# Visually-Assisted Decomposition of Monoliths to Microservices

Breno Salles
Faculty of Engineering, University of Porto, Portugal
scholar@brenosalles.com

Jácome Cunha
Faculty of Engineering, University of Porto
& HASLab/INESC TEC, Portugal
jacome@fe.up.pt

***Abstract*—The architectural style of microservices has received much attention from both business and academia and converting a monolithic application into a microservice-based one has become a regular practice. However, companies struggle with migrating their existing monolithic applications to microservices and software engineers frequently face challenges due to a lack of awareness of alternative migration methodologies, making the migration process even harder.**

**In this paper, we present a framework to help software engineers during the migration process by addressing gaps in understanding various migration tools and approaches, allowing for easy comparison between multiple options. Our tool combines multiple existing approaches into one platform, allowing a comprehensive visualization of migration proposals and comparing different options offered by already existing approaches.**

***Index Terms*—visual assistant, software migration, software evolution, microservices**

## I. Introduction

Microservices is an architectural style that evolved from Service Oriented Architecture (SOA). Just like SOA, microservices are an alternative to monolithic architecture. The main contrasts are that, while monolithic applications are software systems with a single, integrated codebase that includes all necessary components, and features [1], microservices tend to be separated, and loosely coupled [2]. Also, while monoliths tend to be easier to develop, at least in an initial phase, they may scale poorly and are harder to maintain when compared to microservices [3]. Microservices are increasingly being used in the development of modern applications, particularly in the areas of cloud computing [4]. Many organizations, including large enterprises and startups, are adopting microservices as a way to build and deploy applications more quickly and efficiently [5]. Microservices are particularly well-suited for distributed, cloud-based environments, where they can take advantage of the flexibility and scalability of the cloud [6]. This type of architecture is already being applied in multiple well-known companies, like Uber, Netflix, eBay [7], [8], and also being followed by the rest of the herd when compared to monoliths [9].

Refactoring monoliths to microservices is a heavily debated topic both in the academic world and the industry. The main outcomes from this debate are that refactoring is difficult and time-consuming, and companies struggle with migrating their already existing monolithic applications to microservices [10]. To help address this, some tools have been developed [11]–[13]. However, in today's world, where the amount of data and information is constantly increasing, it would be ideal to have a centralized location where software engineers can access and utilize all the tools that are currently available. However, currently, no tool offers such a possibility.

In this paper, in Section III, we present an application that aims to aggregate existing tools into a single platform and provide the means to extend and incorporate new tools. This application offers a convenient and comprehensive way to access and use various tools that help the decomposition from monoliths to microservices and provide them with a perspective on several decomposition proposals, allowing for easily comparable and different combinations options.

It is important to mention that the final objective is not to create a new technique for discovering microservices in a monolith system, but rather to aggregate the already existing ones into a single and comprehensive framework.

## II. Related Work

The majority of the existing works for the decomposition of monoliths into microservices target Java as their primary programming language for input [12], [14]–[22]. This may be attributed to the language's strict syntax rules, which facilitate the examination of source code during the inspection process. Additionally, some of them rely in the Spring Boot framework in conjunction with Java [11], [13], [21], [23]–[25], which further enforces structure through the utilization of decorators. In contrast, those who employed programming languages other than Java, such as Python, utilized corresponding frameworks like Django, to compensate for the language's more lenient syntax constraints [26], [27].

Nevertheless, the number of tools available to decompose monolithic architectures into microservices is very limited. From the works presented, we identified seven free and open-source tools [11], [13], [15], [20], [24], [26], [28], each with varying degrees of completeness. However, it is worth noting that the most promising tool identified, IBM's Mono2Micro [12], [19], [22], is not publicly accessible to the general public.

Moreover, we found no existing application that served as an aggregator of multiple decomposition tools, offering users a graphical and unified interface, which is the goal of this work.

This work is financed by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, within project UIDP/50014/2020.

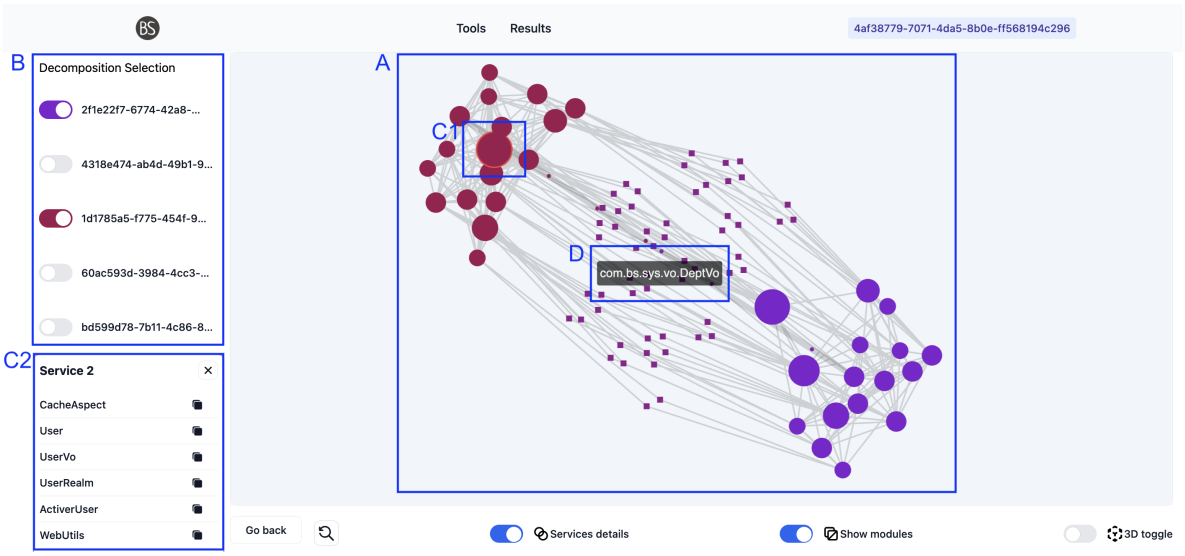


Fig. 1. Tool main page

## III. Proposed Approach

The main features of our tool can be seen in Figure 1:

A) Visualise microservices generated by each decomposition, enabling users to gain insights into the composition and structure of the system components. This visual representation helps users differentiate between the decompositions and perceive the interconnection of modules within related decompositions.

B) Toggle the visibility of each decomposition, allowing users to focus on the ones they want to compare. This functionality is beneficial in various scenarios, such as when users want to concentrate on a single decomposition, compare decompositions side by side, or exclude a discarded decomposition from the view.

C) Focus on a microservice and check its constituent modules, allowing users to delve into the details of each microservice and understand its internal components. When a user clicks on a specific microservice, the selected microservice becomes highlighted (C1). Additionally, a new window opens, providing detailed information about the modules contained within the focused microservice (C2).

D) Compare modules across different decompositions, which helps users to identify similarities, differences, and variations in the composition of the system components. This feature enables users to observe the connections and relationships between modules across different microservices and microservices from different decompositions by hovering each of the squares.

To create the visualization of the decomposition, we considered four tools: Graphviz[1], D3.js[2], Gephi[3], and Chart.js[4]. Each tool was assessed based on four categories: customizability, ease of use, charts available, and real-time interactivity. Considering the industry's preference for node graphs in presenting microservices [29], and after careful consideration of the advantages and disadvantages, we chose D3.js.

In addition to having a powerful tool, the way information is presented and the visual elements used are crucial. As suggested by Moody [30], using different shapes, colors, strokes, and line dashes to depict entities and their relationships is an effective way of expressing variation between entities in visualizations. Table I illustrates how each visual representation expresses different entities, highlighting the specific shapes and visual cues employed to convey information effectively.

A prototype of the tool can be used at https://frontend-mesw.brenosalles.com/.

TABLE I
Visual Expressiveness of Each Entity

| Entity | Shape | Size | Colour |
|---|---|---|---|
| Service | Circle | Variable according amount of modules | Based on the selected decomposition |
| Module | Square | Static | Mix between selected decompositions |
| Relationships | Line | Static | Static |

## IV. Conclusion

In this paper, we present the first framework capable of aggregating in a single tool different microservices' proposals produced by different approaches. This environment allows the users to explore each proposal and compare them, aiding in the decomposition of monoliths.

[1]https://graphviz.org/
[2]https://d3js.org/
[3]https://gephi.org/
[4]https://chartjs.org/

## REFERENCES


[1] J. Kazanavičius and D. Mažeika, "Migrating legacy software to microservices architecture," in *2019 Open Conference of Electrical, Electronic and Information Sciences (eStream)*. IEEE, 2019, pp. 1–5.

[2] S. Newman, *Building microservices*. " O'Reilly Media, Inc.", 2021.

[3] ——, *Monolith to microservices: evolutionary patterns to transform your monolith*. O'Reilly Media, 2019.

[4] A. Balalaie, A. Heydarnoori, and P. Jamshidi, "Migrating to cloud-native architectures using microservices: an experience report," in *Advances in Service-Oriented and Cloud Computing: Workshops of ESOCC 2015, Taormina, Italy, September 15-17, 2015, Revised Selected Papers 4*. Springer, 2016, pp. 201–215.

[5] C. Richardson, "Microservice architecture," https://microservices.io/patterns/microservices.html, last accessed 4 January 2023.

[6] M. Fowler, "Microservice prerequisites," https://martinfowler.com/bliki/MicroservicePrerequisites.html, 2014, last accessed 4 January 2023.

[7] C. Richardson, "Who is using microservices," https://microservices.io/articles/whoisusingmicroservices.html, last accessed 4 January 2023.

[8] Z. Ren, W. Wang, G. Wu, C. Gao, W. Chen, J. Wei, and T. Huang, "Migrating web applications from monolithic structure to microservices architecture," in *Proceedings of the tenth asia-pacific symposium on internetware*, 2018, pp. 1–10.

[9] D. Taibi, V. Lenarduzzi, and C. Pahl, "Processes, motivations, and issues for migrating to microservices architectures: An empirical investigation," *IEEE Cloud Computing*, vol. 4, no. 5, pp. 22–32, 2017.

[10] M. Kamimura, K. Yano, T. Hatano, and A. Matsuo, "Extracting candidates of microservices from monolithic application code," in *2018 25th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2018, pp. 571–580.

[11] M. Brito, J. Cunha, and J. Saraiva, "Identification of microservices from monolithic applications through topic modelling," in *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, 2021, pp. 1409–1418.

[12] A. K. Kalia, J. Xiao, R. Krishna, S. Sinha, M. Vukovic, and D. Banerjee, "Mono2micro: a practical and effective tool for decomposing monolithic java applications to microservices," in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2021, pp. 1214–1224.

[13] F. Freitas, A. Ferreira, and J. Cunha, "Refactoring java monoliths into executable microservice-based applications," in *25th Brazilian Symposium on Programming Languages*, 2021, pp. 100–107.

[14] O. Al-Debagy and P. Martinek, "A microservice decomposition method through using distributed representation of source code," *Scalable Computing: Practice and Experience*, vol. 22, no. 1, pp. 39–52, 2021.

[15] A. Bucchiarone, K. Soysal, and C. Guidi, "A model-driven approach towards automatic migration to microservices," in *International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*. Springer, 2020, pp. 15–36.

[16] W. K. Assunção, T. E. Colanzi, L. Carvalho, A. Garcia, J. A. Pereira, M. J. de Lima, and C. Lucena, "Analysis of a many-objective optimization approach for identifying microservices from legacy systems," *Empirical Software Engineering*, vol. 27, no. 2, pp. 1–31, 2022.

[17] J. Zhao and K. Zhao, "Applying microservice refactoring to object-2riented legacy system," in *2021 8th International Conference on Dependable Systems and Their Applications (DSA)*. IEEE, 2021, pp. 467–473.

[18] V. Nitin, S. Asthana, B. Ray, and R. Krishna, "Cargo: Ai-guided dependency analysis for migrating monolithic applications to microservices architecture," in *37th IEEE/ACM International Conference on Automated Software Engineering*, 2022, pp. 1–12.

[19] A. K. Kalia, J. Xiao, C. Lin, S. Sinha, J. Rofrano, M. Vukovic, and D. Banerjee, "Mono2micro: an ai-based toolchain for evolving monolithic enterprise applications to a microservice architecture," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, pp. 1606–1610.

[20] S. Agarwal, R. Sinha, G. Sridhara, P. Das, U. Desai, S. Tamilselvam, A. Singhee, and H. Nakamuro, "Monolith to microservice candidates using business functionality inference," in *2021 IEEE International Conference on Web Services (ICWS)*. IEEE, 2021, pp. 758–763.

[21] I. Pigazzini, F. Arcelli Fontana, and A. Maggioni, "Tool support for the migration to microservice architecture: An industrial case study," in *European Conference on Software Architecture*. Springer, 2019, pp. 247–263.

[22] R. Krishna, A. Kalia, S. Sinha, R. Tzoref-Brill, J. Rofrano, and J. Xiao, "Transforming monolithic applications to microservices with mono2micro," in *Proceedings of the 36th IEEE/ACM International Conference on Automated Software Engineering*, 2021, pp. 3–3.

[23] Y. Wei, Y. Yu, M. Pan, and T. Zhang, "A feature table approach to decomposing monolithic applications into microservices," in *12th Asia-Pacific Symposium on Internetware*, 2020, pp. 21–30.

[24] L. Nunes, N. Santos, and A. Rito Silva, "From a monolith to a microservices architecture: An approach based on transactional contexts," in *European Conference on Software Architecture*. Springer, 2019, pp. 37–52.

[25] A. Santos and H. Paula, "Microservice decomposition and evaluation using dependency graph and silhouette coefficient," in *15th Brazilian Symposium on Software Components, Architectures, and Reuse*, 2021, pp. 51–60.

[26] T. Matias, F. F. Correia, J. Fritzsch, J. Bogner, H. S. Ferreira, and A. Restivo, "Determining microservice boundaries: a case study using static and dynamic software analysis," in *European Conference on Software Architecture*. Springer, 2020, pp. 315–332.

[27] L. v. Asseldonk, "From a monolith to microservices: the effect of multi-view clustering," Master's thesis, Utrecht University, 2021.

[28] X. Sun, S. Boranbaev, S. Han, H. Wang, and D. Yu, "Expert system for automatic microservices identification using api similarity graph," *Expert Systems*, p. e13158, 2022.

[29] T. Cerny, A. S. Abdelfattah, V. Bushong, A. Al Maruf, and D. Taibi, "Microservice architecture reconstruction and visualization techniques: A review," in *2022 IEEE International Conference on Service-Oriented System Engineering (SOSE)*. IEEE, 2022, pp. 39–48.

[30] D. Moody, "The "physics" of notations: toward a scientific basis for constructing visual notations in software engineering," *IEEE Transactions on software engineering*, vol. 35, no. 6, pp. 756–779, 2009.