attention paid to simulation tools. The proposed platform strives for being the first solution based on both Microservices [5] [6] and Big Data [7] paradigms to empower shop-floor CPSs along the whole plant life-cycle, and realize real-digital synchronization ensuring at the same time security and confidentiality of sensible factory data.

In the remainder of this paper, the overall MAYA ecosystem is presented in Section 2. Then, in Section 3 is detailed the architecture of our middleware. Section 4 discusses some design choices. Finally, conclusions and future steps are drawn in Section 5.

---

[1] Gartner Inc. http://www.gartner.com/newsroom/id/3165317

[2] http://maya-euproject.com/

2. MAYA platform: a vision of the future of manufacturing

Before presenting the proposed microservice based infrastructure in details, we deem important provide a brief introduction to the overall simulation-oriented platform for smart factories envisioned by MAYA. A reference usage scenario is presented as well as guidelines to understand the objectives and the architectural decisions.

2.1. MAYA Platform

MAYA foresees for the forthcoming smart factory the flattening of the automation pyramid. To this end, for research purposes, the project proposes a distributed platform wherein three main components are involved in various capacities:

MAYA Communication Layer (MCL) is a middleware consisting of a runtime environment for distributed automation software. Its role in the platform is to enable aggregation, discovery, orchestration and seamless communication among CPSs, at shop-floor level and with the rest of the platform.

MAYA Support Infrastructure (MSI): this is the component this research hinges on; it is an Microservice/Big Data middleware in charge of managing the DTs along the factory life-cycle, enabling definition, enrichment via data processing, and dismissal. In particular, it provides functionalities for distributed publication of multi-disciplinary simulation models, for real-to-digital synchronization, enforcing security and confidentiality of sensitive data.

MAYA Simulation Framework (MSF): This component is a dedicated runtime for the concurrent execution and orchestration multi-disciplinary simulators. Supporting this approach requires that any MAYA-enabled simulation tools must be able to use in-process the outcomes of other engines, activating in real-time models already published by other tools and synchronizing the execution.

It is important to notice that, in order to support CPS multi-disciplinary simulation and real-to-digital synchronization, MAYA envisions the DTs as entities featuring two types of assets, namely *Simulation* and *Functional models*. These latter ones, in particular, are used to process the data gathered at shop-floor level. The results generated by the functional models are, in turn, used to update CPS nameplate information. Enterprise applications as simulators can greatly benefit from up-to-date information of the factory, which can ultimately be considered as a single CPS.

A graphical representation of MAYA platform components and their relationships is presented in Figure 1 a). The image shows a direct link between MSI and MSF meaning that the former provides a set of services to the latter. Examples of those services are: services that store, retrieve and return DTs, simulation models, endpoints to save the results of the simulation along with the related configuration files, and services for authentication and privacy enforcement.

2.2. MAYA Support Infrastructure

As discussed above, the main objectives of MSI are: managing the life-cycle of Digital Twins to support simulation and providing suitable mechanisms for the CPS-to-DT synchronization. Essentially, MSI implements the following functionalities:

CPS registration and login. The platform enables the creation of new CPSs and implements a machine-to-machine (M2M) protocol for CPS login and logout. Presently, for safety reasons, only human operators can create new CPSs.

Creation of communication channels. Once the CPS is set up and logged in it may require to push data to the platform for online/offline processing; consequently, a suitable mechanism to create a WebSocket channel is provided.

CPS querying. MSI implements a protocol for querying over the set of DTs. This has been designed to be as flexible as possible in order to meet also future requirements. In particular, the platform permits trusted actors the execution of CRUD (Create, Read, Update, Delete) operations to search, filter, and manipulate CPS-related information.

Furthermore, in order to support simulation in all phases of the factory life-cycle, it is important to ensure that the DTs mirror constantly and faithfully the state of CPSs. The reader should consider, in fact, that CPS nameplate values may change over time due to age and strain. Thereupon MAYA envisions the future CPSs as equipped with special assets named Functional Models (FMs) to be uploaded to MSI for synchronization and data analysis. FMs are essentially software routines that are run against data sent by the CPSs. Such routines can regularly update CPS reference values, estimate indirect metrics or train predictive maintenance models. FMs are fully managed (registered, executed, and monitored) by the MSI middleware itself.

2.3. Usage scenario

Several usage scenarios are possible. Nonetheless, we propose the following as a reference use case, as it involves all the components of MAYA platform touching a good part of MSI functionalities. The objective is to use it as a reading key to better understand the relationships among MAYA components and how they are reflected into the architecture of MSI.

1) A human operator registers a new CPS. This action can be performed via the graphical UI or by means of available REST endpoints;
2) The CPS logs in on MSI, its digital identity is verified and the Digital Twin is created or activated;
3) The Functional Model featured by the Digital Twin (if any) is set up, scheduled, and executed;
4) WebSocket channel is established between the CPS and MSI. The CPS starts sending data to the platform.
5) The Functional Model periodically generates updates for a subset of attributes of the corresponding DT;
6) The MSF accesses CPS Digital Twin and the related simulation models and performs the simulation.

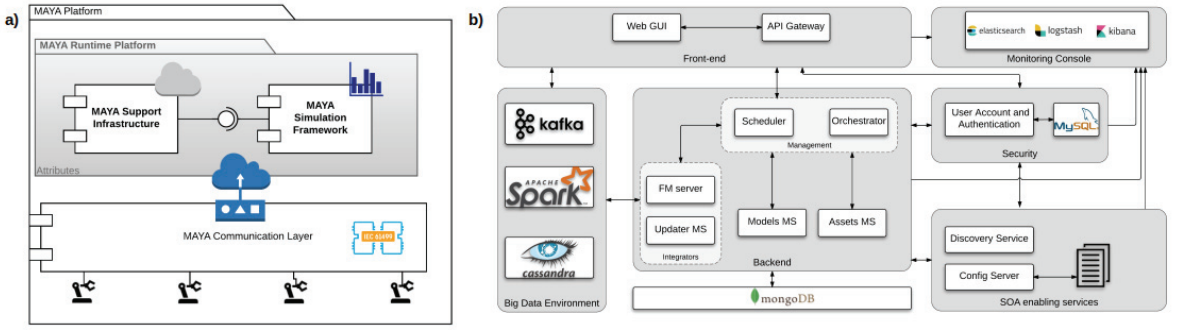

Figure 1: a) High-level Architecture of MAYA platform b) MSI service diagram

3. Architecture

The dichotomy observed at functional level can be find reflected in the MSI architecture. Figure 1 b) presents the services; a relevant part of the platform consists of a microservice-based infrastructure devoted mainly to administrative tasks related to Digital Twins. The remainder is a Big Data deployment accountable for processing shop-floor data. Since the two portions of our middleware have different requirements, being also grounded on different technological solutions, in what follows they are presented and discussed separately.

3.1. Microservice platform

In a nutshell, the microservice-based architecture is the evolution of the classical Service Oriented Architecture (SOA) [6] in which the application is seen as a suite of small services, each devoted to as single activity. Within the MSI each microservice exposes a small set of functionalities and runs in its own process, communicating with other services mainly via HTTP resource API or messages. Five groups of services can be identified and discussed below.

Front-end services are designed to provide the MSI with a single and secure interface to the outer world. As a consequence, any other service can be accessed only through the front-end and only by trusted entities. The main services in this group are the *Web-based UI* and the *API Gateway*. The former is a web application for human-machine interaction; it provides a user friendly interface to register new CPSs or to execute queries. Administration tools such as security management and platform monitoring are available as well. The *API Gateway*, instead, is a service designed to provide dynamic and secure API routing, acting as a front door for the requests coming from authorized players, namely users via the web UI and devices/CPSs executing REST/WebSocket calls. The gateway is based on Netflix Ribbon[1], a multi-protocol inter process communication library that, in collaboration with Service Registry (see SOA enabling services), dispatch incoming requests applying load-balance policy. The API gateway, finally, offers an implementation of the Circuit Breaker[2] pattern impeding the system to get stuck in case the target back-end service fails to answer within a certain time.

Security: implemented by the *User Account and Authentication* (UAA) service, which is in charge of the authentication and authorization tasks; it checks users' (human operators, CPSs or microservices) credentials to verify the identity and issuing a time-limited Oauth2[3] token to authorize a subset of possible actions that depends on the particular role the user has been assigned to.

SOA enabling services: this group of services has the task to support the microservice paradigm; it features:

Service Registry: It provides a REST endpoint for service discovering. This service is meant to allow transparent and agnostic service communication and load balancing. Based on Netflix Eureka[4], it exposes APIs for service registration and for service querying, allowing the services to communicate without referring to specific IPs. This is especially important in the scenario in which services are replicated in order to handle a high workload.

Configuration server: the main task of this service is to store properties files in a centralized way for all the micro-services involved in MSI. Among the benefits of having a configuration server we mention here the ability to change the service runtime behavior in order to, for example, perform debugging and monitoring.

Monitoring console: this macro-component with three services implements the so-called ELK[5] stack to achieve log gathering, analyzing and monitoring services. In other words, logs from every microservice are collected, stored, analyzed and presented in graphical form to the MSI administrator. A query language is also provided to enable the administrator to interactively analyze the information coming from the platform.

Backend services: to this group belong those services that manage the creation, update, deletion, storage, retrieve and query of CPS Digital Twins. In particular, the *Orchestrator* and *Scheduler* microservices coordinate and organize other services to create high-level composite business processes. On the other hand, *Models* and *Assets* services handle the persistence of Digital Twin information (their representation and assets, respectively) providing endpoints for CRUD operations. Finally, the *FM server* and *Updater* interact with the Big Data platform to submit, monitor the execution, and retrieve data generated by the functional models.

3.2. Big Data environment

Big Data technologies are becoming innovation drivers in industry [8]. Big Data algorithms are often used for predictive maintenance, with the aim of reducing downtimes and costs. Similarly, MSI is required to handle unprecedented volumes of data generated by the digital representation of the factory in order to keep updated the CPS nameplate information. To this end, a data processing platform, specifically a Lambda architecture [9], has been implemented according to best practices of the field. In particular, a both *data in rest* and *data in motion* patterns are enforced by our platform, making it suitable for both stream and batch processing. The Lambda architecture encompasses three layer, namely *batch*, *speed* and *serving* layers. The batch layer is appointed to the

---

[1] https://github.com/Netflix/ribbon
[2] https://martinfowler.com/bliki/CircuitBreaker.html
[3] https://oauth.net/2/
[4] https://github.com/Netflix/eureka
[5] https://www.elastic.co/webinars/introduction-elk-stack

analysis of large quantities (but still finite) of data whereas the speed layer is in charge of processing infinite streams of information. In our implementation both layers have been implemented using Apache Spark[1] in standalone mode. As far as the serving layer is concerned, it serves to store results of the processing layers in such a such a way that outcomes can be retrieved using ad-hoc queries or precomputed views. We implemented this layer using Apache Cassandra[2], a NoSQL database particularly suitable for fast updates. Apache Cassandra is also used as data source for the batch layer, whereas the speed layer is fed by stream of data produced by CPSs and canalized within a queue system. For this role we selected Apache Kafka[3] for being fast, reliable, persistent, distributed and well-supported by Spark and Cassandra. The resulting platform is a fully open solution that has been proven to provide important characteristics of scalability and reliability.

4. Discussion

MSI aims at being the first reference middleware for smart factories based on a composite Microservices/Big Data approach paying particular attention to security concerns. In what follows we scrutinize the reasons behind our architectural choices.

4.1. Microservices

The adoption of the microservice paradigm provides several benefits but also presents inconveniences and new challenges. Among the benefits of this architectural style, the following must be enumerated:

Agility: microservices fit into the Agile/DevOps development methodology [10], enabling business to *start small and innovate fast* by iterating on their core products without affording substantial downtimes. A minimal version of an application, in fact, can be created in shorter time reducing time-to-market and up-front investment costs, and providing an advantage with respect to competitors. Future versions of the application can be realized by seamlessly adding new microservices.

Isolation and Resilience: Resiliency is the ability of self-recovery after a failure. A failure in a monolithic software can be a catastrophic event as the whole platform must recover completely. In a microservice platform, instead, each service can fail and heal independently with a possibly reduced impact on the overall platform's functionalities. Resilience strongly dependents on compartmentalization and containment of failure, namely *Isolation*. Microservices can be easily containerized and deployed as single processes, reducing the probability of cascade-fail of the overall application. Isolation, moreover, enables reactive service scaling, and independent monitoring, debugging, and testing.

Elasticity: A platform can be subject to variable workloads especially on seasonal basis. Elasticity is the ability to respond to workload changes provisioning or dismissing computational power. This is usually translated into scaling up and down services. This process can be particularly painful and costly in case of on premise software; easier and automated in case of cloud based applications. Nonetheless, microservices allows for a finer grain approach in which services in distress (e.g., that are not meeting their Quality of Service), can be identified and singularly scaled provisioning of just the right amount of resources.

As far as the challenges derived by the choice of adopting microservices are concerned, we mention here:

The management of distributed data. As each microservice might have its private database it is difficult to implement business transactions that maintain data consistency across multiple databases.

The higher complexity of the system. Proliferation of small services could translate into a tangle web of relationships among them. Experienced teams must be put together to deal in the best possible way with microservice platforms.

---

[1] http://spark.apache.org/
[2] http://cassandra.apache.org/
[3] https://kafka.apache.org/

4.2. Big Data

The phrase *Big Data* usually refers to a large research area that encompasses several facets. In this work, in particular, we refer to Big Data architectures. The following benefits deserve to be enumerates:

Simple but reliable. The Big Data platform has been implemented employing a reduced number of tools; all of them are considered state of the art, are used in production by hundreds of companies worldwide, and are backed by large communities and big ICT players.

Multi-paradigm and general purpose; that is Batch and Stream processing as well as ad-hoc queries are supported and can run concurrently. Moreover, the unified execution model, coupled with a large set of libraries, permits the execution of complex and heterogeneous tasks (as machine learning, data filtering, ETL, etc.).

Robust and Fault tolerant: in case of failure the data processing is automatically rescheduled and restarted on the remaining resources.

Multi-tenant and Scalable: in MAYA this means that several Functional Models can run in parallel sharing computational resources. Furthermore, new resources can be provisioned and the platform would start to exploit them without downtimes.

The downside of this approach is that it is fundamentally and technologically different for the rest of the platform and required quite an integration work. Moreover, Big Data solutions requires steep learning curves to be used and are generally really resource eager.

4.3. Security

Security and privacy issues assume paramount importance in Industrial IoT [11] [12]; in MAYA we enforce those aspects since the earliest stages of the design, focusing on suitable Privacy-Enhancing Technologies (PETs) that encompass Authentication, Authorization and Encryption mechanisms. More in details, authentication is the process of confirming the identity of an actor in order to avoid possibly malicious accesses to the system resources and services. Authentication can be defined as the set of actions a software system has to implement in order to grant the actor the permissions to execute an operation on one or more resources. Specifically, seeking for more flexibility we adopted an extension of SecureUML [13] role-based access control model that permits the authentication process to depend on the actor's role. Suitable Authentication/Authorization mechanisms (based on the Oauth2 protocol) have been developed for human operators, and services and CPSs. Securing communication is the third piece of this security and privacy puzzle, as no trustworthy authentication and authorization mechanism can be built without the previous establishment of a secure channel. For this reason, MAYA platform committed to employ state-of-the-art encryption mechanisms (e.g. SSL and TLS) for the communication and data storage as well.

5. Conclusions and future work

This paper presented MAYA Support Infrastructure, an IoT middleware designed to support simulation in smart factories. To the best of our knowledge it represents the first example of Microservice platform for manufacturing. Since security and privacy are sensitive subjects for the industry, special attention has been paid on their enforcement from the earliest phases of the project. The proposed platform has been described in details in connection with CPSs and simulators. Lastly, the overall architecture has been discussed along with benefits and challenges.

Preliminary tests carried out on the platform are encouraging, but there is still much work to do in many aspects. In particular, our future steps will include the deployment of the middleware on real manufacturing plants in order to validate the scalability capabilities of the platform under different workload conditions. Furthermore, we plan to refine and formalize the interaction protocol with CPSs including other, more IoT-friendly, protocols like XMPP and MQTT. The reason behind this choice is facilitate the integration with legacy or heterogeneous solutions. Finally, we intent to support other enterprise applications other than simulation. To ease the process, we are studying the possibility of basing the inter-application communications on standard data exchange format as AutomationML [14].

Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreements No 678556 and 723094.

References

[1] Z. Bi, L. Da Xu, and C. Wang, "Internet of things for enterprise systems of modern manufacturing," *IEEE Trans. Ind. Informatics*, vol. 10, no. 2, pp. 1537–1546, 2014.
[2] N. Jazdi, "Cyber physical systems in the context of Industry 4.0," *2014 IEEE Autom. Qual. Testing, Robot.*, pp. 2–4, 2014.
[3] E. Hozdić, "Smart factory for industry 4.0: A review," *Int. J. Mod. Manuf. Technol.*, vol. 7, no. 1, pp. 28–35, 2015.
[4] G. Privat, M. Zhao, and L. Lemke, "Towards a shared software infrastructure for smart homes, smart buildings and smart cities," in *International Workshop on Emerging Trends in the Engineering of Cyber-Physical Systems, Berlin*, 2014.
[5] N. Dragoni *et al.*, "Microservices: yesterday, today, and tomorrow," in *Present and Ulterior Software Engineering*, Springer Berlin Heidelberg, 2017.
[6] S. Newman, *Building microservices*. " O'Reilly Media, Inc.," 2015.
[7] J. Manyika *et al.*, "Big data: The next frontier for innovation, competition, and productivity," 2011.
[8] C. Yang, W. Shen, and X. Wang, "Applications of Internet of Things in manufacturing," in *Proceedings of the 2016 IEEE 20th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2016*, 2016, pp. 670–675.
[9] N. Marz and J. Warren, *Big Data: Principles and best practices of scalable realtime data systems*. Manning Publications Co., 2015.
[10] M. Httermann, *DevOps for developers*. Apress, 2012.
[11] A.-R. Sadeghi, C. Wachsmann, and M. Waidner, "Security and privacy challenges in industrial internet of things," *Proc. 52nd Annu. Des. Autom. Conf. - DAC '15*, vol. 17, pp. 1–6, 2015.
[12] A. Razzaq, A. Hur, H. F. Ahmad, and M. Masood, "Cyber security: Threats, reasons, challenges, methodologies and state of the art solutions for industrial applications," *2013 IEEE Elev. Int. Symp. Auton. Decentralized Syst.*, pp. 1–6, 2013.
[13] T. Lodderstedt, D. Basin, and J. Doser, "SecureUML: A UML-based modeling language for model-driven security," in *International Conference on the Unified Modeling Language*, 2002, pp. 426–441.
[14] R. Drath, A. Luder, J. Peschke, and L. Hundt, "AutomationML-the glue for seamless automation engineering," in *Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on*, 2008, pp. 616–623.