Towards Integrating Blockchains with Microservice Architectures Using Model-Driven Engineering

2nd Agility with Microservices Programming Workshop

(AMP 2021)

Simon Trebbau[1], Philip Wizenty[2] and Sabine Sachweh[3]

Smart Environments Engineering Laboratory

University of Applied Sciences and Arts Dortmund

[1]simon.trebbau@fh-dortmund.de

[2]philip.wizenty@fh-dortmund.de

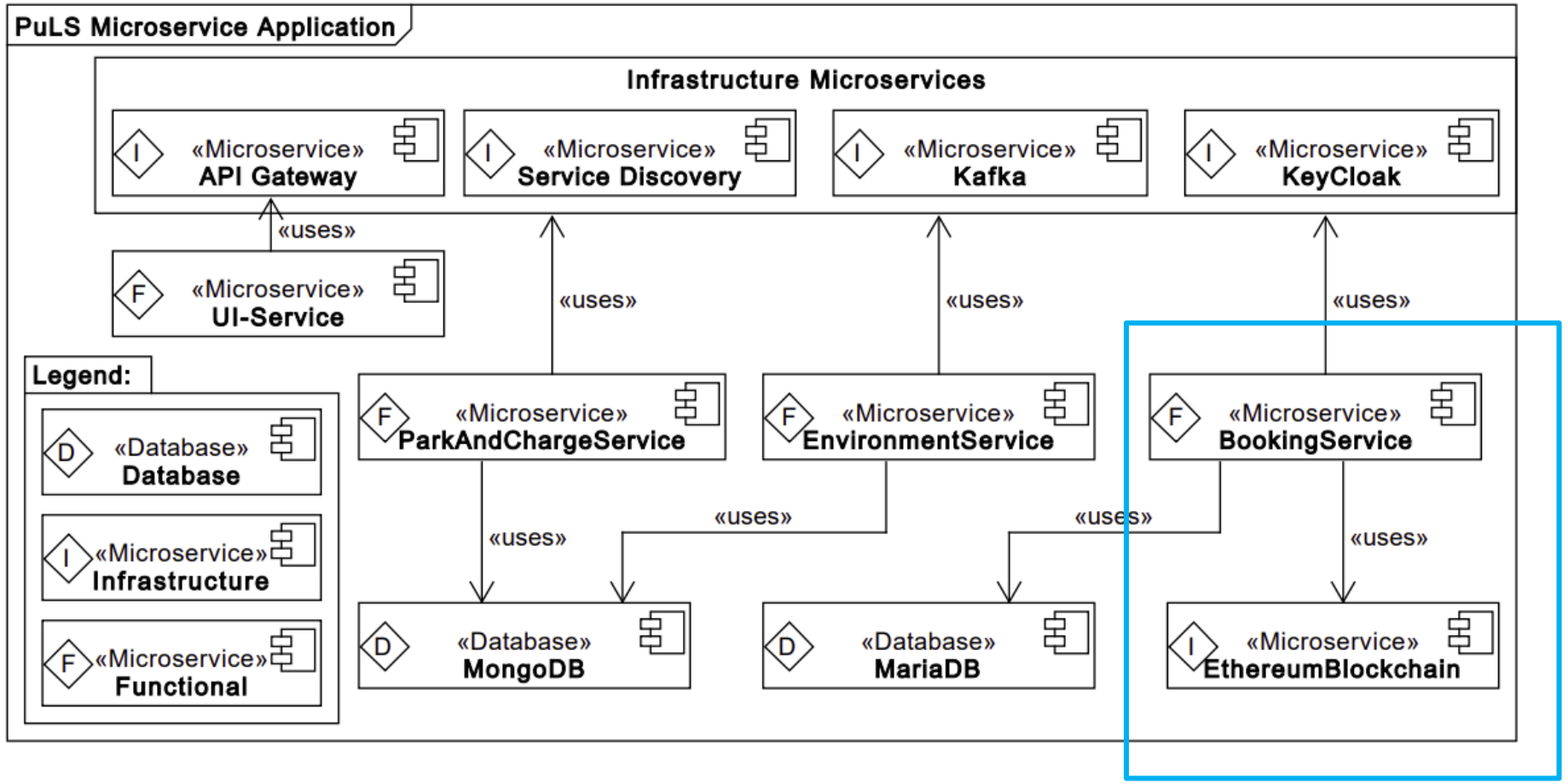
[3]sabine.sachweh@fh-dortmund.de

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**

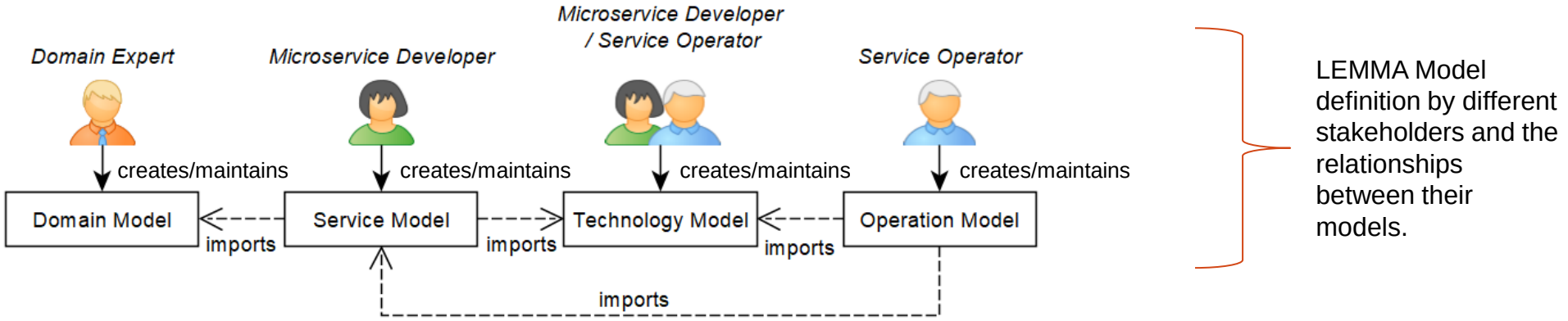
  } Microservice Architecture
- Additional research goal: Blockchain integration with the PCSP for decentralized management and processing of charging infrastructure and bookings.

PuLS PCSP


PuLS Microservice Application
Infrastructure Microservices
«Microservice» API Gateway
«Microservice» Service Discovery
«Microservice» Kafka
«Microservice» KeyCloak
«uses»
«Microservice» UI-Service
«uses»
«uses»
«uses»
Legend:
«Database» Database
«Microservice» Infrastructure
«Microservice» Functional
«Microservice» ParkAndChargeService
«Microservice» EnvironmentService
«Microservice» BookingService
«uses»
«uses»
«uses»
«uses»
«Database» MongoDB
«Database» MariaDB
«Microservice» EthereumBlockchain

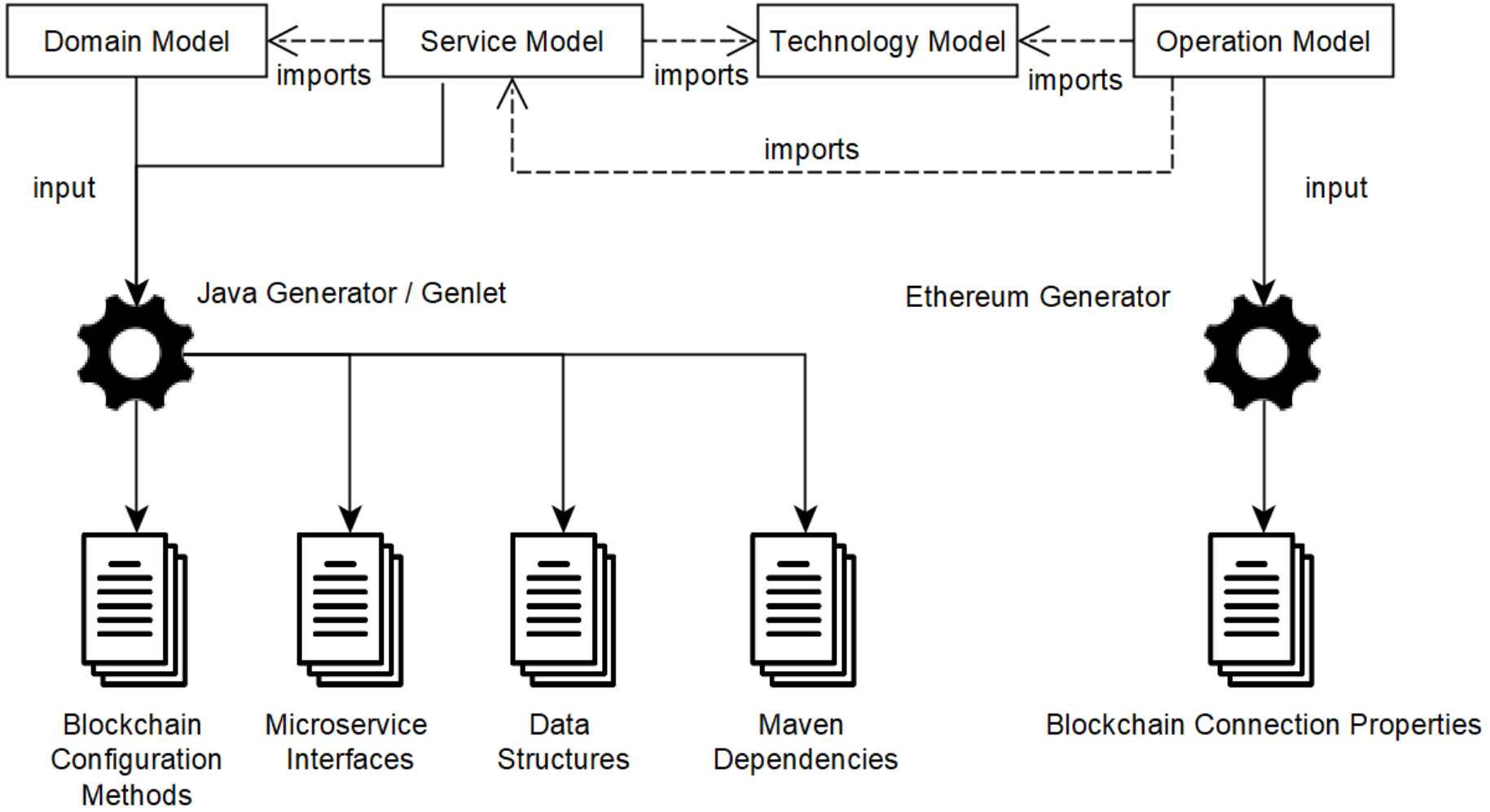
Language Ecosystem for Modeling Microservice Architectures (LEMMA)

- 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



LEMMA Model definition by different stakeholders and the relationships between their models.

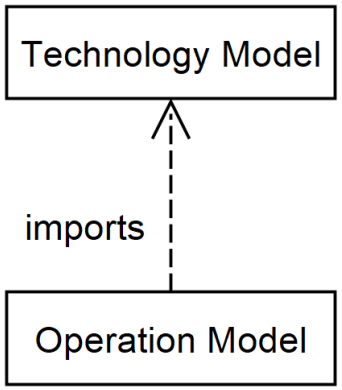
The Model-Based Approach to Integrate Blockchain into Microservice Architectures



Example

Ethereum **Technology Model** - Excerpt

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




```
import technology from "../technology/ethereum.technology" as ethereum
@technology(container_base)
@technology(ethereum)
deployment technology container_base::_deployment.Kubernetes with operation
    environment "openjdk:11-jdk-slim"
deploys bookingService::v01.de.fhdo.puls.BookingService
```

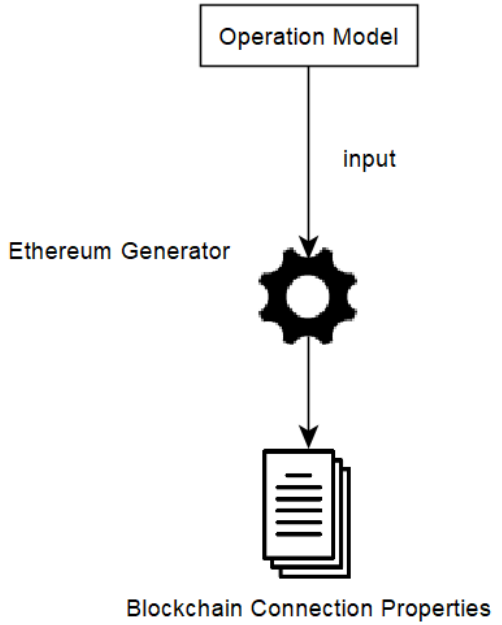
Imports and Deployment Definition for Microservice

```
depends on nodes
  ethereumOperation::Ethereum, eureka::Eureka,
  mariaDB::MariaDB,keycloakOperation::Keycloak, kafka::Kafka
{
```

Service Dependencies

```
  aspects {
    ethereum::_aspects.EthereumNetwork(
    hostName="http://10.0.140.93",
    port=8545,
    gasLimit=8000000,
    gasPrice=20000000000);
}}
```

Aspect Definition

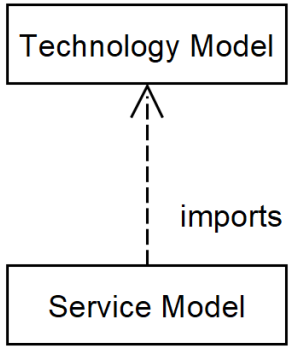


Booking Service **Operation Model** - Excerpt

Example

Ethereum **Technology Model** - Excerpt

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

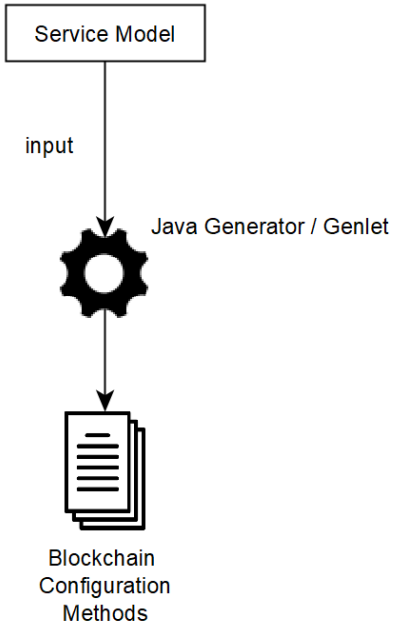



```
import datatypes from "booking.data" as bookingData
import technology from "../technology/spring.technology" as java
import technology from "../technology/ethereum.technology" as ethereum
@technology(java)
@technology(ethereum)
public functional microservice de.fhdo.puls.BookingService version v01 {
  @endpoints(java::_protocols.rest: "/resources/v1";)
  interface ParkAndChargeBooking{
    ...
    @ethereum::_aspects.SmartContractLoadFunction("
        ParAndChargekBookingContract")
    public loadParkAndChargeBookingContract();
    ...
  }
}
```

Imports

Microservice and Interface Definition

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