Mediterranean Smart Cities Conference (MSCC 2024)
2-4 May, 2024, Martil, Morocco

Dynamic Decomposition of Monolith Applications Into Microservices Architectures

AIT MANSSOUR Nassima
Mathematics, computer science and application laboratory FST Mohammedia,
Hassan II university of Casablanca, Morocco
nassima.aitmansour-etu@etu.univh2c.ma

SBAI Hanae
Mathematics, computer science and application laboratory FST Mohammedia,
Hassan II university of Casablanca, Morocco
hanae.sbai@univh2c.ma

Baïna Karim
ADMIR Laboratory , ENSIAS
Mohammed V Uni in Rabat Rabat, Morocco
karim.baina@um5.ac.ma

Abstract— **As monolithic systems increase in size, functionality, and complexity, they become more and more unmaintainable and scalable. As each service is meant to have the following qualities clearly defined functionality, adequate modularity, and the capacity to expand independently it is generally assumed that microservice-based designs can mitigate these issues. Industrial procedures demonstrate how difficult and labor intensive it is to extract services from a legacy monolithic system. In this paper, we present a new framework based on event log (runtime trace collection) and process mining techniques for automatically developing and automating the decomposition of monolithic applications into microservice architecture.**

Keywords—Microservice, monolith decomposition, service candidate, execution trace, functionality, evolvability.

I. Introduction

Successful software systems expand in size and complexity over time as a result of the addition of several functionalities that lead to strongly connected but less cohesive components. Monolithic designs have inherent limitations related to scalability, maintenance, and deployment performance since they concentrate functionality into scalable components. On the other hand, distributed microservice architectures encourage the dismantling of systems into multiple, typically tiny, independent components that may be deployed as needed. These components offer advantages including increased scalability and more frequent deployments.

One specific problem for businesses that have monolith-based systems in place is breaking those systems down into cohesive microservice based implementations. By examining the domain of the application, this decomposition is occasionally intended to assist migration engineers in identifying microservice candidates [1]. Analysis of the source code [2] execution traces are some more methods.

In order to automatically build and automate the breakdown of monolithic applications into microservice architecture, this research provides an approach based on event log (runtime trace collecting) and process mining techniques since the performance and maintainability can be impacted by runtime dependencies, such as repeated method calls.

We provide a background summary in the next section. Section 3 describes the work on microservice deconstruction from monolithic applications. The next piece (section 4) then goes into detail on how to construct a new framework. The final section, "Conclusion," provides an overview of the work completed to date as well as the numerous areas that still need development.

II. BACKGROUND AND FOUNDATION

To better understand the decisions made by the authors while creating the proposed approach, this section provides the background of a literature review conducted on the following topics: Monolithic vs. microservices architecture and process mining technique.

Monolithic vs. microservices architecture

A monolithic application is one that integrates all or most of its modules into sizable, independent, self-contained components that are not dependent on other applications [3]. Monolithic design can limit deployment and maintenance flexibility when an application grows too big and complicated, despite having the benefit of allowing major portions of its operation to be localized in a single, controllable area [4]. Furthermore, as application utilization rises, the scalability of monolithic systems may pose significant issues due to their size and load dispersion. [3] provide a definition of microservices, stating that they are autonomous and cohesive processes that communicate through messages. Every microservice is created to carry out a particular function for the application and has its own data model as well as class model. According to Johannes Thones, a microservice is:

[3]"a small application with a single responsibility that can be deployed, scaled, and tested independently".
Scalability, the ability to update individual microservices on a decentralized basis, and the need for simple mechanisms for orchestration and choreography are the issues that microservice architecture seeks to solve.

Process Mining

Process Mining has developed into a powerful family of process science techniques for business process analysis and optimization. However, Process Mining offers benefits to management, including better decision making. Three primary strata are frequently distinguished in process mining techniques:

- Process discovery, which generates a business process as an output and needs an event log.
- Process compliance: to ascertain whether it complies with the given event log, two things are needed, such as an event log and the pertinent business process.
- Process augmentation: this technique allows processes to be enhanced with new data obtained from event logs.

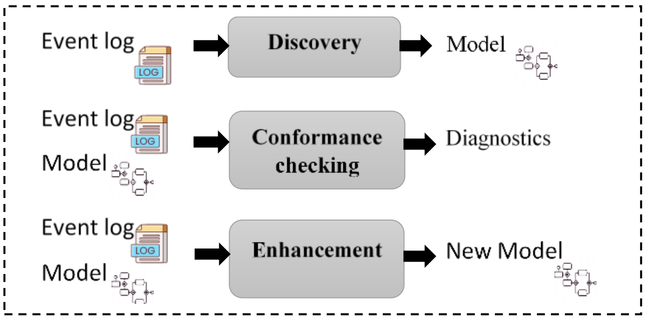

Figure 1: Process mining layers

III. Related Work

[7]the issue of migrating from monoliths to microservices stems from the absence of well-defined methodologies and instruments for recognizing and evaluating microservices breakdown techniques[5].The systematic review's conclusions imply that the process of breaking down monolithic apps into microservices is still in its infancy. [6] current methods for locating microservices in monolithic systems lack a systematic approach and are frequently dependent on experience and intuition [7]. Combining data from several representations of monolith systems, such as code structure and development history, results in improved decompositions, according to recent study [5]. Furthermore, modelling techniques and tools are required for the migration of monoliths to microservices, in addition to automated methods for locating microservices and creating microservice models[6]. All things considered, additional testing, contrasting, and tool and technique development are needed to ease the transition from monolithic to microservices architecture.
[5]Based on different approaches for the automated identification of candidate microservices in monolith systems The approach in demonstrates how crucial it is to take the development history into account when selecting microservices and offers information on the relative merits of various decomposition process representations. The study finds that, although only a small number of codebases and decompositions were examined, the best decompositions on each measure were obtained by combining data from the sequences of accesses and the development history representations. There are not many ways that can be compared. Utilizing the hierarchical clustering technique [5] and [6], they aim to reduce the length of traces.
In[7], the authors propose sub process model discovery utilizing state-based region algorithms and region-based algorithms on languages. Every trace in a log can be replicated in the model thanks to state-based region algorithms, which guarantee flawless fitness. The state explosion problem and other potential issues brought on by a high level of concurrency and inadequate event logs are acknowledged in the study.
In the work [7], authors contend that the process of breaking down monolithic apps into microservices is still in its infancy, citing a review as support. The findings of this systematic review can help scholars identify areas for future research and can also help practitioners understand the aspects of monolith breakdown and present capabilities in this field, using five quality metrics that evaluate modularity, minimization of the number of transactions per functionality, and reduction of teams and communication, in[7],the analysis comes to the conclusion that integrating data from the development history representations and access sequences produced the best decompositions on each metric.
Microservices can be created by breaking down monolithic programs via process mining. It gives information about the system's real behavior, which makes microservice identification more precise[7]. Process mining can assist in determining the proper boundaries for microservices by examining the source code of the application and its development history[8]. In order to group components with related business domains and produce a more accurate decomposition, this approach depends on certain keyword extraction and traversal algorithms [9]. Process mining has advantages when it comes to decomposition, though. One problem is the absence of standardized metrics and datasets for the assessment and validation of the decomposition process [10]. Other drawbacks include inadequate tool support and a lack of techniques for merging static, dynamic, and evolutionary data [5].
Upon closer inspection, we find that most existing techniques have limitations regarding their automatic capacity to decompose relationships and construct suitable decompositions. It seems that these methods are not appropriate for optimizing event logs according to variability principles. Consequently, we present a process mining based technique. After the event logs have been improved, process mining techniques will be applied to find configurable process models.

IV. Approch overview

Figure 2 displays all of the processes involved in the decomposition process, along with its intermediate stages and end result. Both single microservice and monolithic apps can be broken down using this method. This data can be saved in an event log as part of the mining process and decomposed based on the many user requests with the assistance of a configurable business.

We begin our decomposition process with data extraction; in this step, we create a web log containing the traces of the methods and classes that users have requested; all traces are organized in a chorological fashion. Following that, we must perform a decomposition using methods for defining variability and the process mining algorithm to produce a set of models.

Based on those models, we can extract our microservice architecture.

We will utilize a decomposed specification file, which includes decomposition criteria like granularity, which indicates the size of each microservice, cohesiveness, which guarantees that each microservice has a unique, well-defined purpose, autonomous, and other characteristics, to produce a good decomposition.

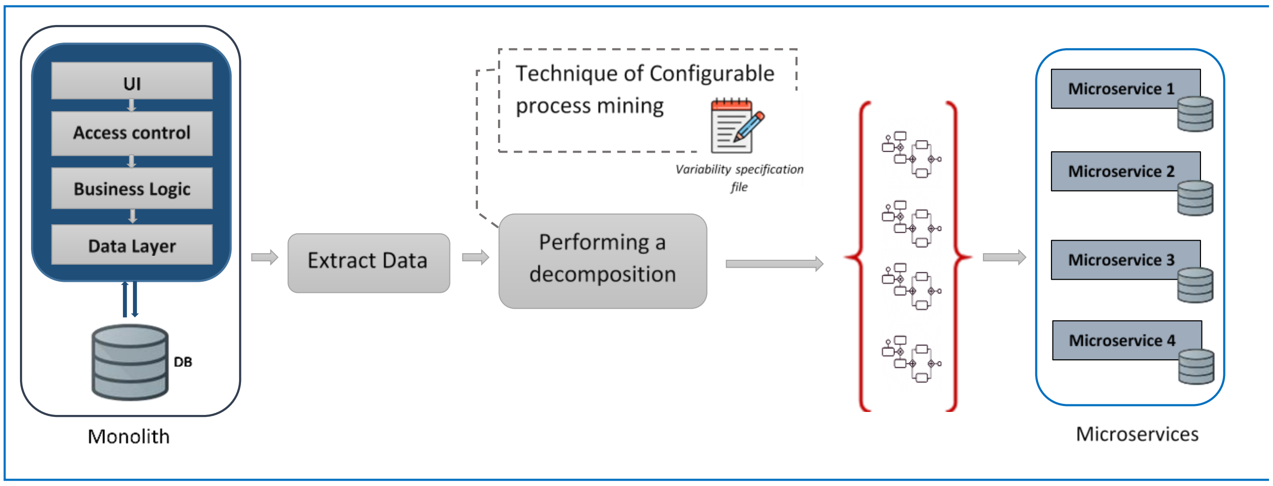

Figure 2 Automated Decomposition of Monolith Applications Into Microservices Architectures via Process Mining Techniques

V. Conclusion

The use of microservice design in business has grown rapidly. With this architecture, we may divide enormous, monolithic programs into more manageable, autonomous services that are loosely connected. Each microservice in the software is designed to fulfill a specific purpose, and they communicate with each other using lightweight protocols such as HTTP or messaging. The separation of concerns is the cornerstone of the microservices architecture, in which each service is deployed in a different container and is in charge of a specific feature or business function. This allows for greater flexibility, scalability, and agility in the development and deployment of software applications.

The program architect typically handles this tedious and error-prone task of disassembling monolithic systems into microservices by hand. While there are a number of migration options, the dynamic approach—which uses runtime execution to identify all dependencies and classes used at runtime—is one of the most crucial and successful ways to set up the microservice architecture. The independent dependencies of the monolithic code can be easily identified using a customisable business method. As part of the mining process, this data may be recorded in an event log. Because each microservice is responsible for managing its own data store, database decomposition is a fundamental component of microservice design.

References

[1] M. Daoud, A. El Mezouari, N. Faci, D. Benslimane, Z. Maamar, and A. El Fazziki, "A multi-model based microservices identification approach," Journal of Systems Architecture, vol. 118, p. 102200, Sep. 2021, doi: 10.1016/j.sysarc.2021.102200.

[2] "(PDF) Towards Automated Microservices Extraction Using Muti-objective Evolutionary Search." Accessed: Jan. 29, 2024. [Online]. Available: https://www.researchgate.net/publication/336816445_Towards_Automated_Microservices_Extraction_Using_Muti-objective_Evolutionary_Search

[3] N. Dragoni et al., "Microservices: yesterday, today, and tomorrow," arXiv.org. Accessed: Jul. 27, 2023. [Online]. Available: https://arxiv.org/abs/1606.04036v4

[4] J. Fritzsch, J. Bogner, A. Zimmermann, and S. Wagner, "From Monolith to Microservices: A Classification of Refactoring Approaches," vol. 11350, 2019, pp. 128–141. doi: 10.1007/978-3-030-06019-0_10.

[5] J. Lourenço and A. R. Silva, "Monolith Development History for Microservices Identification: a Comparative Analysis," in 2023 IEEE International Conference on Web Services (ICWS), Jul. 2023, pp. 50–56. doi: 10.1109/ICWS60048.2023.00019.

[6] "Log2MS: a framework for automated refactoring monolith into microservices using execution logs | IEEE Conference Publication | IEEE Xplore." Accessed: Jan. 30, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/9885771

[7] Y. Abgaz et al., "Decomposition of Monolith Applications Into Microservices Architectures: A Systematic Review," IEEE Transactions on Software Engineering, pp. 1–32, 2023, doi: 10.1109/TSE.2023.3287297.

[8] "Decomposing Monolithic to Microservices: Keyword Extraction and BFS Combination Method to Cluster Monolithic's Classes,"

Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi), vol. 7, no. 2, Art. no. 2, Mar. 2023, doi: 10.29207/resti.v7i2.4866.

[9] J. Kazanavičius and D. Mažeika, “EVALUATION OF MICROSERVICE COMMUNICATION WHILE DECOMPOSING MONOLITHS,” Computing and Informatics, vol. 42, no. 1, pp. 1–36, 2023, doi: 10.31577/cai_2023_1_1.

[10] M. H. Hasan, M. H. Osman, N. I. Admodisastro, and M. S. Muhammad, “A Quality Driven Framework for Decomposing Legacy Monolith Applications to Microservice Architecture,” In Review, preprint, Jun. 2023. doi: 10.21203/rs.3.rs-3060410/v1.