Microservice Migration: Face-Off Decomposition and Integration Challenges

Riccardo Matteo Boggetti[1,2]
[1] Profesia s.r.l.
[2] https://www.linkedin.com/groups/12160870/

1 Introduction

Microservice architecture is the most architectural style in demand nowadays. Reasons are known by everyone by now: thanks to the availability of technology such as Docker and Kubernetes, microservices underlie cloud-ready solutions (to engage the porting of your application from bare metal to cloud), they also leverage the agile development approach and, finally, they set the stage to think about a platform business model.

Especially for the usage of cloud environments (private or public), that obviously allows companies to ease the infrastructure management and cut-off the infrastructure costs, approaching to microservices is prevalent but requires migration of the application landscape from monolith-style applications, and a microservice architecture is not just a technical issue, but above all a change of state of mind

Because of that, you have to face-off a lot of challenges when you start a "monolith-to-microservice" initiative in your company, but the good result of your journey depends necessarily on the success of mainly two of these challenges: the monolith decomposition and the isolation and decoupling of your business integration processes.

Taking these applications, that are running in production since years after business approvals, UATs and bugfixes, and dismember them on paper, taking care of preserve a yearly history of developments and functionalities, is a huge responsibility and, if in addition of this, you have also a migration deadline, that's why you could have the need of some tools to guide, control and speed up this process.

2 From Monolith to Microservices

Decomposition in a migration project is a teamwork of assessment and analysis that should involve architects, product owners, developers…

There are several approaches to manage the decomposition[1]: service-per-team, business capabilities isolation, domain-driven design.

Domain-driven design decomposition is not just a technical matter, but often entails a corporate reorganization because of the data and process ownership. The domain-driven design is de-facto supported by event-storming[2] sessions, that should involve not just technical professionals but also business references.

Decomposition by business capability is an approach to detect business functionalities inside the system under analysis but is not so simple the distillation of business functionalities even more so the monolith system is complex: you need to match the static code with the dynamicity of the business flows.

The usage of a tool that can automatically and simultaneously manage the scan of your code (static analysis) and detect the live transactions of the runtime applications (dynamic analysis), should ease the task. vFunction[3] is a solution that provides both capabilities for Java-based applications: with a package of components, detects JVM transactions and makes matches with the data collected by the static analysis of the compiled java objects of your application. After that, via a ML engine, vFunction elaborates an assessment graph of your application splitted in isolated functionalities.

vFunction is a tool for analysis purposes, to reduce the effort and time for assessment and, more important, for the modeling of self-consistent components, thanks to a graphical tool for the java classes exclusivity management.

3 Business Integration Processes

The goal of a microservices architecture is the set-up of portable components, each of them providing a specific business logic, ensuring easy maintenance for the developer and the chance to stay focused on the quality of the code, with fewer worries about integration and infrastructure.

With vFunction you speed up the proposal of a microservices mesh.

But in the enterprise world, microservices without integration do not bring any value: the business processes are what add value to the company, but to guarantee self-consistency and autonomy of microservices, you must abstract your business processes and host them in an integration/decoupling layer, manageable via APIs by leveraging the api-centric view of a cloud-native architecture.

Business integration capabilities often follow the enterprise integration patterns[4] and there are solutions that provide these capabilities for developers (do not reinvent the wheel, developing code from scratch): depending on your needs you can choose between iPaaS solutions such as Choreo[5] and WSO2 API Manager[6] for private cloud purposes.

References

[1] https://microservices.io/index.html
[2] https://www.eventstorming.com/
[3] https://vfunction.com/
[4] https://www.enterpriseintegrationpatterns.com/
[5] https://wso2.com/choreo/
[6] https://wso2.com/api-manager/