Towards Viewpoint-Based Microservice Architecture Reconstruction

Philip Wizenty and Florian Rademacher

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

1 Introduction

Microservice Architecture (MSA) is a comparatively novel approach to the realization of service-based software architectures [16]. It promotes to increase the independence of a service by (i) letting it realize a distinct, self-contained capability; (ii) decreasing its coupling to other software components w.r.t., e.g., implementation, testing, and operation; and (iii) transferring its ownership to a dedicated team, being responsible for all aspects related to service design, implementation, and operation [16].

The resulting *microservices* and their *service-specific independence* have the potential to make software systems more scalable and reliable [16]. In addition, MSA is expected to greatly benefit maintainability by facilitating the replacement of services with improved versions [6]. However, this flexibility may also lead to a proliferation of microservices and a subsequent erosion of the anticipated architecture design, e.g., when teams autonomously advance different parts of a software system [4]. Consequently, the investigation of Software Architecture Reconstruction (SAR) [3] of existing systems is considered an important area in MSA research [6].

Therefore, we propose to present our design, implementation, and preliminary results of an MSA-enabled SAR approach. Specifically, our approach aims to automate as much tasks as possible to provide an efficient means to reconstruct microservice architectures. Furthermore, our approach reifies reconstruction results as architecture models which are expressed in modeling languages dedicated to reducing the complexity in MSA engineering on the basis of Model-driven Engineering (MDE) [5]. By contrast to related approaches [1, 11], the models resulting from our SAR approach respect different viewpoints and concerns in MSA engineering, thereby allowing stakeholder-focused examination of knowledge gathered from SAR processes [3]. As a result, domain experts can perform a targeted review of reconstructed domain concepts independently of, e.g., reconstructed microservice APIs which are in the responsibility of technology-savvy MSA stakeholders like microservice developers. The preliminary results from applying our SAR approach on a case study show that our approach reconstructs the microservices and their corresponding interfaces and operations. Furthermore, the data structures associated with those interfaces or data persistence functionalities are also recovered in the process.

The remainder of the paper is organized as follows. Section 2 provides background information about MDE, and LEMMA[1] (Language Ecosystem for Modeling Microservice Architecture), a concrete MDE approach. Section 3 describes our approach to reconstruct models from source code using LEMMA. We validate the preliminary results of our approach in the following Section 4 and provide them in an Github repository[2]. The paper concludes with a conclusion and potential future work in Section 5.

[1] https://github.com/SeelabFhdo/lemma
[2] https://github.com/SeelabFhdo/microservices2022

2 Viewpoint-Based MSA Modeling with LEMMA

This section provides insight in the MDE process of MSA using LEMMA. MDE [5] is an approach to software engineering that aims to facilitate the design, implementation, and execution of software systems through the use of *models*. A model in the sense of MDE is an artifact that (i) abstracts from selected characteristics of the considered software system to benefit its realization in a certain way; (ii) is expressed in a dedicated *modeling language* which prescribes well-formedness constraints and semantics for valid models; and (ii) is (semi-) automatically processible for specific purposes in the software engineering process.

MDE is particularly helpful in the engineering of complex software systems as it allows the description of such systems or parts thereof from model-based *viewpoints* [12, 8]. Model-based viewpoints can, for example, foster domain experts' understanding of a software system by gathering domain-specific information in dedicated domain models that are expressed in a modeling language which uses familiar terms from the application domain [7].

By contrast to source code, viewpoint models are specifically effective in making the parts and underlying concepts of complex software architectures explicit to facilitate reasoning about them [24]. In addition, MDE provides means to systematically process models, e.g., to generate code from them, which can significantly benefit implementation efficiency [2, 15], or analyze them for automated quality assessment.

LEMMA is an MDE-based ecosystem that focuses on the concerns of stakeholders in MSA engineering [20, 18]. Specifically, LEMMA aims to support stakeholders in coping with challenges inherent to MSA engineering—among them domain-driven service design [4] and, API management [10].

To this end, LEMMA integrates four stakeholder-oriented modeling languages:

- **Domain Data Modeling Language (DDML):** The DDML focuses the concerns of domain experts and microservice developers in collaborative domain modeling. For this purpose, the language provides linguistic support for Domain-Driven Design [7] as a popular design methodology for MSA engineering [16, 14, 9].
- **Technology Modeling Language (TML):** The TML enables microservice developers and operators to capture information related to microservice implementation and operation, e.g., programming languages, deployment specifications, or configurations for orchestration platforms. Moreover, it allows aspect-based specification of metadata [18].
- **Service Modeling Language (SML):** Microservice developers can use the SML to model microservice APIs and endpoints. For this purpose, service models can import DDML-based domain models to use domain concepts as types for microservice operations and thus determine a microservice's domain responsibility [16]. Moreover, the SML allows the import of technology models to, e.g., assign protocol information to modeled endpoints, or configure microservice APIs as expected by frameworks like Spring[3].
- **Operation Modeling Language (OML):** The OML defines modeling concepts for microservice operators to express microservices' deployment and use of operation infrastructure. The language provides a generic syntax which aims to be applicable in the configuration of heterogeneous MSA operation technologies [16].

LEMMA's modeling languages and model processing framework [10] have already been validated in real-world use cases [19, 22, 21]. By contrast to related approaches [13, 23],

[3]`https://www.spring.io`

LEMMA does not assume a certain technology for microservice implementation, e.g., Java [23]. LEMMAs modeling languages and their integration based on model imports support multi-concern modeling. That is they do not focus on only a single concern in MSA engineering such as domain modeling, or the implementation and provisioning of APIs [13, 23].

3 LEMMA-Enabled Approach for MSA Reconstruction

This section introduces a four-phased process for recovering a software system's architecture and our concrete approach to it utilizing LEMMA. The semi-automated reconstruction is a complex and tool-dependent process [3] that generally requires a set of different tools to support the source code artifacts of the development process, e.g., programming languages and deployment specifications. Therefore, the reconstruction requires a set of tools to recover the software system's architecture. Additionally, the reconstruction environment or workbench should provide the possibility for the integration of new tools without unnecessarily modifying existing data [3].

The SAR reconstruction process includes four phases with specific activities and results.

1. *Raw View Extraction* is the first phase of the SAR process with a strong focus on obtaining information about the software system's architecture, mainly from source code artifacts, deployment specifications, and service interactions [11, 3].

2. *Database Construction* phase consists of transforming the reconstructed architecture information into a standardized data format [3].

3. *View Fusion and Manipulation* phase combines the various view information stored in the database to improve the accuracy of the reconstructed information. The manipulation part in this phase aggregates and interprets the reconstructed combined information to create a *hypotheses* on the software system's architecture [3].

4. *Architecture Analysis* is the concluding phase of the SAR process and relates the analysis of the hypothesis about the software system's architecture from the previous stage. The hypothesis needs to be analyzed and tested to prove its correctness. [3].

Figure 1 depicts the different components of our software reconstruction workbench for restoring the architecture of a technology heterogeneous software system using LEMMA.

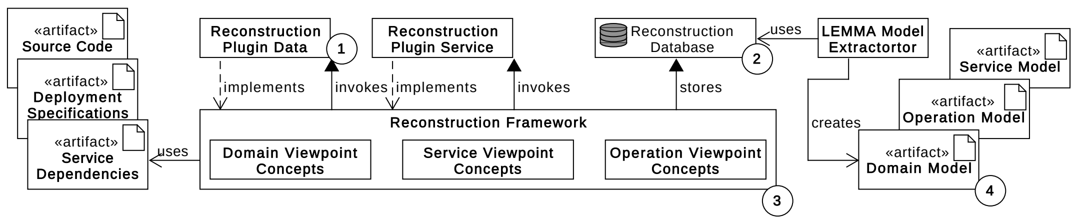

Figure 1: Structure of the RF and RP for the reconstruction of a software system's architecture.

The `Reconstruction Frameworks` (RF) orchestrates the first three phases of the SAR process. Therefore, the framework is responsible for the raw view extraction and the fusion and manipulation of the reconstructed information by using the standardized data formats from the `Reconstruction Database`. The first phase of the SAR restores architecture information from e.g., `Source Code` and `Deployment Specifications`. For this purpose, the RF loads the development artifacts and invokes the `Reconstruction Plugins` (RP). The RP implements viewpoint-dependent functionalities for the technology-specific reconstruction of architecture information. When the RF invokes the RP, they perform the raw view extraction and forward the reconstructed architecture information to the framework.

The second phase consists of constructing the database to store the recovered information about the software system's architecture. Therefore, we derived a standardized data format for each viewpoint in MSA (cf. Section 2). The data format consists of the specific concepts for each viewpoint to store the reconstructed architecture information.

In the third phase of the SAR process, the RF performs a fusion and manipulation task based on the raw view information recovered from the RP to increase the accuracy of the reconstructed architecture. For this purpose, the RF transforms the raw view information into hypotheses about the software system's architecture and `stores` them into the `Reconstruction Database` in the standardized viewpoint-specific data format.

The fourth and last phase of the SAR process consists of the architectural analysis of the recovered architecture. Our approach addresses this phase in two sequential steps. To provide the Software Architect with a readable representation of the reconstructed software architecture, the `LEMMA Model Extractor` [17] uses the information in the `Reconstruction Database` to derive LEMMA models from it. Particularly `Domain Data`, `Service`, and `Operation Models` utilizing the eponymous modeling languages. In the final step of our approach, the Software Architect analyzes the hypothesis of the software system's architecture from the LEMMA models to prove or disprove them to provide an accurate representation of the software system's architecture. To ease the manual analysis of the reconstructed software architecture, LEMMA provides a set of functionalities to support this task, e.g., static analyzers[4].

4 Preliminary Results

This section contains the evaluation of the preliminary results of the reconstruction for the Lakeside Mutual[5] case study. The case study consists of five functional, three infrastructural, and four frontend microservices. At the current point of development, the RF and RP's support the recovery of functional microservices using the Java and Spring technology stack. Table 1 illustrates the preliminary results for the SAR Process for the RF and RP's using LEMMA's modeling languages for the Domain Data and Service viewpoint in MSA.

Table 1: Preliminary reconstruction results from the Reconstruction Framework and Plugins.

Element	**Expected**	**Reconstructed**	**Precision**
Microservices	5	4	80
Interfaces	16	14	87,5
Operations	61	50	82
Parameters	161	117	72,7

The RF and RP's support the reconstruction of four out of five microservices. The frame-

[4]`https://github.com/SeelabFhdo/lemma/tree/main/de.fhdo.lemma.analyzer`
[5]`https://github.com/Microservice-API-Patterns/LakesideMutual`

work was unable to recover one of the microservices because the service uses NodeJS as a programming language, which is not supported by our approach at the moment but could be seamlessly integrated with a RP to our reconstruction workbench. The RF recovers all interfaces and operations from the rest of the microservices. The reconstructed parameters refer to the operation's in- and outgoing data types. The discrepancy for the parameters results from the fact that the plugins do not support the reconstruction of external dependencies, where the source code is not present for the reconstruction, e.g., Spring dependencies. Additionally, the FR and RP's also recover complex data types of the parameters as data structures with LEMMA's domain models.

5 Conclusion and Future Work

In this contribution, we summarized our MDE-based approach for MSA reconstruction by utilizing LEMMA as a presentation proposal to Microservices 2022. Our proposal depicts an early stage of development, where some functionalities are not fully implemented, e.g., the recovery of deployment specification. Nevertheless, the preliminary results of architecture reconstitution show promising results with a recovery rate of approximately 80 percent.

In conclusion, we introduced viewpoint-based MSA modeling with LEMMA in Section 2 to provide the background knowledge for our approach. In the following Section 3, we describe the reconstruction using the RF and RP's to derive architecture information from source artifacts and LEMMA to present the recovered information to the Software Architect. Additionally, we evaluate our preliminary results in Section 4.

In the future, we plan to extend our reconstruction approach to increase the accuracy of the reconstructed architecture. Therefore, we plan to integrate runtime information to derive service interaction, include deployment specifications to address additional viewpoints in MSA, and include infrastructural components to consider dependencies to databases or service discoveries to prove a better representation of the overall architecture.

References

[1] Nuha Alshuqayran, Nour Ali, and Roger Evans. Towards micro service architecture recovery: an empirical study. In *2018 IEEE International Conference on Software Architecture (ICSA)*, pages 47–56. IEEE, 2018.

[2] Paul Baker, Shiou Loh, and Frank Weil. Model-Driven Engineering in a large industrial context — Motorola case study. In *Model Driven Engineering Languages and Systems*, pages 476–491. Springer, 2005.

[3] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice.* Addison-Wesley, third edition, 2013.

[4] Justus Bogner et al. Microservices in industry: insights into technologies, characteristics, and software quality. In *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*, pages 187–195. IEEE, 2019.

[5] Benoit Combemale et al. *Engineering Modeling Languages: Turning Domain Knowledge into Tools.* CRC Press, 2017.

[6] Paolo Di Francesco, Ivano Malavolta, and Patricia Lago. Research on architecting microservices: trends, focus, and potential for industrial adoption. In *2017 IEEE International Conference on Software Architecture (ICSA)*, pages 21–30. IEEE, 2017.

[7] Eric Evans. *Domain-Driven Design*. Addison-Wesley, 2004.

[8] Robert France and Bernhard Rumpe. Model-driven development of complex software: a research roadmap. In *2007 Future of Software Engineering*, pages 37–54. IEEE, 2007.

[9] Martin Garriga. Towards a taxonomy of microservices architectures. In *Software Engineering and Formal Methods*, pages 203–218, Cham. Springer, 2018.

[10] Saverio Giallorenzo et al. Model-Driven Generation of Microservice Interfaces: From LEMMA Domain Models to Jolie APIs, 2022. arXiv: `2202.11397 [cs.SE]`.

[11] Giona Granchelli et al. MicroART: a software architecture recovery tool for maintaining microservice-based systems. In *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*, pages 298–302. IEEE, 2017.

[12] ISO/IEC/IEEE. Systems and software engineering — Architecture description. Standard ISO/IEC/IEEE 42010:2011(E), 2011.

[13] Stefan Kapferer and Olaf Zimmermann. Domain-driven service design. In *Service-Oriented Computing*, pages 189–208. Springer, 2020.

[14] Genc Mazlami, Jürgen Cito, and Philipp Leitner. Extraction of microservices from monolithic software architectures. In *2017 IEEE International Conference on Web Services (ICWS)*, pages 524–531. IEEE, June 2017.

[15] Parastoo Mohagheghi and Vegard Dehlen. Where is the proof? - A review of experiences from applying MDE in industry. In *Model Driven Architecture – Foundations and Applications*, pages 432–443. Springer, 2008.

[16] Sam Newman. *Building Microservices: Designing Fine-Grained Systems*. O'Reilly, 2015.

[17] Florian Rademacher, Sabine Sachweh, and Albert Zündorf. A modeling method for systematic architecture reconstruction of microservice-based software systems. In *Enterprise, Business-Process and Information Systems Modeling*, pages 311–326. Springer, 2020.

[18] Florian Rademacher, Sabine Sachweh, and Albert Zündorf. Aspect-oriented modeling of technology heterogeneity in Microservice Architecture. In *2019 IEEE International Conference on Software Architecture (ICSA)*, pages 21–30. IEEE, 2019.

[19] Florian Rademacher, Sabine Sachweh, and Albert Zündorf. 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)*, pages 229–236. IEEE, 2020.

[20] Florian Rademacher et al. Graphical and textual model-driven microservice development. In *Microservices: Science and Engineering*, pages 147–179. Springer, 2020.

[21] Florian Rademacher et al. Towards holistic modeling of microservice architectures using LEMMA. In *Companion Proceedings of the 15th European Conference on Software Architecture*, pages 11–20. CEUR-WS, `http://ceur-ws.org/Vol-2978/mde4sa-paper2.pdf`.

[22] Jonas Sorgalla et al. Applying Model-Driven Engineering to stimulate the adoption of DevOps processes in small and medium-sized development organizations. *SN Computer Science*, 2(6):459, 2021.

[23] Branko Terzić et al. Development and evaluation of MicroBuilder: a model-driven tool for the specification of REST microservice software architectures. *Enterprise Information Systems*, 12(8-9):1034–1057, 2018. Taylor & Francis.

[24] Jon Whittle, John Hutchinson, and Mark Rouncefield. The state of practice in Model-Driven Engineering. *IEEE Software*, 31(3):79–85, May 2014. IEEE.