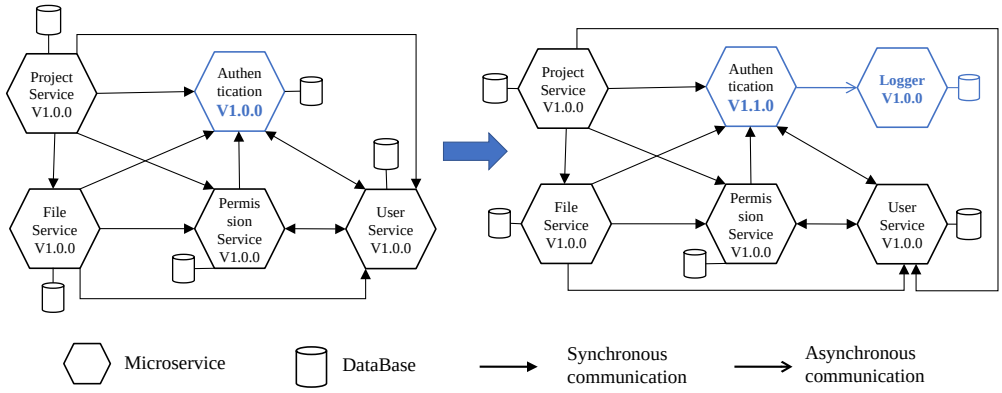
Master Internship—5 to 6 months

ARBORE: Dynamic software updating of microservice architectures

Denis Conan and Sophie Chabridon

Keywords: Distributed middleware and algorithms; Microservice architecture; Software evolution.
Technological background: REST and Pub/Sub; Concurrent prog. in Golang; Docker and Kubernetes.

Context. Microservice architectures contribute to building complex distributed systems as sets of independent microservices. The decoupling and modularity of distributed microservices facilitates their independent replacement and upgradeability. Since the emergence of agile DevOps and CI/CD, there is a trend towards more frequent and rapid evolutionary changes of the running microservice-based applications in response to various evolution requirements. Applying changes to microservice architectures is performed by an evolution process of moving from the current application version to a new version.



Research Work. The objective of ARBORE (ARchitecting Based on microservice versiOns with REconfiguration) is to address the following issues: When can microservice-based applications, especially those with long-running activities, be dynamically updated without stopping the execution of the whole system? How should the safe updating be performed to ensure service continuity and maintain system consistency? In response to these questions, ARBORE is a snapshot-based approach for dynamic software updating (DSU) of microservices. The basis of the DSU algorithm is a distributed termination detection algorithm that relies on consistent distributed snapshots. Mainly, microservices count application messages that are sent and received for each collaboration, and provide these counters as part of their snapshot. Then, a central entity, called the autonomic manager (AM), takes a consistent distributed snapshot and calculates termination detection by using the counters provided by the microservices. In the context of a consistent distributed snapshot, AM evaluates the update conditions from the termination detection of collaborations. Next, the DSU algorithm is organised into waves and around two main actions: waiting for the update condition (When can updating be performed safely?) and applying then an update strategy (does the system allow both versions of a microservice to run simultaneously? should the DSU algorithm block certain messages, or certain architectural elements when updating?).

Expected tasks.

- Understand the three update conditions of the literature (quiescence [2], freeness [3, 1], and essential freeness [4]) and the ARBORE solution [5]
- Contribute to the open source implementation: `https://gitlabev.imtbs-tsp.eu/mimosae/arbore` and `https://gitlabev.imtbs-tsp.eu/mimosae/mimosae`.
- Benchmark the solution.

This research project is part of the research work of the DiSSEM group of the ACMES team of the SAMOVAR laboratory, Télécom SudParis, Institut Polytechnique de Paris.

References.

[1] L. Baresi, C. Ghezzi, X. Ma, and V. Panzica La Manna. Efficient Dynamic Updates of Distributed Components Through Version Consistency. *IEEE Trans. on Software Eng.*, 43(4):340–358, April 2017.

[2] J. Kramer and J. Magee. The evolving philosophers problem: dynamic change management. *IEEE TOSE*, 16(11), November 1990.

[3] X. Ma, L. Baresi, C. Ghezzi, V. Panzica La Manna, and J. Lu. Version-Consistent Dynamic Reconfiguration of Component-Based Distributed Systems. In *19th ACM SIGSOFT Symp. and 13th European Conf. on Foundations of Software Engineering*, 2011.

[4] D. Sokolowski, P. Weisenburger, and G. Salvaneschi. Change Is the Only Constant: Dynamic Updates for Workflows. In *Prco. of the 44th ACM ICSE*, 2022.

[5] Y. Wang. *Evolution of Microservice-based Applications: Modelling and Safe Dynamic Updating.* PhD thesis, Institut Polytechnique de Paris, October 2022.