Microservices composition. A systematic literature review

Georges Leschener M. Essomba
glme1@leicester.ac.uk

December 2020

Contents

Abstract

1 Introduction

2 Background

Service Oriented Architecture (SOA) and web services emerged in the early 2000s as a new paradigm for designing software applications, and they are presented by many researchers as the foundation of microservices. Baresi & Garriga (2020) refer to microservices as "SOA done right" which supports the theory that microservices are an evolution of web services. Given the close relationship between the two technologies, we believe that it is pertinent as part of this study to provide the reader with an overview of what web services are, some of their composition approaches, as well as the evolutionary snapshot of services from SOA through microservices which are fundamental for understanding the rest of the study.

2.1 SOA and web services

Erl (2004), Huhns and Singh (2005) cited in Jiang et al. (2017) define Service-Oriented Architecture (SOA) as a way of using web service to design large software system that are made of sub-components that are distributed on different remote servers - in a loosely-coupled architecture -which provide services to consumers which can be an end-users using a client component, or another service. The authors describes how a service and a client communicates. A client sends a request over the Internet to a web service running a a remote server using a communication protocol such as Simple Object Access Protocol (SOAP), the incoming request is received by the target web service which reads its information, performs the associated processing which produces an output that is sent back to the client as a response to its request. One of the pros of SOA is its flexibility with regards to the technologies (hardware and programming languages) that can be used to develop and deploy web services.

2.2 Web services composition

According to Angel et al. (2015), service composition encompasses all the processes that create from existing services new value add services, also referred to as composite or aggregated services. The authors explain how web service composition should be tackled from five main areas including i. accessibility: which defines how components making up the composite services can be accessed. In other words how to invoke operations of a web service, send notifications, or to respond to events. ii. Conversation management: which specifies in which order the operations of a service must be enacted thereby ensuring the correct interaction is done with the service. For instance, the specification may require a client

to first authenticate with the service, then to initiate some operations (processing), and then wait for an output event. iii. Control flow: which specifies the order in which the execution composition activities must take place in order to in order to achieve the given objective. For instance, a customer that purchase an item from an ecommerce platform would need to select their items, make a payment before the order confirmation email is sent to her which also triggers the shipment process. iv. Data flow: that specifies the source and destination of data in the form of input and output. Typically, the input of one service is produced by another service as an output. v. Data transformation: which may be required in case of mismatch (e.g. data format, protocol) between two services that communicate as part of service composition activities. Claro et al. (2006) discuss the two main languages namely BPEL4WS and OWL-S that are used to compose web services. The former provides a mechanism for manually specifying composite web services, while the latter provides a machine-readable description of web services make them easily discoverable and composable.

BPEL4WS allows the collaboration of services as activities and processes. Each composite service has an interface which is a collection of WSDL PortTypes. Processes are treated as partners in order to integrate services, and the one that specifies how all the interactions between a process instance and its partners are coordinated is known as business process. Each partner by a partner link and a role name (Claro et al., 2006). Web services composition using semantic web language such as OWL-S augment the automatic discovery and composition. This is done using agents that can automatically find services based on their machine-readable description thereby fulfilling the main motivating task for OWL-S.

2.3 Microservices

Garriga (2018) defines Microservices as a novel architectural style that overcomes the shortcomings of centralized, monolithic architectures, in which application logic is encapsulated in big deployable chunks. In contrast to monolithic applications, microservices are small components, built around business capabilities, that are easy to understand, deploy, and scale independently, even using different technology stacks. Each microservice runs in a dedicated process and communicates through lightweight mechanisms, often a RESTful API. Namiot & Sneps-Sneppe (2014) introduce the microservices approach as a relatively new term in software architecture patterns which consists in developing an application as a set of small independent services with each service running in its own independent process. Five years have passed since the introduction of microservices. Today, the fact of the matter is that the adoption of microservices is growing at a rapid pace as many businesses are looking to move away from legacy architecture design style. Monilith applications which is one such legacy approach as remark Singleton (2016) are known to significantly slowdown application development lifecycle and lead to very risky and costly maintenance and upgrades. With microservices instead, businesses take advantage

of new computing paradigms such as cloud computing, DevOps, Continuous integration, continuous delivery (CI/CD) to name but a few. There are many definitions of microservices that have been given by many researchers and they all seem to convey a common view of the attributes of microservices which include "loosely coupled", "independently developed, deployed and maintained", "using lightweight communication", "small in size". According to Chen (2018) microservices enable teams to produce software reliably in short release cycles, make changes easily and innovate faster. The ease, speed with which developers are able to build microservices based applications do introduce a different kind of challenges. For example the challenge around duplication of effort in trying to build microservices that fulfill similar functions which leads to waste of time and resources. In order to realize the opportunities offered by microservices, some challenges need to be addressed. That is why Chen (2018) contrasts the benefits of microservices with their complexities and challenges, including their discovery and composition. It is still a very complex and time-consuming task to assemble microservices into an application or service that fulfill a given function. A simplification and automation of such task would help unlock the benefits of microservices

3 Related work

In this section we present some of the systematic literature reviews (SLR) that have been done already for studies that focus on service composition. One of the most significant is that of Hayyolalam & Kazem (2018) who conducted a SLR in which they review papers that discuss QoS-aware approaches to service composition in a cloud environment. Customers expect a certain level of service quality from their service provider which is specified in the form of service level agreements (SLAs) that bind the two parties. Those SLAs are made of a number of KPIs which Hayyolalam et al. (2018) refer to as QoS attributes or QoS metrics. In their study, they concentrate specifically on the following attributes: cost, availability, response time, reliability, throughput, execution time and reputation of which cost, availability and response time are the most used by researchers according to their survey with an average of 16% out the 50 papers that were in scope of their studies. Hayyolalam & Pourhaji Kazem (2018) also classify papers on one had based on the service composition environment as well as and whether the study is conducted on a single or multi-cloud, and on the other based on whether is single or multi-objective optimization. Their survey uncover a pattern in which service composition in a cloud environment is described 'as a single-objective problem with local/global QoS optimization or a multi-objective problem with global QoS optimization' (Hayyolalam & Pourhaji Kazem, 2018). One of the limitations of this survey is that it does not focus on how the actual service composition is done nor does it inform the reader on what frameworks are used in the service composition process.

Another notable SLR on cloud service composition is that of Vakili & Nav-

imipour (2017) who view service composition as a new way of integrating multiple individual services in the cloud order to fulfill complex user requirements. Their study focus on identifying the challenges associated service composition in the cloud environment as well as the current approaches for service composition and service selection and the underlying activities that are involved in the on the-fly integration of cloud services as they are provisioned to help address complex end-users requests. Vakili & Navimipour (2017) start by acknowledging the efforts that. Have been done in providing frameworks (platforms and languages) for web service composition which they classify into three main categories which include: workflow-based approaches; XML-based approaches (e.g. BPEL4WS) and ontology-based approaches (e.g. OWL-S and DAML-S). The authors uncovered from their study, three main cloud service composition techniques including the framework-based, heuristic-based and agent-based technique. One of the selected papers for this study is from Zhang et al. (2014) who proposed an example of framework-based technique based on a model generated from a solution algorithm that analyzes the characteristics of services resources in cloud manufacturing.

An example of agent-based service composition technique is proposed by Wang et al., 2016a, Wang et al., 2016b and the method uses a multi-agent reinforcement learning algorithm to interact with the environment in real time in order to generate an optimal composition strategy on the fly. In Karimi et al. (2016), the authors discuss an example of heuristic-based technique for which the service composition is done based on the SLA contract in the cloud environment. The researchers use data mining techniques in service composition and genetic algorithm to ensure a quality service is provided as fast as possible to end-users thereby helping ensure SLA compliance. Vakili & Navimipour (2017) survey shows that efficiency, optimization and time are the QoS metrics that are the most widely used in comparison to scalability and cost.

The fast adoption of cloud computing and the growing number of cloud services has made it impossible for a single service to address the variety of (complex) use cases from multiple cloud services consumers. For that reason, Jula et al. (2014) recognize the need to have a service composition module embedded within a cloud environment in order to bring together a set of atomic services in order to provide a complex functionality. The authors in Jula et al. (2014) consider that a composite service is made of n number of unique services (USs) with p number of QoS parameters. To that end, service composition constitutes a workflow made of a sequence of unique services. Their study identified five categories of service composition approaches which are: classic and graph-based algorithms (CGBAs), combinatorial algorithms (CAs), machine-based approaches (MBAs), structures (STs), and frameworks (FWs) (Jula et al., 2014).

Ye, Zhou, & Bouguettaya (2011) cited in Amin et al. (2014) demonstrate that the combinatorial algorithm for instance uses QoS parameters which is divides into equal, ascending and descending groups and normalise their values by way of additive weighting, which generates a new model that computes the Q0S

of composite service is proposed. Similarly, the machine-based method as described by Amin et al. (2014) selects the path for composite services with highest QoS based on a two-phase process. First it creates a service tree and a target process discarding any improper paths in the tree by applying a policy which also helps reduce processing time. The second phase is about selection the most optimal path which corresponds to QoS parameters that best match the user requirements.

4 Research Methodology

5 Service composition techniques & Frameworks

6 Discussion: results & comparison

7 Research limitations

8 Conclusion & future works

9 Acknowledgements

10 References

1. Vahideh Hayyolalam, Ali Asghar Pourhaji Kazem, A systematic literature review on QoS-aware service composition and selection in cloud environment, Journal of Network and Computer Applications, Volume 110, 2018, Pages 52-74
2. Asrin Vakili, Nima Jafari Navimipour, Comprehensive and systematic review of the service composition mechanisms in the cloud environments, Journal of Network and Computer Applications, Volume 81, 2017, Pages 24-36
3. Amin Jula, Elankovan Sundararajan, Zalinda Othman, Cloud computing service composition: A systematic literature review, Expert Systems with Applications, Volume 41, Issue 8, 2014, Pages 3809-3824
4. Baresi L., Garriga M. (2020) Microservices: The Evolution and Extinction of Web Services?. In: Bucchiarone A. et al. (eds) Microservices. Springer
5. Peishi Jiang, Mostafa Elag, Praveen Kumar, Scott Dale Peckham, Luigi Marini, Liu Rui, A service-oriented architecture for coupling web service models using the Basic Model Interface (BMI), Environmental Modelling & Software, Volume 92, 2017, Pages 107-118
6. Lagares Lemos, Angel & Daniel, Florian & Benatallah, Boualem. (2015). Web Service Composition. ACM Computing Surveys. 48. 1-41. 10.1145/2831270
7. Claro, Daniela & Albers, Patrick & Hao, Jin-Kao. (2006). Web Services Composition.
8. Garriga M. (2018) Towards a Taxonomy of Microservices Architectures. In: Cerone A., Roveri M. (eds) Software Engineering and Formal Methods. SEFM 2017. Lecture Notes in Computer Science, vol 10729. Springer, Cham
9. Dmitry Namiot, Manfred Sneps-Sneppe (2014). On Micro-services Architecture. International Journal of Open Information Technologies ISSN: 2307-8162 vol. 2, no.9, 2014
10. Singleton, A. (2016). The Economics of Microservices. IEEE Cloud Computing,3(5), 16-20
11. Chen, L. (2018). Microservices: Architecting for Continuous Delivery and DevOps. 2018 IEEE International Conference on Software Architecture (ICSA).