MAGMA: Generating Microservice Infrastructure

Philip Wizenty

University of Applied Sciences and Arts Dortmund
Institute for Digital Transformation of Application and Living Domains
44227 Dortmund, Germany
philipnils.wizenty@fh-dortmund.de

1 Introduction

Microservice Architecture (MSA) is an architectural style for the realization of distributed, service-oriented software systems [2]. In comparison to the well established Service-oriented Architectural style (SOA), MSA differs in both conceptual and practical nature [3]. This comprises (i) a lack of implicit support for interaction between services with different message formats and structures; (ii) missing protocol transformation capabilities; (iii) the absence of functionalities of message mediation, routing and transformation capabilities; (iv) lack of implicit support for services coordinations following the orchestration pattern[4].

In SOA, these functionalities are typically provided by an Enterprise Service Bus (ESB), which operates as a comprehensive messaging middleware [4], by conveying between service communication and providing routing, mediation, transformation and coordination capabilities. In contrast, MSA leverages several loosely coupled standalone components, e.g. API Gateways and Discovery Services, to selectively add only those infrastructural components, which are mandatory to the software system.

This paper presents Maven[1] Archetype for Generating Microservice Architectures (MAGMA), a tool that eases the providing of an extensible, yet runnable MSA, which comprises only those infrastructure components mandatory for a certain software system[5]. The MAGMA tool is based on the Maven Archetype mechanism. This enables the tool to be integrated into *continuous Microservice delivery pipelines*. To simplify the usage of MAGMA, it provides although a graphical user interface (GUI) based on JavaFX.

The remaining of this paper is organized as follows. Section 2 provides a brief description of an architectural design for MSA infrastructure. Section 3 presents the usage of the archetype's architecture and the usage of the MAGMA tool. Section 4 covers the conclusion for this paper.

2 Architectural Design for Microservice Architecture

This section defines an architectural design (AD) for MSA infrastructure components which eases the operation of *functional Services* responsible for the realization of business capabilities [4].

An architectural style for MSA has to cope with the following six challenges: (C1) Openness for service integration; (C2) Security of services communication; (C3) Scalability enabled by service independence; (C4) Failure Handling; (C5) Service concurrency; (C6) Transparency concerning interface-based service access [5, 4].

In SOA, where typically an ESB addresses all these challenges as an all-in-one solution [3], MSA deals with each challenge by leveraging separate infrastructure components in an *as-a-*

[1] https://maven.apache.org

service manner. These components enable NSA to rely on *infrastructural design patterns* like API Gateway and Load Balancer to cope with the challenges C1-C6 [1].

An AD, which approaches those challenges is depicted in figure 1 as a UML component diagram. Thereby, each component represents a Microservice. `Discovery Service`, `Security Service` and `API Gateway` are a set of infrastructural services. Specific components i.e. `Load Balancer` and `Circuit Breaker` can be optionally added to each service regardless of its type.

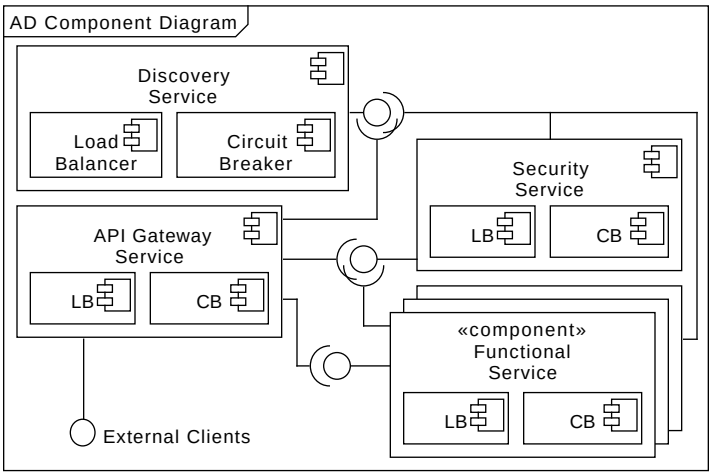

Figure 1: Architectural Design

The `Discovery Service` addresses challenge C1 by providing a mechanism, which exposes the service interfaces to internal and external consumers. Challenge C2 is coped by the `Security Services` by adding functionalities of authentication and authorization. Failure handling and scalability are enabled through the optional components of `Load Balancer` and `Circuit Breaker`(challenge C3, C4). The AD relies on synchronous and RESTful HTTP for service communication, which provides the possibility of service concurrency(challenge C5). The `API Gateway` provides transparency for interface-based service access and addresses challenge C6.

3 The MAGMA Tool

The following section describes MAGMA (figure 2), a tool which adapts the AD presented in section 2. MAGMA generates infrastructural components for MSA which are (i) specially configured for the target application domain; (II)directly runnable; (iii) extensible by functional services leveraging a generated *service template* [2].

3.1 Features

Starting from the AD, MAGMA generates the following fully configured and runnable infrastructure components for message-based MSA systems:

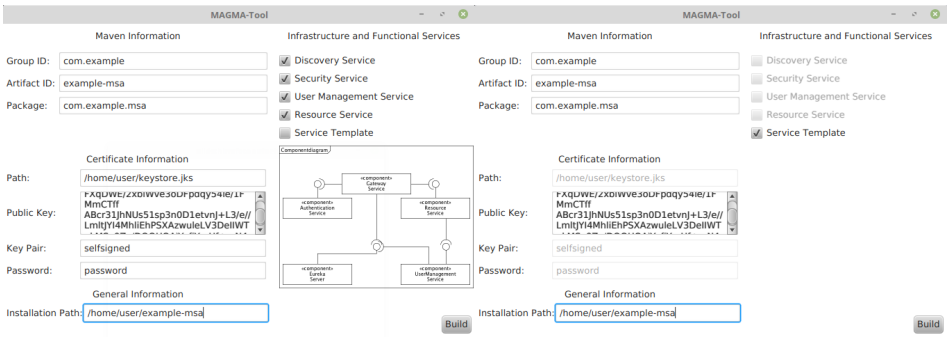

Figure 2: Graphical User Interface of the MAGMA Tool

Security Service. The functionalities of authentication and authorization are provided by the Security Service, which acts as an *identity provider* for securing internal and external communication.

Discovery Service. The Discovery Service provides central means for services to register themselves. There it integrated a naming system that subsumes exposed interfaces for discovery purposes under the exposing service's name.

API Gateway Service. The API Gateway Service realizes a *single sign-on gateway* and acts like as a single point of access to the MSA-based system. In addition, it manages communication between clients and the Security Service.

Resource Service. The Resource Service demonstrates the realization of a functional service that exemplifies the usage of the Service Template. It presents how the different infrastructure components might be used for service communication.

User Management Service. Concerning our AD (cf. Section 2), this service provides a user-based concept of identity handling for authentication and authorization purposes to an MSA-based system.

Service Template. A skeleton implementation of a functional service can be generated by the Usage of the Service Template function of MAGMA. It comprises all boilerplate code and configuration files necessary to enable the services interaction with generated infrastructure components as depicted in Figure 1.

3.2 Usage

MAGMA can be used to set up a MSA-based system that comprises all infrastructure components, or to integrate new services into an existing MSA with the provided service templates.

It is possible to invoke MAGMA from the command line by using the Maven archetype mechanism as shown in Listing 1, but we although provide a more user-friendly graphical user interface (figure2). To support all Java-compatible platforms, the GUI was implemented with JavaFX.

Listing 1: Commandline invocation of MAGMA via Maven.

```
mvn archetype:generate
  -DarchetypeGroupId=Infrastructure
  -DarchetypeArtifactId=MAGMA-archetype
  -DgroupId=com.example
  -DartifactId=example-msa
  -Dpackage=com.example.msa
```

```
-DCertPath=<Location of certificate for OAuth2>
-DCertPublicKey=<Certificate public key>
-DCertKeyPair=<Certificate key pair>
-DCertPassword=<Certificate password>
```

The left side of the GUI (figure 2) enables the user to enter "Maven Information" and "Certificate Information". These information are needed to generate the MSA components. At the right side the user selects the "MSA Components" which are included to the generation. When hitting "Build", Maven is invoked and the selected services and components are generated.

4 Conclusion

In this paper we presented MAGMA, a tool for generating MSA infrastructure components and functional service templates. Thereby, MAGMA relies on an AD, we defined for MSA. We further explained the basic usage of the tool, by providing a short description of the GUI.

MAGMA and its GUI can be found on GitHub[2]. We successfully used both in teaching as well as research projects with industrial partners. We plan to extend the archetype with further infrastructure components, enable a better configuration of basic functional services.

References

[1] Paolo Francesco, Patricia Lago, and Ivano Malavolta. Research on Architecting Microservices: Trends, Focus, and Potential for Industrial Adoption. In *Proc. of the Int. Conf. on Software Architecture (ICSA)*, pages 21–30. IEEE, 2017.

[2] Sam Newmann. *Building Microservices*. O'Reilly Media, 2016.

[3] Florian Rademacher, Sabine Sachweh, and Albert Zndorf. Differences between model-driven development of service-oriented and microservice architecture. *Proc. of the First Int. Workshop on Architecting with MicroServices (AMS) co-located with ICSA*, 2017. To be published.

[4] Mark Richards. *Microservices vs. Service-Oriented Architecture*. O'Reilly Media, 2015.

[5] Philip Wizenty, Jonas Sorgalla, Florian Rademacher, and Sabine Sachweh. Magma: Build management-based generation of microservice infrastructures. *Proc. of the 11th European Conference on Software Architecture*, 2017. To be published.

[2] https://www.github.com/pwizenty/Microservice-BasicInfrastructure