Towards Integrating Blockchains with Microservice Architectures Using Model-Driven Engineering

2nd Agility with Microservices Programming Workshop
(AMP 2021)

Simon Trebbau¹, Philip Wizenty² and Sabine Sachweh³
Smart Environments Engineering Laboratory
University of Applied Sciences and Arts Dortmund

¹simon.trebbau@fh-dortmund.de
²philip.wizenty@fh-dortmund.de
³sabine.sachweh@fh-dortmund.de
PuLS Research Project

- PuLS is an ongoing research project that aims to increase the availability of parking spaces with charging stations for electric vehicles.

- PuLS enables citizens to share their private parking spaces or charging infrastructure with other citizens.

- Development of a Park and Charge Software Platform (PCSP) with the following requirements:
  - High Scalability
  - Modifiability
  - Compability

- Additional research goal: Blockchain integration with the PCSP for decentralized management and processing of charging infrastructure and bookings.
PuLS PCSP

PuLS Microservice Application

Infrastructure Microservices

- API Gateway
- Service Discovery
- Kafka
- KeyCloak
- UI-Service

Legend:
- Database
- Microservice
- Functional

- ParkAndChargeService
- EnvironmentService
- BookingService
- MongoDB
- MariaDB
- EthereumBlockchain
Language Ecosystem for Modeling Microservice Architectures (LEMMA)

- LEMMA is a framework for model-driven development that is currently under development in our research group.

- LEMMA aims to facilitate the design, the development and the deployment of microservice architectures.

- LEMMA provides:
  - Modeling languages for different stakeholders in the Microservice Architecture engineering process
  - Model transformations and model processors, e.g., for code generation and architecture analysis
The Model-Based Approach to Integrate Blockchain into Microservice Architectures

- Domain Model
- Service Model
- Technology Model
- Operation Model

Input

Java Generator / Genlet
- Blockchain Configuration Methods
- Microservice Interfaces
- Data Structures
- Maven Dependencies

Ethereum Generator
- Blockchain Connection Properties

Imports
Ethereum Technology Model - Excerpt

```java
import technology from "./technology/ethereum.technology" as ethereum
@technology(container_base)
@technology(ethereum)
deployment technology container_base::_deployment.Kubernetes with operation
  environment "copenjdk:11-jdk-slim"
deploys bookingService::v01.de.fhdo.puls.BookingService
depends on nodes
  ethereumOperation::Ethereum, eureka::Eureka,
mariadb::MariaDB, keycloakOperation::Keycloak, kafka::Kafka
aspects
  ethereum::_aspects.EthereumNetwork{
    hostName="http://10.0.140.93",
    port=8545,
    gasLimit=8000000,
    gasPrice=200000000000
  }
```

Example

Booking Service Operation Model - Excerpt
Example

**Ethereum Technology Model** - Excerpt

```java
    technology Ethereum {
        service aspects{
            aspect SmartContractLoadFunction for operations {
                string smartContractWrapperName<mandatory>;
            }
        }
        operation aspects {
            aspect EthereumNetwork for containers {
                string hostName<mandatory>;
                int port<mandatory>;
                long gasPrice<mandatory>;
                long gaslimit<mandatory>;
            }
        }
    }
```

---

**Imports**

```java
import datatypes from "booking.data" as bookingData
import technology from ".\technology\spring.technology" as java
import technology from ".\technology\ethereum.technology" as ethereum
```

---

**Microservice and Interface Definition**

```java
@technology(java)
@technology(ethereum)
public functional microservice de.fhdo.puls.BookingService version V01 {
    @endpoints():java:_protocols.rest: "/resources/v1";
    interface ParkAndChargeBooking {
        ...
        @ethereum::_aspects.SmartContractLoadFunction("ParkAndChargeBookingContract")
        public loadParkAndChargeBookingContract();
        ...
    }
    }
```

---

**Method Definition**

**Service Model** of Booking Service – Excerpt
Validation and Future Work

• We were able to validate the basic feasibility of our approach with the PuLS Platform and gather a first impression of how blockchain information can be integrated into microservice models using code generation.

• Generated Artifacts:
  ➢ Java Code specific to Ethereum Blockchain Interaction
  ➢ Microservice Interfaces and POJOs based on Java and Spring
  ➢ Required Maven Dependencies
  ➢ Blockchain Connection Configuration

• Support and derivation of smart contracts using the model-driven engineering.
  ➢ Structural description of smart contracts using LEMMA domain models
  ➢ Analysis, derivation and implementation of smart contract behavior