Monolith Application to Microservices Model Driven Analysis Migration: State-of-The-Art Techniques

Muhammad Imran Mohamad Sasudin*[1], Hazlifah Mohd Rusli[2] and Nazri Kama[3]

[1,2,3]Advanced Informatics Department, Razak Faculty of Technology and Informatics, Universiti Teknologi Malaysia
[1]imran98@graduate.utm.my, [2]hazlifah@utm.my, [3]mdnazri@utm.my

Article history

Received:
18 Sept 2022

Received in revised form:
29 Nov 2022

Accepted:
1 Dec 2022

Published online:
15 Dec 2022

*Corresponding author
imran98@graduate.utm.my

Abstract

Microservices architecture has become enormously popular as traditional monolithic architectures no longer meet the needs of scalability and rapid development cycle. Furthermore, the success of large companies in building and deploying services is a strong motivation for others to consider making the change. However, performing the migration process is not trivial. Most systems acquire too many dependencies between their modules and thus cannot be sensibly broken apart. For this reason, studies that provide information associated with the migration process to practitioners are necessary. Existing migration techniques are categorized into three main approaches: static analysis, dynamic analysis, and model-driven analysis. This paper focuses on the model-driven analysis approach. A literature search was conducted using search strings to discover recent migration approaches based on model-driven analysis. The migration steps were extracted and identified for each proposed model-driven analysis technique. Based on identified migration steps from each proposed model-driven analysis technique, a migration model is generated by combining all steps from all techniques and simplifying it with three incremental versions of the simplification model. By understanding the differences and similarities between the approaches, the strength and weaknesses of each technique can be identified.

Keywords: *microservices architecture, monolith, software migration, model-driven analysis*

1. Introduction

Microservices Architecture (MSA) is a cloud-native architectural style inspired by Service-Oriented Architecture (SOA). It consists of small, autonomous services that communicate and work together. MSA allows organizations to deliver software faster, respond quicker to change, and embrace newer technologies. Some of the world's most innovative and lucrative businesses, such as Amazon, Netflix, Uber, and Etsy, owe their massive success in IT efforts to microservices. Migrating a monolithic system into a microservice is a long and challenging process that requires much effort from the stakeholders. Proper decomposition of monolith into microservices with appropriate granularity can be seen as the main challenge in architectural migration. There exist several approaches for extracting and identifying candidate microservices, such as model-driven analysis [1]–[5], static analysis [6]–[8], and dynamic analysis [9]–[11].

** Corresponding author. imran98@graduate.utm.my*

Migrating a monolithic system into a microservice is a long and difficult process that requires much effort from the stakeholders. Proper decomposition of the monolith into microservices with the appropriate granularity can be seen as the main challenge in architectural migration. Despite all those proposed approaches, there is still a lack of standard metrics for evaluating these techniques.

Most migration efforts fail due to poor planning, frequently due to cost miscalculation (typically a significant understatement) [12]. Finding the proper service cut and developing the requisite skills with new technologies is a huge technical issue. Most businesses take a non-systematic or personalized approach to consider service cuts [13]. Migrating from a monolithic architecture to a microservices ecosystem is a long and winding road [14]–[17]. Hence, if the migration plan is done correctly, the tendency for the migration to fail can be reduced. To properly plan a software migration, the cost is an essential element to consider[12]. However, there is no best way to migrate from monolith to microservice application in microservice migration. There is only successful migration and good migration practice [14].

This paper discusses the differences and similarities in terms of the general steps in service identification and extraction between state-of-the-art migration techniques based on model-driven analysis. A comparison table on state-of-the-art migration techniques is produced.

The organization of the paper is as follows. Section 2 discusses similar review articles on state-of-the-art techniques for migrating from monolith to microservices. Section 3 discusses the methodology involved in gathering state-of-the-art migration techniques. Next, the state-of-the-art migration techniques will be discussed in Section 4, and finally, Section 5 will conclude the article.

2. Related Work

Several research papers have described reviews on state-of-the-art techniques for microservices migration.

Ponce, Marquez, and Astudillo (2019) gathered, organized, and analysed 20 monolith-to-microservice migration techniques. This paper identifies 20 migration approaches from all three categories, some of which are hybrid. This paper also identifies which category the migration approach belongs to. However, this paper mainly describes general information related to identified migration approaches, such as the category, proposed programming language, and database migration. There is no analysis on the cost or time of any identified migration approaches.

Kazanavicius and Mazeika (2019) elaborated on the benefits and drawbacks of 6 of the migration approaches from monolith architecture to microservice architecture. However, this paper only elaborated on the technical benefits and drawbacks without considering other important factors in software development, such as cost and time.

3. Methodology

The methodology used in this research was a literature search from several databases such as Web of Science, ScienceDirect, and Scopus. The articles were selected based on search strings "monolith to microservice migration techniques". The articles were selected regardless of their category and domain. Some articles were also discovered and selected based on state-of-the-art techniques described in related work or references.

There are two domains regarding migration techniques from monolith to microservice architecture: infrastructure and application. The approaches were categorised based on their domain and category by reading and understanding the selected articles. Three categories and two domains are identified in this process. The three categories are static analysis, dynamic analysis, and model-driven analysis. Based on the identified approaches, we select only approaches with model-driven analysis categories and application domains. As a result, ten migration approaches were chosen to be used in this study and will be described in section 4.

4. State-Of-The-Art Model-Driven Analysis Migration Techniques

Several state-of-the-art model-driven analysis techniques have been identified in our study. Table 1 shows the list of state-of-the-art migration techniques for model-driven analysis. The table shows the author, year, and migration steps involved in each approach.

Table 1. State-of-The-Art Migration Techniques

Number	Author and Year	Migration Steps
1	Kuryazov, Jabborov, and Khujamuratov (2020)	1. Identify how tightly coupled the parts within the system are. 2. Extract metadata and business logic of the system. 3. Correct, improve, optimize or reimplement any extracted microservices 4. Orchestrate applicable services based on certain data and control flow to fulfill the software system's business logic.
2	Amiri (2018)	1. Define two relations regarding their structural and data dependencies for each pair of activities in a process model. 2. Define the final relation between activities by aggregating the relations 3. Use the relation to cluster activities and identify microservices.
3	Sayara, Towhid, and Hossain (2018)	1. Identify the broad business capabilities. 2. Break down business capabilities into sub-business capabilities

		3. Find and determine the independent update probability of the services. 4. Construct update matrix. 5. Modify the updated matrix to get the scaling matrix. 6. Apply the Multidimensional Scaling Technique (MDS) to group similarly weighted element. 7. Identify technology conflict (if any). Service that needs to use different technology needs to be a standalone service.
4	Li, Ma, and Lu (2020)	1. Analyze the monolith and divide it into modules. 2. Create a service registry and an API gateway. 3. Select a service based on ratings to replace the module implementation. 4. Develop the service. 5. Design integration glue. 6. Remove the target legacy module.
5	Chen, Li, and Li (2018)	1. Engineers, together with users, conduct business requirement analysis and construct a purified while detailed dataflow diagram of the business logic. 2. An algorithm combines the same operations with the same type of output data into a virtual abstract dataflow. 3. The algorithm extracts individual modules of "operation and its output data" from the virtual abstract dataflow to represent the identified microservice candidates
6	Tyszberowicz, Heinrich, Liu, and Liu (2018)	1. Analyze the use case specifications (write out their detailed descriptions if needed). 2. Identify the system operations and state variables (based on the use cases and their scenarios). 3. Create an operation/relation table. 4. Advise a possible decomposition into highly cohesive and low coupled components (using a visualization tool). 5. Identify the microservices APIs. 6. Identify the microservices databases. 7. Implement the microservices (using RESTful protocols to communicate between microservices). 8. Deploy.
7	Fan and Ma (2017)	1. Analyse the internal system architecture using Domain-Driven Design (DDD).

		2. Determine whether the database schema is consistent with the candidate microservices and the filtering out of inappropriate candidates. 3. Extract and organize code related to the service candidates using Java. 4. Treat the Java interface as a temporary service interface to ease communications between services. 5. After service code extraction, the operator selects a communications protocol, data format, and microservice framework. 6. Finally, the Java interfaces are transformed into actual service interfaces, such as REST or MQTT, after which service invocations are implemented to enable communication between the various services.
8	Ahmadvan and Ibrahim (2016)	1. Identify security requirements in a system. 2. Initialize the model with a list of functional requirements and security policies. 3. Determine the level of scalability required for each functional requirement. 4. Balance scalability and security. 5. Extract microservice based on the reconciliation result.
9	Michael Gysel, Lukas Kölbener, Wolfgang Giersche, and Olaf Zimmermann (2016)	1. Decompose input. 2. Decompose process. 3. Integrate algorithm. 4. Prioritize scoring.
10	Levcovitz, Terra, and Valente (2016)	1. Map the database tables into subsystems. 2. Create a dependency graph. 3. Identify parts. 4. For each subsystem, select pairs identified in the previous step. 5. Identify candidates to be transformed on microservices. 6. Create API gateways.

Some of the papers have similarities in terms of the migration steps. For example, 9 out of 10 articles contain an analysis step for the first step. The differences between the analysis process between the techniques are the artifacts and items that are analysed, which are the database schema, system security, use case specifications, domain-driven design architecture, and coupling between the systems.

Amiri (2018) began service extraction without analysis in the first step. The technique assumes the existence of an analysis diagram (BPMN diagram) to extract microservice candidates. The paper is also the only one that evaluates its own approach's accuracy by comparing it with different microservice identification approaches. The evaluation initiative gives more confidence and better validation to ensure the approach is on the same track as the other approaches. Based on each analysis result, each approach moves to the next step, extraction, using the analysis result using its extraction formula.

A monolith-to-microservices migration model is produced based on the analysis of all ten state-of-the approaches. The following section describes how the migration model is derived.

5. Derivation of the Migration Model

To derive the migration model, the first step is to generalize each technique's migration steps. Table 2 lists the selected migration techniques for model-driven analysis. The table shows the author, year, and general steps involved in each approach. The steps are generalised from the steps listed in Table 1 in the previous section and consist of *Analysis, Extract, Conflict Identification, Refactor, Develop, Integrate, Evaluate* and *Deploy*.

Table 2. State-of-The-Art Model-Based Migration Techniques

Number	Author and Year	Findings (General step)
1	Kuryazov, Jabborov, and Khujamuratov (2020)	*Analysis, Extract, Refactor, Integrate*
2	Amiri (2018)	*Extract, Evaluate*
3	Sayara, Towhid, and Hossain (2018)	*Analysis, Extract, Conflict Identification*
4	Li, Ma, and Lu (2020)	*Analysis, Extract, Develop*
5	Chen, Li, and Li (2018)	*Analysis, Extract*
6	Tyszberowicz, Heinrich, Liu, and Liu (2018)	*Analysis, Extract, Develop, Deploy*
7	Fan and Ma (2017)	*Analysis, Extract, Develop*
8	Ahmadvand, and Ibrahim (2016)	*Analysis, Extract*
9	Michael Gysel, Lukas Kölbener, Wolfgang Giersche, and Olaf Zimmermann (2016)	*Analysis, Extract*
10	Levcovitz, Terra, and Valente (2016)	*Analysis, Extract, Develop*

The objective of generalizing the steps is to simplify all the steps into more standardized terms in software development. The first version of the model is constructed by unifying all of these steps. Figure 1 shows the first version of the state-of-the-art model.

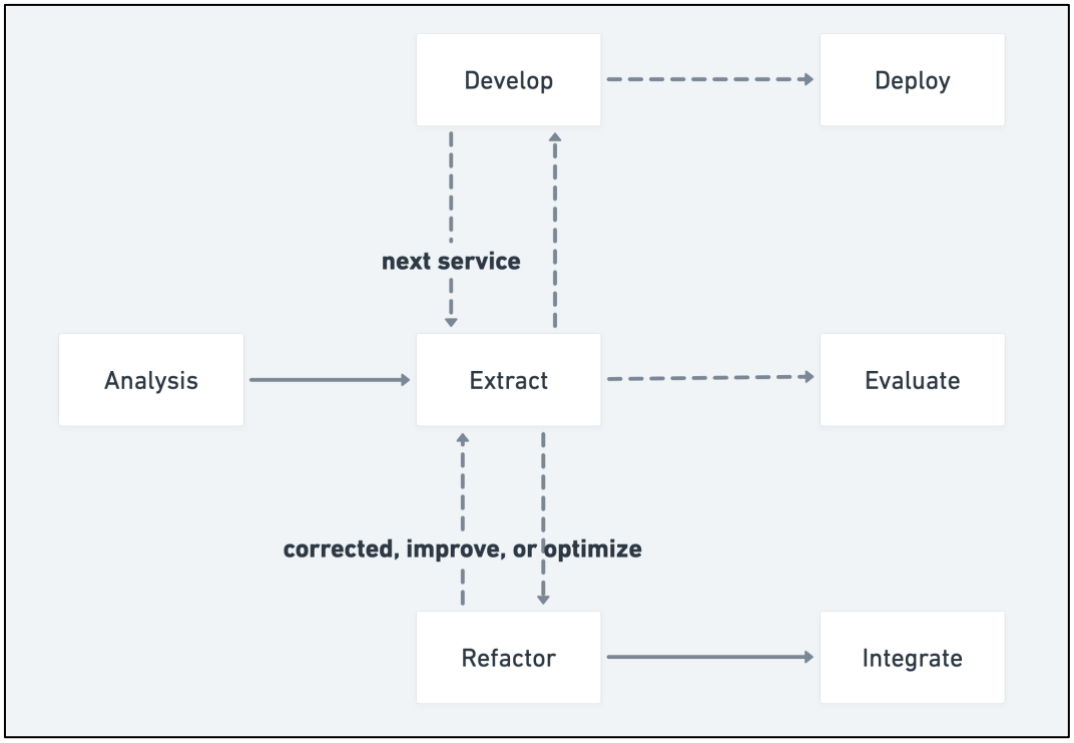

Figure 1. First Version of the Migration Model

Based on the first version of the migration model, four components which are *Refactor*, *Integrate*, *Develop* and *Deploy* are grouped under *Develop* component as the processes are related to the development. Figure 2 shows the second version of the migration model.

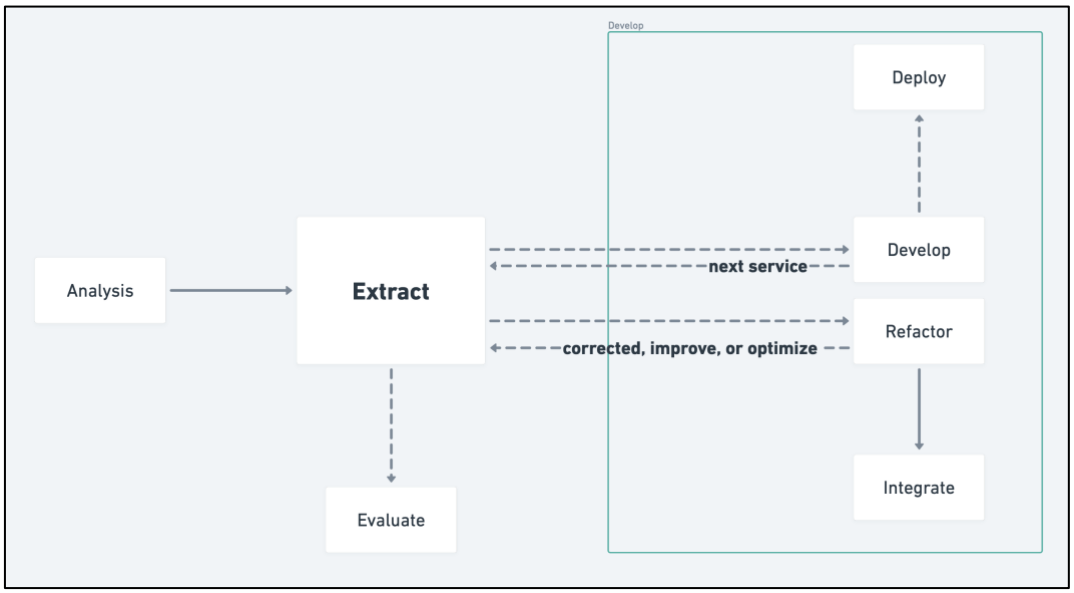

Figure 2. Second Version of the Migration Model

The model is further simplified by eliminating the 4 components and putting them under the *Develop* component. Figure 3 shows the third version of the migration model.

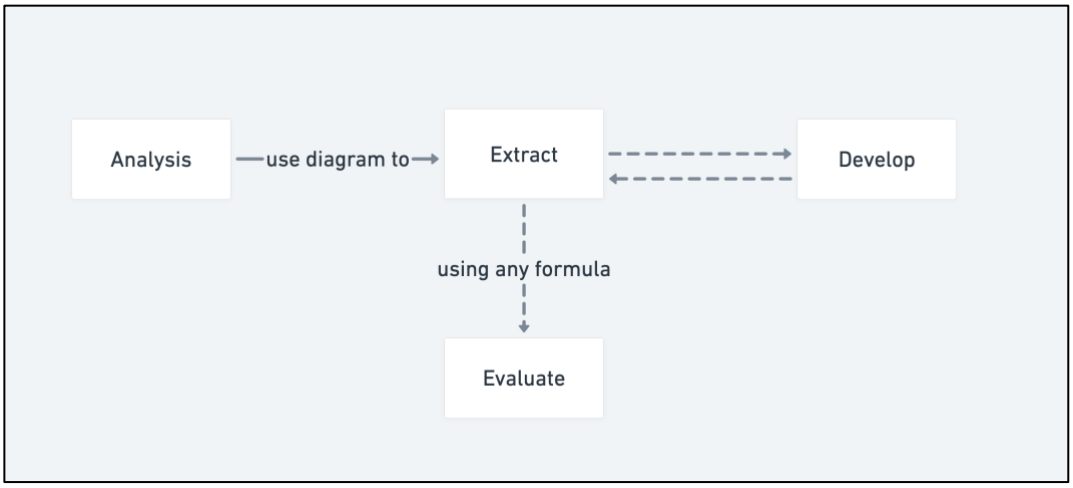

Figure 3. Third Version of The State-of-The-Art Migration Model

The third version of the model is the finalized summary of the migration model. This model indicates the general processes for all ten state-of-the-art approaches for model-driven analysis migration techniques.

6. Conclusion

In conclusion, a literature search has identified several state-of-the-art migration techniques in this study. Different migration techniques require different steps, and each step has its guidelines to be executed. A monolith-to-microservices migration model based on model-driven analysis was derived from the identified state-of-the-art approaches. The purpose of deriving the model is to help understand the differences and similarities between the approaches, which will support future work to identify the approaches' strengths and weaknesses. The strength and weaknesses can also be determined by experiencing those approaches when migrating monolith to microservice applications. These findings will then be used further to solve the main research problem.

Acknowledgement

The study is financially supported by the Encouragement Research Grant Scheme (Vote No. Q.K130000.3856.20J92) awarded by University Teknologi Malaysia.

References

[1] C. Y. Fan and S. P. Ma, 'Migrating Monolithic Mobile Application to Microservice Architecture: An Experiment Report', *Proceedings - 2017 IEEE 6th International Conference on AI and Mobile Services, AIMS 2017*, pp. 109–112, 2017, doi: 10.1109/AIMS.2017.23.

[2] A. Sayara, M. S. Towhid, and M. S. Hossain, 'A probabilistic approach for obtaining an optimized number of services using weighted matrix and multidimensional scaling Md. Shamim Towhid', in *20th International Conference of Computer and Information Technology, ICCIT 2017*, 2018, vol. 2018-Janua, pp. 1–6. doi: 10.1109/ICCITECHN.2017.8281804.

[3] R. Chen, S. Li, and Z. Li, 'From Monolith to Microservices: A Dataflow-Driven Approach', *Proceedings - Asia-Pacific Software Engineering Conference, APSEC*, vol. 2017-Decem, pp. 466–475, 2018, doi: 10.1109/APSEC.2017.53.

[4] M. J. Amiri, 'Object-Aware Identification of Microservices', in *2018 IEEE International Conference on Services Computing (SCC)*, 2018, pp. 253–256. doi: 10.1109/SCC.2018.00042.

[5] C. Y. Li, S. P. Ma, and T. W. Lu, 'Microservice Migration Using Strangler Fig Pattern: A Case Study on the Green Button System', *Proceedings - 2020 International Computer Symposium, ICS 2020*, pp. 519–524, 2020, doi: 10.1109/ICS51289.2020.00107.

[6] S. Eski and F. Buzluca, 'An automatic extraction approach - Transition to microservices architecture from monolithic application', *ACM International Conference Proceeding Series*, vol. Part F1477. 2018. doi: 10.1145/3234152.3234195.

[7] H. Knoche and W. Hasselbring, 'Using Microservices for Legacy Software Modernization', *IEEE Softw*, vol. 35, no. 3, pp. 44–49, 2018, doi: 10.1109/MS.2018.2141035.

[8] A. Krause, C. Zirkelbach, W. Hasselbring, S. Lenga, and D. Kroger, 'Microservice Decomposition via Static and Dynamic Analysis of the Monolith', *Proceedings - 2020 IEEE International Conference on Software Architecture Companion, ICSA-C 2020*, pp. 9–16, 2020, doi: 10.1109/ICSA-C50368.2020.00011.

[9] D. Taibi and K. Systä, 'From Monolithic Systems to Microservices: A Decomposition Framework based on Process Mining', in *Proceedings of the 9th International Conference on Cloud Computing and Services Science - CLOSER*, 2019, pp. 153–164. doi: 10.5220/0007755901530164.

[10] W. Jin, T. Liu, Q. Zheng, D. Cui, and Y. Cai, 'Functionality-Oriented Microservice Extraction Based on Execution Trace Clustering', in *Proceedings - 2018 IEEE International Conference on Web Services, ICWS 2018 - Part of the 2018 IEEE World Congress on Services*, 2018, pp. 211–218. doi: 10.1109/ICWS.2018.00034.

[11] W. Jin, T. Liu, Y. Cai, R. Kazman, R. Mo, and Q. Zheng, 'Service Candidate Identification from Monolithic Systems Based on Execution Traces', *IEEE Transactions on Software Engineering*, vol. 47, no. 5, pp. 987–1007, 2021, doi: 10.1109/TSE.2019.2910531.

[12] H. M. Sneed and C. Verhoef, 'Cost-driven software migration: An experience report', *Journal of Software: Evolution and Process*, vol. 32, no. 7, Jul. 2020, doi: 10.1002/smr.2236.

[13] J. Fritzsch, J. Bogner, S. Wagner, and A. Zimmermann, 'Microservices Migration in Industry: Intentions, Strategies, and Challenges', in *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Sep. 2019, pp. 481–490. doi: 10.1109/ICSME.2019.00081.

[14] J. Kazanavicius and D. Mazeika, 'Migrating Legacy Software to Microservices Architecture', *2019 Open Conference of Electrical, Electronic and Information Sciences, eStream 2019 - Proceedings*, 2019, doi: 10.1109/eStream.2019.8732170.

[15] V. Velepucha and P. Flores, 'Monoliths to microservices - Migration Problems and Challenges: A SMS', in *2021 Second International Conference on Information Systems and Software Technologies (ICI2ST)*, Mar. 2021, pp. 135–142. doi: 10.1109/ICI2ST51859.2021.00027.

[16] J. Carlos Ribeiro Dias Neves, 'Technical Challenges of Microservices Migration', *Instituto Politecnico do Porto (Portugal) ProQuest Dissertations Publishing*, Oct. 2019.

[17] F. Zapata, O. Lerma, L. Valera, and V. Kreinovich, 'How to speed up software migration and modernization: Successful strategies developed by precisiating expert knowledge', *Annual Conference of the North American Fuzzy Information Processing Society - NAFIPS*, vol. 2015-September, Sep. 2015, doi: 10.1109/NAFIPS-WCONSC.2015.7284166.

[18] F. Ponce, G. Marquez, and H. Astudillo, 'Migrating from monolithic architecture to microservices: A Rapid Review', *Proceedings - International Conference of the Chilean Computer Science Society, SCCC*, vol. 2019-Novem, 2019, doi: 10.1109/SCCC49216.2019.8966423.

[19] D. Kuryazov, D. Jabborov, and B. Khujamuratov, 'Towards Decomposing Monolithic Applications into Microservices', *14th IEEE International Conference on Application of Information and Communication Technologies, AICT 2020 - Proceedings*, pp. 2–5, 2020, doi: 10.1109/AICT50176.2020.9368571.

[20] S. Tyszberowicz, R. Heinrich, B. Liu, and Z. Liu, 'Identifying Microservices Using Functional Decomposition', 2018, pp. 50–65. doi: 10.1007/978-3-319-99933-3_4.

[21] M. Ahmadvand and A. Ibrahim, 'Requirements Reconciliation for Scalable and Secure Microservice (De)composition', in *2016 IEEE 24th International Requirements Engineering Conference Workshops (REW)*, Sep. 2016, pp. 68–73. doi: 10.1109/REW.2016.026.

[22] M. Gysel, L. Kölbener, W. Giersche, and O. Zimmermann, 'Service Cutter: A Systematic Approach to Service Decomposition', 2016, pp. 185–200. doi: 10.1007/978-3-319-44482-6_12.

[23] A. Levcovitz, R. Terra, and M. T. Valente, 'Towards a Technique for Extracting Microservices from Monolithic Enterprise Systems', May 2016.