Generation of Container-Based Deployment Units Using an Ecosystem of Microservice-Oriented Modeling Languages

Philip Wizenty and Florian Rademacher

IDiAL Institute, University of Applied Sciences and Arts Dortmund, Germany
philip.wizenty,florian.rademacher@fh-dortmund.de

Introduction. Microservice Architecture (MSA) promotes loose coupling between and high cohesion in services, which is expected to lead to increased scalability and reliability [3]. However, to fully take advantage of those benefits the system needs to be deployed to a suitable environment. Generally, a container-based environment like Kubernetes fosters the benefits of MSA by increasing the scalability through horizontal scaling possibilities and self-healing functions improving the reliability. However, to enable the deployment of MSA, container-based infrastructure technology requires technology-specific configurations with a wide variety of interdependent options. Thus, the complex configuration of container-based technology like Kubernetes can lead to operational failure, often caused by improper configuration [2].

A potential solution to ease the deployment of MSA and to ensure the correctness of its configuration is provided by Model-Driven Engineering (MDE) [1]. MDE enables the development of models that capture software properties and functions at a level of abstraction above source code. Therefore, the systems complexity is reduced with the goal to support the development process by, e.g., facilitating architecture analysis and providing code generation means. [1].

LEMMA[1] (Language Ecosystem for Modeling Microservice Architecture) is an MDE approach towards MSA engineering. It provides modeling languages tailored to MSA-specific architecture viewpoints for specifying domain concepts, service composition and operation configurations [4].

Problem Statement. The configuration of container-based deployments for MSA is a complex and error-prone process [2]. There are a variety of technologies that can be used for the deployment of the system. These not only have individual configuration options, but also differ in the configuration format. In addition, the configuration of the deployment is distributed to several different files, which will be called Deployment Units (DU) in the following. Usually a DU includes service specific configuration files, Dockerfiles and build scripts. Additionally, there are also configurations for service composition and orchestration, such as Docker-Compose or Kubernetes files.

Contribution. LEMMA's Operation Modeling Language (OML) enables developers to specify the deployment of microservices including their dependencies to infrastructural components like a Service Discovery. Listing 1 specifies the deployment of the `OrderService` in LEMMA's OML. The service is part of an MSA-based online shop example. Line 1 imports the `OrderService` defined in a LEMMA service model [4]. Line 2 imports the operation technology description for the service deployment, e.g., `kubernetes`. Lines 3 to 6 specify the deployment of the `OrderService` with an `operationTechnology`, which refers to the technology stack for the deployment. The service is deployed in the `OrderContainer` with an `openjdk` operation environment in a Docker Container defined in Line 5. Additionally, the model includes the deployment of a `DiscoveryService` in Lines 7 to 9. The service is used by the `OrderService`

[1] https://github.com/SeelabFhdo/lemma

to communicate with other services. The LEMMA models and corresponding source code of the example is published on GitHub[2].

Listing 1: Example of an operation model.

```
1 import microservices from "order.services" as OrderService
2 import technology from "operation.technology" as operationTechnology
3 @technology(operationTechnology)
4 container Order deployment technology operationTechnology::_deployment.Kubernetes
5     with operation environment "docker/openjdk:11-jdk-slim"
6     deploys OrderService::v01.de.fhdo.ms2020.OrderService {/* Specific Service Configuration. */}
7 @technology(operationTechnology)
8 DiscoeveryService is operationTechnology::_infrastructure.DiscoeveryService used by services
9     OrderService::v01.de.fhdo.ms2020.OrderService {/* Specific DiscoveryService Configuration.*/}
```

Figure 1 depicts our approach of a code generation pipeline for the generation of container-based DU's by the usage of LEMMA's OML for easing the deployment configuration. After the specification of deployment-relevant configuration options in the operation model (cf. Figure 1). The model is transformed via a refinement *model-to-model transformation* [1] into an intermediate operation model enhanced with additional deployment-relevant details regarding the used technology and deployed services, e.g., the visibility or type of a microservice, technology specific protocol and data formats. Based on the transformed intermediate model, a *model-to-text transformation* [1] is executed for performing the code generation. The presented code generation pipeline currently supports the Docker and Kubernetes technology stacks by creating DU's including the extension of existing configuration files, the creation of Dockerfiles and Build Scripts. Furthermore, for service composition and orchestration docker-compose and kubernetes deployment file are also provided. From the operation model in Listing 1, which comprises 27 Line of Code (LoC) without omissions, a total of 127 LoC could be generated. This shows that a total of 5 lines of source code can be generated from one line of model code. The efficiency of the code generator can be explained by the fact that the configuration of DU's for the Docker and Kubernetes technology stacks includes a span of different files with interdependent options.

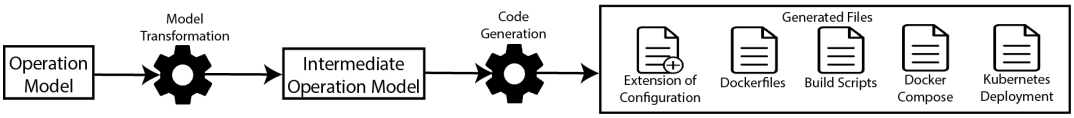

Figure 1: Workflow for generating deployment-related artifacts.

In our talk, we will extend the example of the online shop with additional components and deploy them to a Kubernetes environment by the usage of the generated files.

References

[1] Benoit Combemale et al. *Engineering modeling languages.* Taylor & Francis, CRC Press, 2017.

[2] Hui Kang, Michael Le, and Shu Tao. Container and microservice driven design for cloud infrastructure DevOps. In *2016 IEEE Int. Conf. on Cloud Engineering (IC2E).* IEEE, 2016.

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

[4] Florian Rademacher, Jonas Sorgalla, Philip Wizenty, Sabine Sachweh, and Albert Zündorf. Graphical and textual model-driven microservice development. In *Microservices: Science and Engineering.*, pages 147–179. Springer, dec 2019.

[2] https://github.com/SeelabFhdo/microservices2020