A Survey: Microservices Architecture in Advanced Manufacturing Systems

Aydin Homay
Faculty of Computer Science
Technische Universität Dresden
Dresden, Germany
https://orcid.org/0000-0002-6425-7468

Alois Zoitl
LIT | CPS Lab
Johannes Kepler University Linz
Linz, Austria
alois.zoitl@jku.at

Mário de Sousa
Faculty of Enigineering
University of Porto
Porto, Portugal
msousa@fe.up.pt

Martin Wollschlaeger
Faculty of Computer Science
Technische Universität Dresden
Dresden, Germany
martin.wollschlaeger@tu-dresden.de

***Abstract*— Plug & Produce is an important part of the 4th industrial revolution (I4.0) that focuses on product development and production scenarios, demanding rapid adaptation to customer needs, with minimum effort, and no business interruption. Achieving this requires the use of flexible and agile automation systems. Microservices Architecture can be an opportunity to bring such flexibility and agility to industrial automation systems. Since Microservices Architecture is a new paradigm in service-oriented systems, an in-depth analysis from the industrial automation perspective is necessary in order to figure out why and how this new architecture could be helpful to build flexible automation systems. In this paper we first clearly describe and characterize the Microservices Architecture. We next discuss an existing paradox in microservice granularity, and suggest an approach to solve the mentioned paradox. Then we compare this new approach to the Service Oriented Architecture, highlight the key (dis)advantages of utilizing Microservice Architecture in building flexible automation systems that fits in 4th industrial revolution needs, and describe the challenges to its future adoption.**

Keywords—Automation, Microservices, Granularity, Flexibility, Production, Function Block.

I. INTRODUCTION

The 4th industrial revolution (I4.0) focuses on product development and production scenarios [1]. More specifically, it demands rapid adaptation to customer needs, with minimum effort, and no business interruption. This requires an evolution in automation systems to improve their flexibility and agility.

I4.0 tries to adapt the Plug & Produce approach, which is designed by analogy of the Plug & Play concept from the computer world. Plug & Produce is an architecture that tries to ease the installation or removal of a (new) device from an production system without any change to configuration or re-programming [2].

I4.0 is itself inherently heterogeneous across application infrastructures and contains a complex computing environment. High interoperability and predominantly loose modularity coupling could therefore facilitate the adoption of Plug & Produce architecture in I4.0. The Service Oriented Architecture (SOA) was very successful in building loosely coupled modules, commonly referred to as services in large enterprise systems. Its new iteration, the Microservices Architecture (MSA), tries to strip away the unnecessary complexities that SOA has in order to improve several aspects such as flexibility, interoperability, independence, and scalability [3]–[7].

In this paper we evaluate the possible use of Microservices Architecture with a view of improving the flexibility, interoperability, and scalability of automation and manufacturing systems.

Section II of the paper presents related work in the area. Background is discussed in section III, while section IV provides a short survey of MSA and its key characteristics including differences with SOA from several selected papers, books, and industrial reports. Section V discusses the key benefits of MSA in Industrial Automation and Adaptive Production Systems. Section VI explains the MSA anti-patterns and pitfalls, while the challenges that need to be tackled by using MSA in Industrial Automation are discussed in section VII.

II. RELATED WORK

There are successful studies showing that SOA can bring flexibility and high interoperability to industrial automation. The authors of [8] used a Domain Specific Language (DSL) to implement a new IEC61499 function block in 4diac™ [9] that uses MQTT (ISO standard publish-subscribe-based messaging protocol) to expose its functionality. They showed how this approach made a Festo Didactic batch process plant more flexible in terms of using different recipes. The authors of [10] presented an architecture to enhance SOA with real-time capability in industrial automation.

The iLAND project [11] introduced a middleware that supports service-oriented time deterministic reconfiguration in distributed soft real-time environments. The published results demonstrated the efficient behavior of the designed system.

The authors of [12] defined a set of criteria to evaluate the application of SOA in creating communication infrastructure for modular process automation. They concluded that SOA fulfills all defined criteria in their study. In [13] the authors show how SOA and OPC-UA can bring flexibility and reusability to industrial automation systems, resulting in the optimization of manufacturing processes. They used three different frameworks for development of services and chose the one with minimum execution time.

A. Ismail, in his doctoral dissertation [14], considers service-orientation in manufacturing systems to tackle agility, interoperability, and flexibility requirements that I4.0 based manufacturing systems need.

C. Stoidner et al. [15] introduces a solution called SOAP4PLC. In that proposed approach a web service invokes an IEC 61131-3 POU (Program Organization Unit) [16]. The web service is named a *SOA function block*. The work does not however present why such a web-service should be considered a service in the SOA sense, and does not describe the service criteria such as size and granularity.

K. Thramboulidis et al. introduced a Cyber-physical approach based on Microservices Architecture (CPMS) in [17]. The CPMS approach benefits from service-oriented features. Then they used the proposed approach in [18] to setup a new Assembly System (AS) according to Assembly System 40 (AS40) [19]. Unfortunately, this work does not clearly state why their service-oriented approach should be considered a MSA instead of standard SOA.

W. Dai et al. [20] map the IEC 61499 function blocks to SOA based services. They introduce a formal mapping that maps each function block into a service. The study does not however consider defining any criteria for service granularity and composition. This could lead to another problem in service communication. It may also add extra complexity by introducing semantic tight coupling.

III. Background

Distributed enterprise applications have evolved through several technological evolutionary changes to tackle reusability, interoperability and modularity of large systems. Prominent examples are Client-Service Architecture, Remote Procedure Call (RPC), Distributed Computing Environment (DCE), Distributed Component Object Model (DCOM), Component Object Request Broker Architecture (CORBA), Object-Oriented Architecture (OOA), Service-Oriented Architecture (SOA), and now Microservices Architecture (MSA), and perhaps later Nanoservices Architecture (NSA). Without understanding the term “Service” it is not possible to achieve a clear understanding about SOA, just as without a clear idea of the definition and meaning of “Object” in OOA it is not possible to understand what object-orientation is about and why it is so useful to apply in system architecture, design, and implementation.

A. What is a service?

Oxford Dictionary defines service as an “action of helping or doing work for someone.” Other dictionaries suggest similar definitions.

In [21] and [22], a service is defined as a software component with finite behavior, and an interface to realize the described functionality as a request and a response mechanism and with the possibility of defining specific properties which would be called Quality of Service (QoS).

In the SOA context, [23] gives a more specific definition to the term service - i.e. a service encapsulates different components into a single interface to handle a discrete business function.

B. Service Oriented Architecture

Flexibility is the silver bullet of SOA. In a very simple and abstract way, [24] defines SOA in three words: Interoperability by introducing Enterprise Service Bus (ESB), Self-contained business functionality, and Loose coupling. According to [25], SOA defines guidelines, principles, and techniques to classify related business processes into independent services that later can be combined with other available services in a network through service orchestration to create higher-level composite services and applications. SOA aims at composing similar functionalities from the process layer (or the business layer) into a coarse-grained service and implementing very fine-grained services in fundamental layer (the infrastructure layer) with the idea of sharing the basic functionalities like, read/write from/into database, etc. as much as possible with other services by exposing them in federation (orchestration) and process (business) layer. This approach leads SOA to a horizontal service composition [25].

IV. Microservices Architecture

S. Newman [6] defines microservices as small services that all work together. The word small means do one thing and do it well, as Robert C. Martin`s says about Single Responsibility Principle, “Gather together those things that change for the same reason, and separate those things that changes for different reasons.” This can be expressed in one simple word: cohesive.

Recent studies [3]–[7] show that the MSA is the second iteration of SOA which tries to strip away unnecessary levels of complexity in order to narrow down functionality and individuality of service and make them more flexible, independent, easily replaceable, and individually deployable.

The term “Microservices” was first introduced in 2011 to implement only functionalities strongly related to the concern that it is meant to model (cohesive), and also wrap the implemented functionalities into an independent process interacting via messages, ideally in a stateless way [3].

MSA and SOA differ extremely in terms of some conceptual views. Following is a short list of characteristics in which they differ [26]:

A. Service Composition

Unlike SOA, there is no horizontal service composition in MSA. Similar business activities are therefore independent services with all related functionalities aiming at satisfying a particular business need, rather than a coarse-grained service that handles the entire business process in one service implementation.

B. *Granularity*

Coarseness and fineness are two different extremes in service granularity. A service could be more coarse-grained or more fine-grained based on the number of business functionalities that are being exposed by the service.

Unlike SOA, MSA is single task oriented (task in terms of business responsibility), and provides a very narrow business functionality. Building independent and fine-grained services is the essence of this architecture [7].

As a result, this amount of independency and self-containment introduces a paradox in service granularity of MSA - the number of functionalities that a service needs to contain in order to fulfill its own single business functionality without creating extra dependencies on other services (see self-containment and independency definition of microservices) requires additional functionalities to be added into the service, mostly from the infrastructure layer. These extra functionalities increase the service granularity. To solve this paradox, we suggest shifting granularity to design level rather than implementation level. In essence, we will consider granularity as semantic-size.

C. *Service Size*

Shifting granularity to design level and not considering infrastructure functionalities as part of service granularity demands a new metric to keep an eye on the vertical service dimension (physical size). We therefore differentiate the size of service from its granularity, and we suggest the use of service size to measure vertical dimension of a microservice.

D. *Bounded Context*

SOA tries to share as much as possible, such as sharing the application and infrastructure services from fundamental layer to process layer with enterprise services, while MSA tries to share as little as possible by relying on the *bounded-context* notion [27] originating from DDD (Domain Driven Design) [28].

E. *Service Choreography*

SOA mostly relies on service orchestration rather than service choreography to cooperate and bring up elaborate functionalities, but MSA prefers relying only on service choreography and using events and publish/subscribe mechanisms (similar to REST) to establish collaboration between services [3], [29] (instead of request and response mechanisms [6]).

F. *Independency/Self-contained*

Microservices are individually deployable and operationally independent. In other words microservices should contain everything that they need to fulfill their task on their own. This includes not also the business logic but also its front and back-end, as well as all required libraries. The only form of communication between services is through their published interfaces [7], [30].

G. *Share Nothing/ Individual Storage*

Microservices own their own data storage - each Microservice needs to have it is own persistence strategy and should not care how data persists behind another service. The persistence methods should be easily exchangeable [7]. This characteristic of MSA strips away the complexity of handling Data-Flow Paradigms in SOA [31].

H. *Statelessness*

A service can be stateful or stateless. Having a stateless service can bring more flexibility, high availability, load balancing, on-demand elasticity, and high reliability through service redundancy. On the other hand, a stateful service requires more effort for state tracking, synchronization, failure recovery, and brings less scalability to the system [32].

I. *Service Discovery and Deployment*

Unlike SOA, in MSA the deployment process is very important and challenging as the services are narrowly functionalized in small, independent and highly decoupled software pieces. Fundamentally each microservice shall be deployed independently. This means that having a solid deployment environment that provides a standalone execution platform is an essential part of MSA. The container-based visualization such as Docker, Zookeeper, and Kubernetes are very well known platforms that provide such an environment [33]–[35].

V. Challenges to be Tackled

The IEC61499 and IEC61131-5 contain communication function blocks that support major protocols such as TCP, UDP, Modbus and some other related ones. Therefore, adapting service-oriented protocols is not a challenge especially in the case of IEC61499 that contains by nature its own communication protocol stack. Fig. 1 shows an example of a such function block in IEC61499 standard.

The Microservice Function Block (MSFB) refers to the idea of exposing an IEC61499 or IEC61131-3 Function Block as a Service (FBaaS). The expected function block will have an identifier as an access route (end-point), an internal functional block that handles Constrained Application Protocol (CoAP), or any other protocol such as WS-* family, REST, and Feed/Stream, and an output that provides result as a JSON, Stream, or XML format. Implementation of an MSFB does not seem to be a big challenge. The only major effort would be the extension of the communication protocol stack to support the suitable protocols, like the work accomplished in [8]. Adapting microservices architecture into model-based engineering applications such as IEC61499 is the main challenge.

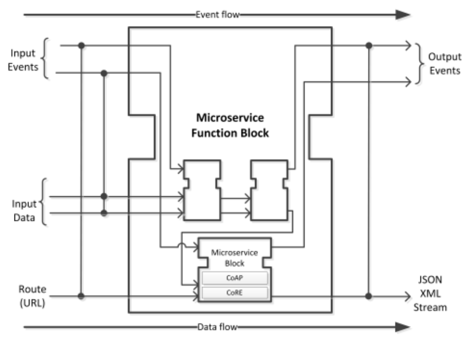

Fig. 1. Microservice Function Block that contains an identifier (URL) as an access route and provides output as JSON or XML.

CONCLUSION

In this paper we demonstrated an existing paradox in service granularity definition of Microservices Architecture. We proposed shifting of granularity to design level (horizontal metric or semantic size) rather than to implementation level (vertical metric or physical size), and we argued for not considering external infrastructure functionalities as part of service granularity. This solves the paradox of microservice granularity and will allow the use of service size as a metric to cover vertical dimension (physical size). Based on our investigation, Microservices Architectures could bring more opportunity to support Plug & Produce and even new assembly systems. In general, it allows to build more interoperable and independent manufacturing system.

Independency by nature eases reconfigurability and makes the entire system independently updateable and deployable. This could be a tremendous shift and improvement for industrial automation system's flexibility.

However, next to the key benefits of using Microservices Architecture such as flexibility, modularity, evolutionary, heterogeneity, certain pitfalls and challenges that need to be tackled also exist, such as statelessness, choreography, granularization, predictability, and synchronization.

REFERENCES

[1] D. P. Adolphs et al., "Reference Architecture Model Industrie 4.0 (RAMI4.0)," vol. 0, no. July, p. 28, 2015.

[2] T. Arai, Y. Aiyama, Y. Maeda, M. Sugi, and J. Ota, "Agile assembly system by `Plug and Produce'," CIRP Ann. - Manuf. Technol., vol. 49, no. 1, pp. 1–4, 2000.

[3] N. Dragoni et al., "Microservices: yesterday, today, and tomorrow," pp. 1–17, 2016.

[4] C. Pautasso, O. Zimmermann, M. Amundsen, J. Lewis, and N. Josuttis, "Microservices in Practice, Part 1: Reality Check and Service Design," IEEE Softw., vol. 34, no. 1, pp. 91–98, 2017.

[5] C. Pautasso, O. Zimmermann, M. Amundsen, J. Lewis, and N. Josuttis, "Microservices in Practice, Part 2: Service Integration and Sustainability," IEEE Softw., vol. 34, no. 2, pp. 97–104, 2017.

[6] Sam Newman, Building Microservices. O'Reilly Media, 2015.

[7] R. V. Shahir Daya Nguyen Van Duy, Kameswara Eati, Carlos M Ferreira, Dejan Glozic, Vasfi Gucer, Manav Gupta, Sunil Joshi, Valerie Lampkin, Marcelo martins, Shishir Narain, "Microservices from Theory to Practice Creating Applications in IBM Bluemix Using the Microservices Approach," Ibm, p. 170, 2015.

[8] M. Melik Merkumians, M. Baierling, and G. Schitter, "A service-oriented domain specific language programming approach for batch processes," 2016 IEEE 21st Int. Conf. Emerg. Technol. Fact. Autom., pp. 1–9, 2016.

[9] Eclipse.org/4diac/, "Open Source PLC Framework for Industrial Automation & Control," 2019. [Online]. Available: http://www.fordiac.org. [Accessed: 06-Jan-2019].

[10] T. Cucinotta et al., "A Real-Time Service-Oriented Architecture for Industrial Automation," IEEE Trans. Ind. Informatics, vol. 5, no. 3, pp. 267–277, 2009.

[11] M. G. Valls et al., "iLAND : An Enhanced Middleware for Real-Time Reconfiguration of Service Oriented Distributed iLAND : An Enhanced Middleware for Real-Time Reconfiguration of Service Oriented Distributed Real-Time Systems," vol. 9, no. FEBRUARY 2013, pp. 228–236, 2013.

[12] H. Bloch, A. Fay, and M. Hoernicke, "Analysis of service-oriented architecture approaches suitable for modular process automation," IEEE Int. Conf. Emerg. Technol. Fact. Autom. ETFA, vol. 2016-Novem, 2016.

[13] A. Girbea, C. Suciu, S. Nechifor, and F. Sisak, "Design and implementation of a service-oriented architecture for the optimization of industrial applications," IEEE Trans. Ind. Informatics, vol. 10, no. 1, pp. 185–196, 2014.

[14] M. Mcgrath, "Service Oriented Manufacturing," 2010.

[15] C. Stoidner and B. Freisleben, "Invoking web services from programmable logic controllers," Proc. 15th IEEE Int. Conf. Emerg. Technol. Fact. Autom. ETFA 2010, pp. 1–5, 2010.

[16] K. H. John and M. Tiegelkamp, "IEC 61131-3: Programming industrial automation systems: Concepts and programming languages, requirements for programming systems, decision-making aids," IEC 61131-3 Program. Ind. Autom. Syst. Concepts Program. Lang. Requir. Program. Syst. Decis. Aids, pp. 1–390, 2010.

[17] K. Thramboulidis, D. C. Vachtsevanou, and A. Solanos, "Cyber-physical microservices: An IoT-based framework for manufacturing systems," Proc. - 2018 IEEE Ind. Cyber-Physical Syst. ICPS 2018, pp. 232–239, 2018.

[18] K. Thramboulidis, D. C. Vachtsevanou, and I. Kontou, "CPuS-IoT : A Cyber-Physical Microservice and IoT-based Framework for Manufacturing Assembly Systems," 2019.

[19] M. Bortolini, E. Ferrari, M. Gamberi, F. Pilati, and M. Faccio, "Assembly system design in the Industry 4.0 era: a general framework," IFAC-PapersOnLine, vol. 50, no. 1, pp. 5700–5705, 2017.

[20] W. Dai, V. Vyatkin, S. Member, J. H. Christensen, and V. N. Dubinin, "Bridging Service-Oriented Architecture and IEC 61499 for Flexibility and Interoperability," vol. 11, no. 3, pp. 771–781, 2015.

[21] C. M. Mackenzie, K. Laskey, F. Mccabe, R. Metz, and A. Hamilton, "Reference Model for Service Oriented Architecture 1.0 Committee Specification 1, 2 August 2006," no. August, pp. 1–31, 2006.

[22] F. Leymann and D. Karastoyanova, "Service Oriented Architecture – Overview of Technologies and Standards," it - Inf. Technol., vol. 50, no. 2, pp. 83–123, 2008.

[23] D. Shadija, M. Rezai, and R. Hill, "Towards an understanding of microservices," ICAC 2017 - 2017 23rd IEEE Int. Conf. Autom. Comput. Addressing Glob. Challenges through Autom. Comput., no. September, pp. 7–8, 2017.

[24] N. M. Josuttis, SOA in Practice, The Art of Distributed System Design, 1st ed. O'Reilly, 2017.

[25] T. Erl, Service-Oriented Architecture: Concepts, Technology & Design. Pearson Education, 2016.

[26] C. H. Gammelgaard, Microservices in .NET Core : with examples in Nancy. 2016.

[27] H. Bloch et al., "A microservice-based architecture approach for the automation of modular process plants," IEEE Int. Conf. Emerg. Technol. Fact. Autom. ETFA, pp. 1–8, 2018.

[28] O. Zimmermann, "Microservices tenets: Agile approach to service development and deployment," Comput. Sci. - Res. Dev., vol. 32, no. 3–4, pp. 301–310, 2017.

[29] P. Di Francesco, I. Malavolta, and P. Lago, "Research on Architecting Microservices: Trends, Focus, and Potential for Industrial Adoption," Proc. - 2017 IEEE Int. Conf. Softw. Archit. ICSA 2017, pp. 21–30, 2017.

[30] B. Butzin, F. Golatowski, and D. Timmermann, "Microservices approach for the internet of things," IEEE Int. Conf. Emerg. Technol. Fact. Autom. ETFA, vol. 2016-Novem, 2016.

[31] H. Paik, A. L. Lemos, M. C. Barukh, B. Benatallah, and A. Natarajan, Web Service Implementation and Composition Techniques. 2017.

[32] O. Multitenancy, "Migrating Enterprise Legacy Source Code to Microservices," pp. 63–72.

[33] N. Kratzke, "About Microservices, Containers and their Underestimated Impact on Network Performance," 2017.

[34] D. Guo, W. Wang, G. Zeng, and Z. Wei, "Microservices architecture based cloudware deployment platform for service computing," Proc. - 2016 IEEE Symp. Serv. Syst. Eng. SOSE 2016, pp. 358–364, 2016.

[35] A. Harrison and I. Taylor, "Dynamic web service deployment using WSPeer," Proc. 13th Annu. Mardi Gras Conf. Grid Appl. Technol., no. October, pp. 11–16, 2005.