Give a Preference to Micro Architecture!

There are good reasons for making as many decisions as possible at the micro architecture level rather than at the macro architecture level. Let's discuss each.

Macro architecture decisions: Best practices and advice

- Specifying only a few points in the macro architecture helps with focusing. Many teams have failed when trying to implement a far-reaching unification in a complex project or IT landscape. If there are few macro architecture rules, the chances increase that the rules are actually successfully implemented.

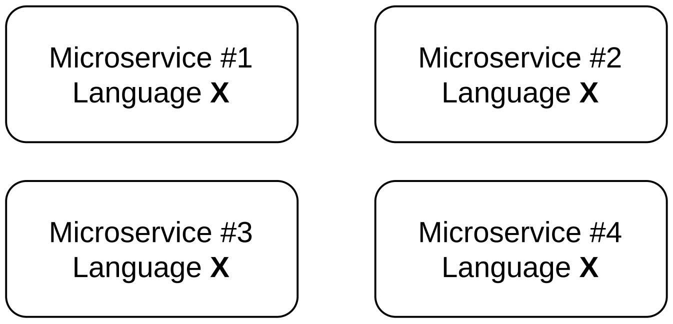
- The rules should be **minimal**.

    - For example, a macro architecture rule can *define the monitoring technology*. However, it is not necessary to standardize *how* the metrics are measured in the application. After all, it is important only that the metrics are created. How this happens is irrelevant. The macro architecture rule should only define a protocol for transferring metrics but leave the selection of the library for creating and transferring metrics to the micro architecture. In this way, the teams can choose the most appropriate technologies.

- The macro architecture rules have to be **consequently enforced**. For example, when the metrics are not generated, the operations team cannot simply bring the microservice into production. It is important to get rid of all unnecessary macro architecture elements.
- In addition, **independence** is an important goal of microservices. Too many macro architecture rules run counter to this goal as they hinder the independence of the teams through central control.
- Complying with macro architecture should be in the **self-interest** of the teams responsible for the microservices. Violations of macro architecture usually mean that microservices cannot go into production because operations cannot support them.

In addition to mandatory macro architecture rules, recommendations and best practices are advisable. However, they do not have to be enforced but are optional for every microservice.

In the end, the goal of macro architecture is to create freedom. Advice and references to best practices are therefore good additions.



Initially, one language may be a restriction

1 of 2

Evolution of macro architecture

At the beginning of a project, **restrictive rules may initially apply**. For example, a single programming language and a fixed stack of libraries can be defined. This reduces learning effort and operating costs.

Over the duration of the project, **more programming languages and libraries can then be allowed** to introduce modern technologies. This leads to a more heterogeneous system which is certainly preferred over updating all microservices at once because such updates entail a high risk.

QUIZ

1 Which of the following macro architecture rules are better and why?

1. User data for analytics must be stored in a JSON format
2. The technology xyz should be used to gather user data for analytics and the data should be stored in a

JSON format

A) Rule #2 because JSON is a standard way to store and retrieve data

B) Rule #1 because it is minimal

C) Either rule #1 or #2 are fine

Submit Answer

Question 1 of 3
0 attempted

Reset Quiz

In the next lesson, we'll discuss organizational aspects!

← Back
Operation: Micro or Macro Architecture?

Next →
Organizational Aspects

✓ Mark as Completed

Report an Issue