Towards Integrating Blockchains with Microservice Architecture Using Model-Driven Engineering

Simon Trebbau[1], Philip Wizenty[1][0000−0002−3588−5174], and Sabine Sachweh[1]

IDiAL Institute, University of Applied Sciences and Arts Dortmund,
Otto-Hahn-Straße 27, 44227 Dortmund, Germany
{simon.trebbau,philip.wizenty,sabine.sachweh}@fh-dortmund.de

Abstract Blockchain presents a feasible method to persist immutable information in a distributed ledger to improve the level of authentication and trust. Moreover, smart contracts enable the automated execution of any contract concluded between participants of the Blockchain network. On the other hand, Microservice Architecture (MSA) is a novel approach towards service-based scalable applications. In our paper, we present an approach based on Model-Driven Engineering (MDE) that aims to facilitate the integration process of Blockchains into MSA-based applications in order to benefit from the advantages attributed to Blockchains.

Keywords: Microservice Architecture · Model-Driven Engineering · Code Generation · Distributed Ledger · Blockchain · Smart Contract.

1 Introduction

Blockchain constitutes a technology for information exchange and transactions that require a specific level of authentication and trust [17]. Additionally, a blockchain reduces the risk of data manipulation, system failure, and dependency at a single system component [9]. In recent years, blockchains emerged as an important and influential technology for businesses and society [9].

Moreover, modern blockchain technologies like Ethereum[1] also supports the usage of *smart contracts* [17]. Smart contracts act as autonomous agents in the blockchain network and contain program code that executes by a specific message from a user's transaction or another smart contract. Typical use cases for smart contracts are financial payments or contractual agreements.

Microservices Architecture (MSA), in which services are used as autonomously software building blocks, shares several similarities with the concept of smart smart contracts [16]. Both provide their functionalities over a specific interface. Also, they manage their own data and have an isolated deployment environment, e.g., a Kubernetes[2] pod or an Ethereum Virtual Machine (EVM). MSA as well as smart contracts represent complex distributed systems. Furthermore,

[1] https://ethereum.org/
[2] https://kubernetes.io/

smart contracts can provide similar functionalities similarly to microservices, e.g., individual data storage and API [4]. Therefore, they can be used in a MSA application to increase authentication and trust [17].

This paper introduces an Model-Driven Engineering [2] (MDE)-based approach to ease the integration of blockchain technology into MSA-based applications. By using models as first class citizens in the development process, we abstract from implementation details to reduce the overall complexity. Additionally, the presented approach includes the usage of code generators to support the development process and increase code quality.

The remainder of this paper is structured as follows. Section 2 gives a brief introduction to blockchain technology and MDE of MSA. Section 3 introduces our MDE-based approach for the integration of blockchain technology into MSA. We validate the approach in Section 4. Section 5 provides related work. This paper concludes and outlines future work in Section 6.

2 Background

Blockchain. A blockchain is a shared digital and distributed ledger [10], which stores transaction data on multiple network nodes without a central party and according to an agreed policy. The involved nodes connect directly over a peer-to-peer network. Executed transactions are combined into blocks, which are linked together using cryptographic hash values [8]. For contributing a transaction to the blockchain, it is necessary to build consensus among all participants. The consensus is formed via a *consensus algorithm* [17] that synchronizes the distributed ledger on the different participant nodes.

Generally, a smart contract is an automated transaction protocol, which executes the terms of a contract [17]. Therefore, a smart contract represents a fragment of source code that could be executed automatically on a dedicated environment on the blockchain and perform various functionalities, e.g., a financial transaction or report at the end of an electric vehicle charging process.

Model-Driven Engineering of Microservice Architecture. A well researched approach to enable MDE for MSA is the Language Ecosystem for Modeling Microservice Architecture[3] (LEMMA) [11]. LEMMA provides a set of modeling languages and model transformations that are built using the Eclipse Modeling Framework (EMF) [15]. LEMMA utilizes methods and techniques from MDE to reduce the complexity of MSA engineering for various stakeholder groups, e.g., domain experts and microservice developers. The modeling languages provide the possibility to create models as an artifact in the software engineering process for enabling code generation and reasoning about microservice architectures.

LEMMA's Domain Data Modeling Languages (DDML) [13] enables the modeling of domain concepts and addresses the *domain viewpoint* on microservice

[3] `https://github.com/SeelabFhdo/lemma`

architectures. DDML is used by domain experts and service developers to capture domain concepts in models and supports Domain-Driven Design (DDD) [5] patterns such as Entity, Aggregate, and Repository.

The Technology Modeling Language (TML) [13] provides a means for service operators and developers to construct models targeting technology-specific information for service implementation and operation. Therefore, TML allows capturing and modularization of information regarding programming languages, frameworks, and deployment technologies. Additionally, *technology aspects* [11] are supported as a concept for the integration of technology-specific metadata, e.g., database mappings or microservice interaction configuration, into the models.

LEMMA's Service Modeling Language (SML) [13] focuses on the *service viewpoint* on MSA. SML enables service developers to construct models for specifying the API of a microservice. In detail, the SML allows to specify interfaces including their data structure as well as interface dependencies on other microservices. To this end, SML models are able to import previously defined DDML models as well as other SML models.

The Operation Modeling Language (OML) [13] of LEMMA addresses the *operation viewpoint* on MSA and is used by service operators. OML encapsulates concepts for service deployment, e.g., deployment technologies, operation environments, service-specific configurations, and dependencies to infrastructure components.

In addition to the modeling languages, LEMMA also provides means to processing resulting models. Firstly, LEMMA contains *intermediate metamodels* and *intermediate model transformations* [7] to facilitate the processing of the constructed models. Based on these intermediate models, LEMMA includes a Model Processing Framework to ease the development of model processors like code generators, model analyzers, and model visualization.

3 A Model-Based Approach to Integrate Blockchain with Microservice Architecture

Our approach focuses on the research context of supporting the integration process of blockchain technology for MSA. For this purpose, we use MDE to abstract from implementation details to reduce the overall complexity by using microservice architecture viewpoint-specific modeling languages. Precisely, our approach provides the functionalities to realize the integration process of blockchain technology utilizing LEMMA's modeling languages.

With a view to MSA and blockchains as well as their potentially combination, some challenges arise. A specific challenge in MSA is the deployment and operation of the microservices [1], which also applies to blockchain because of their similarities. Furthermore, the handling of smart contracts also can be a challenging process because they need to be integrated into the application and blockchain [3]. Based on these challenges, the question is how can MDE be used in suitable places to abstract frequently occurring and possibly complex processes.

LEMMA-Based Integration of Blockchain Functionality into Microservice Architecture. Our approach uses LEMMA's modeling languages (cf. Sect. 2) to abstract from implementation details in MSA development to support the development process. Moreover, we use code generators to generate multiple artifacts, e.g., Java classes and configuration files. The presented approach focuses on the generation of blockchain-related artifacts like implementing the connection to the blockchain network. Figure 1 depicts our approach and the relation between the different LEMMA models, the code generators, and generated artifacts.

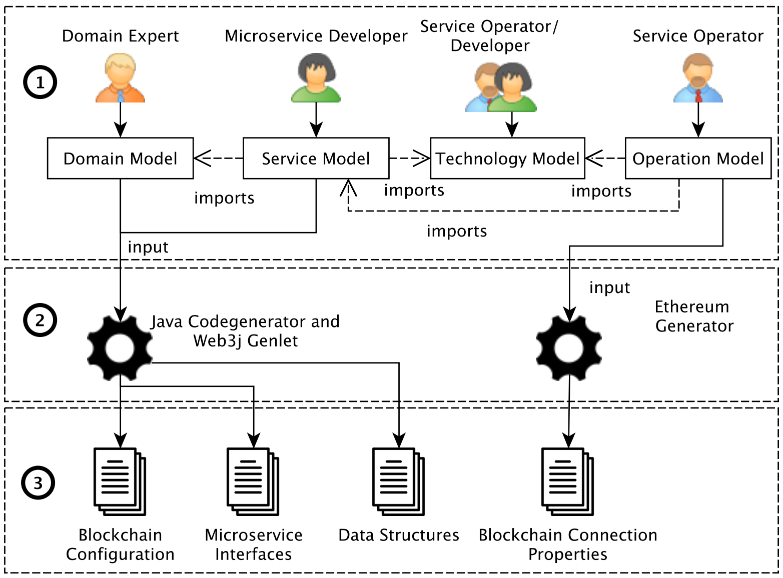

Figure 1. LEMMA-based approach for integration of blockchain functionality into MSA.

The presented approach divides into three consecutive stages. Stage 1 comprises an agile modeling process by the stakeholder groups of MSA engineering. They collaborate to construct the models, which describe the MSA [11]. This approach considers all groups and their models, but with an increased focus on the models dealing with blockchain aspects of the application. It includes the `Domain Model` for data structures, the `Service Model` for configuration aspects or API definitions, the `Operation Model` for blockchain connectivity property initialization, and a `Technology Model` that defines Ethereum specific aspects like network properties.

Stage 2 utilizes the created models from Stage 1 as an input for the model processing. The `Java Base Generator` and `Web3j Genlet` use the `Domain` and `Service Model` to generate Java source code. *Genlets* are code generation modules introduced by LEMMA to generate individual source code for passed domain model and service model [12]. Additionally, the `Ethereum Generator` utilizes the `Operation Model` to derive blockchain service configuration properties for establishing the connection to a blockchain. Both the `Web3j Genlet` and the `Ethereum Generator` represent defined extensions of the LEMMA framework, which also could be extended by additional generation functions in the further if necessary.

Stage 3 shows the code generators created artifacts for the development process of the MSA application. The `Blockchain Configuration` configures communication with the blockchain network using the Web3j[4] library. It is generated as a separate artifact by the `Web3j Genlet`. It includes predefined java methods for connection establishment, transaction management, and adjustable methods for deployment and loading of smart contracts. Additionally, the `Microservice Interfaces` and `Data Structures` are generated based on the `Domain Model` and `Service model` to support the development process. The `Microservice Interfaces` can be used to trigger a smart contract by using the methods provided via the blockchain configuration artifact. Moreover, to enable the deployment process of the microservice in association with blockchain, the `Ethereum Generator` creates service-specific `Blockchain Connection Properties` for connecting the microservice to the blockchain.

4 Validation

This section validates the presented LEMMA-based approach for the integration of blockchain for MSA. For this purpose, we first introduce a case study as a basis for validation, followed by the results of our approach.

Case Study. To validate our approach, we introduce the PuLS[5] Park and Charge platform as a case study. The platform is being developed using LEMMA and a model-first approach and aims, among other things, to demonstrate the feasibility of our approach to ease the integration of blockchain functionality and MSA technologies in the context of MDE. The PuLS Park and Charge platform itself is being developed as part of an ongoing research project that aims to increase the availability of charging infrastructure for electric cars in inner-city areas. For this purpose, the platform allows citizens to share their private charging infrastructure with others. The architecture of the platform, which provides the sharing functionalities, is depicted in Figure 2.

[4] `https://github.com/web3j`

[5] `https://parken-und-laden.de/`

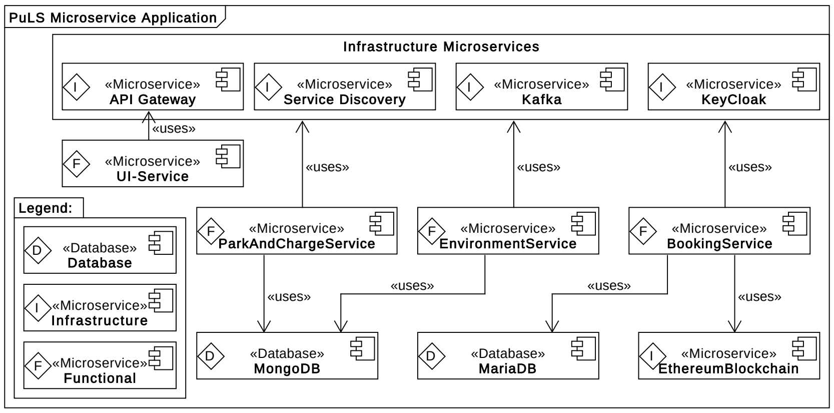
PuLS Microservice Application
Infrastructure Microservices
«Microservice» API Gateway
«Microservice» Service Discovery
«Microservice» Kafka
«Microservice» KeyCloak
«uses»
«Microservice» UI-Service
«Microservice» ParkAndChargeService
«Microservice» EnvironmentService
«Microservice» BookingService
«Database»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

Figure 2. Microservice Architecture of the PuLS case study application.

The PuLS platform consists of three functional microservices. The `ParkAndChargeService` is responsible for sharing and managing the charging infrastructure for the citizen. To monitor the city's environmental data, the `Environment Service` collects data from IoT devices to keep track of particulate matter pollution. Bookings of the charging stations are realized via the `BookingService`. For ensuring the integrity of the bookings, the information is persisted in a blockchain. Storing the bookings increases citizens' trust, as the information can no longer be changed in the blockchain, ensuring that the process of using the charging stations is secured. The intention to incorporate blockchain at this point builds on the interest of PuLS project partners, who have proposed a corresponding integration in this context. Accordingly, a research context in the PuLS project is to find out how and for which possible use cases blockchain technology can be integrated into the Park and Charge platform. Our use case illustrates a corresponding opportunity.

To provide a better understanding regarding the possible modeling of blockchain-specific aspects using LEMMA, Listing 1.1 presents an excerpt of the booking operation model used by the `Ethereum Generator` to create the `Blockchain Connection Properties` for microservice deployment. Lines 1 to 6 describe the general deployment of the `BookingService`. The blockchain-specific configuration is defined via a `technology aspect` (cf. Sect. 2) in Lines 7 to 15. Depending on the use case, the technology model referenced here can be extended by any other specific aspects that also can be used in context of service and operation models. The shown `EthereumNetwork` aspect initializes a `hostName` and `port` for communicating with an Ethereum network node. Additionally, a `gasLimit` and `gasPrice`, which are necessary for transaction management are defined in the model. Furthermore, the `privateKey` associates an attribute for accessing an Ethereum wallet available on the addressed network node to execute or receive transactions.

```
...
@technology(container_base)
@technology(ethereum)
container BookingContainer
  deployment technology container_base::_deployment.Kubernetes
    deploys bookingService::v01.de.fhdo.BookingService
        depends on nodes ethereumOperation::Ethereum {
      aspects {
        ethereum::_aspects.EthereumNetwork(
          privateKey="...",
          hostName="http://localhost",
          port=8545,
          gasLimit=4712388,
          gasPrice=20000000000
        ); }}
```

Listing 1.1. Excerpt of `BookingService` Operation Model.

Results. For our approach we provide a model representation of the PuLS architecture using LEMMA's modeling languages. Based on the resulting models, the `Ethereum Generator` and `Web3j Genlet` (cf. Figure 1) create the code artifacts needed for the integration of basic blockchain functionality. In this context, the `Web3j Genlet` and the generation of the `Blockchain Configuration` demonstrate, that it is possible to support the integration of blockchain technology for MSA using MDE. Moreover, utilizing the `Ethereum Generator`, it was possible to abstract the deployment of microservices in association with blockchain. We use a basic lines-of-code metric to gain a first estimate of the efficiency of our approach by comparing the manually created models with the generated artifacts. This shows that the number of lines of generated code is higher than the number of lines of code needed to define the models. However, it should be noted that the metric refers to Java code. This may well produce different results for other programming languages. For replicability purposes, all artifacts for the approach are provided in a GitHub repository[6]. In addition to the LoC comparison, current research is working on other comparisons that provide a better idea of the generators efficiency. For the presented case study, the integration of the blockchain for MSA works successfully and is used in the presented PuLS research project in the development process.

5 Related Work

De Sousa *et al.* [14] presents an approach to constructing a prototype based on microservices and blockchain technology. It enables the integration and interaction between notary offices and other institutions, ensuring security and high speed in exchanging information between parties. As a case study, microservice architecture has been developed in which external institutions such as a hospital and post office and a notary's office interact with an Ethereum blockchain to manage a birth registration. In contrast to our approach, the integration of MSA and blockchain is related to a specific scenario and spares the usage of MDE.

[6] `https://github.com/SeelabFhdo/xp2021`

Gorski and Bednarski [6] propose modeling support from the perspective of deploying distributed ledger solutions. Their approach uses MDE for transforming distributed ledger models into source code, e.g., deployment scripts or deployment configuration. Like in our approach, MDE is used to facilitate the development process of a distributed ledger. However, our approach addresses the integration of blockchain for MSA and focuses on blockchain as distributed ledger solutions.

6 Conclusion and Future Work

This paper has shown by means of a concrete example that it is feasible to integrate blockchain technology into MSA to increase authentication and trust (cf. Sect. 1) by using MDE. To this end, Sect. 2 introduced background information about blockchain and MDE of MSA. To support the integration of blockchain in MSA, we presented our MDE-based approach in Sect. 3. This approach utilizes LEMMA to generate Service-specific blockchain configurations. We validate the approach through a case study in Sect. 4. Additionally, we provide a brief overview of related work regarding MSA and blockchain (cf. Sect. 5).

For future research, we plan to diverge smart contracts program code from LEMMA's domain models. Additionally, we want to extend the existing code generators to provide better support for different blockchain technologies. Moreover, the code generator should also create blockchain deployment-related artifacts to enable the deployment process of blockchain components and its various network nodes. Another topic we will address relates to the compatibility of MDE with verification technologies. Also, a challenge which we are going to address in the future is the abstraction and modeling of relationships between microservices, the various blockchain network nodes and their user wallets.

References

1. Alshuqayran, N., Ali, N., Evans, R.: A systematic mapping study in microservice architecture. In: 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA). pp. 44–51. IEEE (2016)
2. Benoit Combemale, e.a.: Engineering modeling languages. Taylor & Francis, CRC Press (2017)
3. Dannen, C.: Solidity programming. In: Introducing Ethereum and Solidity, pp. 69–88. Springer (2017)
4. Esposito, C., Castiglione, A., Choo, K.K.R.: Challenges in delivering software in the cloud as microservices. IEEE Cloud Computing **3**(5), 10–14 (2016)
5. Evans, E.: Domain-Driven Design Reference. Dog Ear Publishing, first edn. (2015)
6. Gorski, T., Bednarski, J.: Applying model-driven engineering to distributed ledger deployment. IEEE Access **8**, 118245–118261 (2020). https://doi.org/10.1109/access.2020.3005519
7. Jézéquel, J.M., Combemale, B., Derrien, S., Guy, C., Rajopadhye, S.: Bridging the chasm between mde and the world of compilation. Software & Systems Modeling **11**(4), 581–597 (2012)

8. Malik, S., Dedeoglu, V., Kanhere, S.S., Jurdak, R.: TrustChain: Trust management in blockchain and IoT supported supply chains. In: 2019 IEEE International Conference on Blockchain (Blockchain). IEEE (jul 2019). https://doi.org/10.1109/blockchain.2019.00032
9. Ølnes, S., Ubacht, J., Janssen, M.: Blockchain in government: Benefits and implications of distributed ledger technology for information sharing (2017)
10. Quiniou, M.: Blockchain : the advent of disintermediation. ISTE, Ltd. John Wiley & Sons, Inc, London, UK Hoboken, NJ (2019)
11. Rademacher, F., Sachweh, S., Zündorf, A.: Aspect-oriented modeling of technology heterogeneity in microservice architecture. In: 2019 IEEE International Conference on Software Architecture (ICSA). pp. 21–30. IEEE (2019)
12. Rademacher, F., Sachweh, S., Zundorf, A.: Deriving microservice code from underspecified domain models using DevOps-enabled modeling languages and model transformations. In: 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). IEEE (aug 2020). https://doi.org/10.1109/seaa51224.2020.00047
13. Rademacher, F., Sorgalla, J., Wizenty, P., Sachweh, S., Zündorf, A.: Graphical and textual model-driven microservice development. In: Microservices, pp. 147–179. Springer (2020)
14. de Sousa, P.S., Nogueira, N.P., dos Santos, R.C., Maia, P.H.M., de Souza, J.T.: Building a prototype based on microservices and blockchain technologies for notary's office: An academic experience report (mar 2020). https://doi.org/DOI:10.1109/ICSA-C50368.2020.00031
15. Steinberg, D., Budinsky, F., Paternostro, M., Merks, E.: EMF: Eclipse Modeling Framework. Addison-Wesley, second edn. (2008)
16. Tonelli, R., Lunesu, M.I., Pinna, A., Taibi, D., Marchesi, M.: Implementing a microservices system with blockchain smart contracts. In: 2019 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE). pp. 22–31. IEEE (2019)
17. Zheng, Z., Xie, S., Dai, H.N., Chen, X., Wang, H.: Blockchain challenges and opportunities: A survey. International Journal of Web and Grid Services **14**(4), 352–375 (2018)