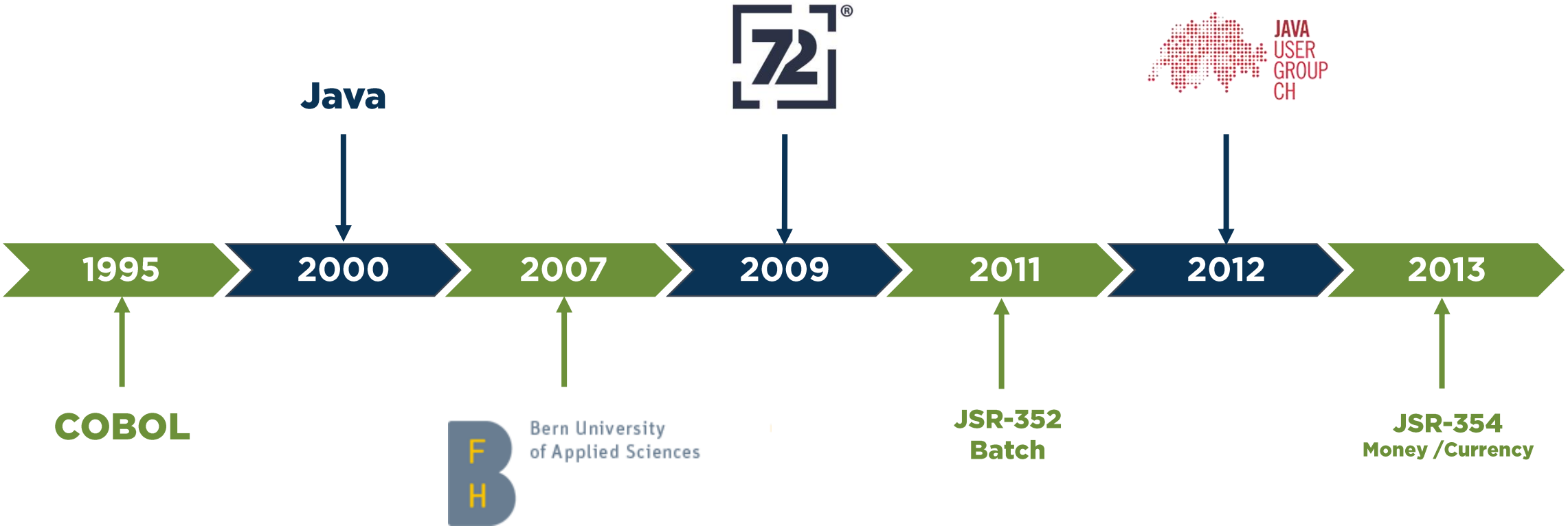
72 Services

THE REAL VALUE OF MICROSERVICES

Simon Martinelli, 72 Services LLC
@simas_ch

ABOUT ME

@simas_ch

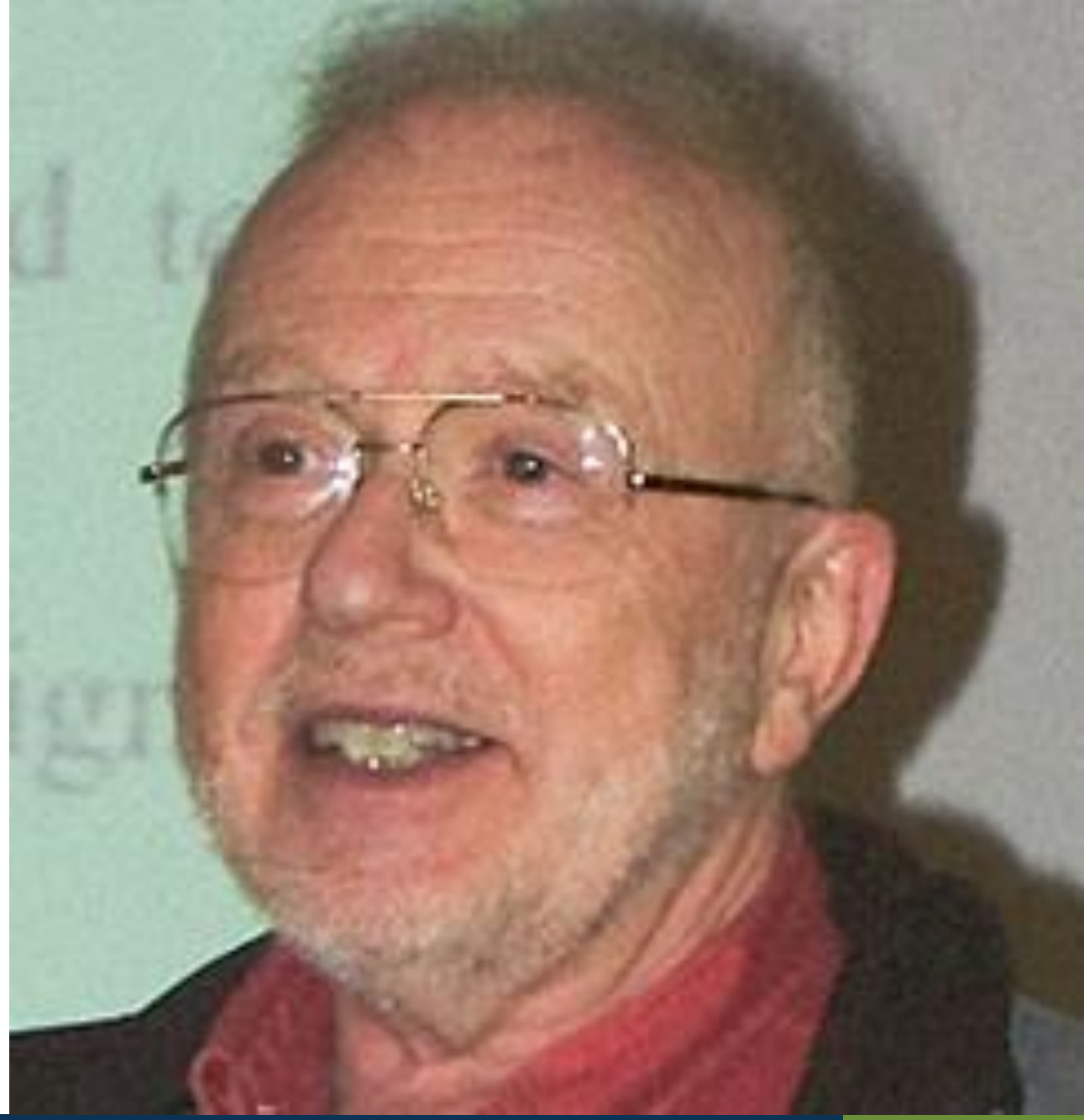


1972

The effectiveness of a “modularization” is dependent upon the criteria used in dividing the system into modules.

On the criteria to be used
in decomposing systems
into modules - David L. Parnas

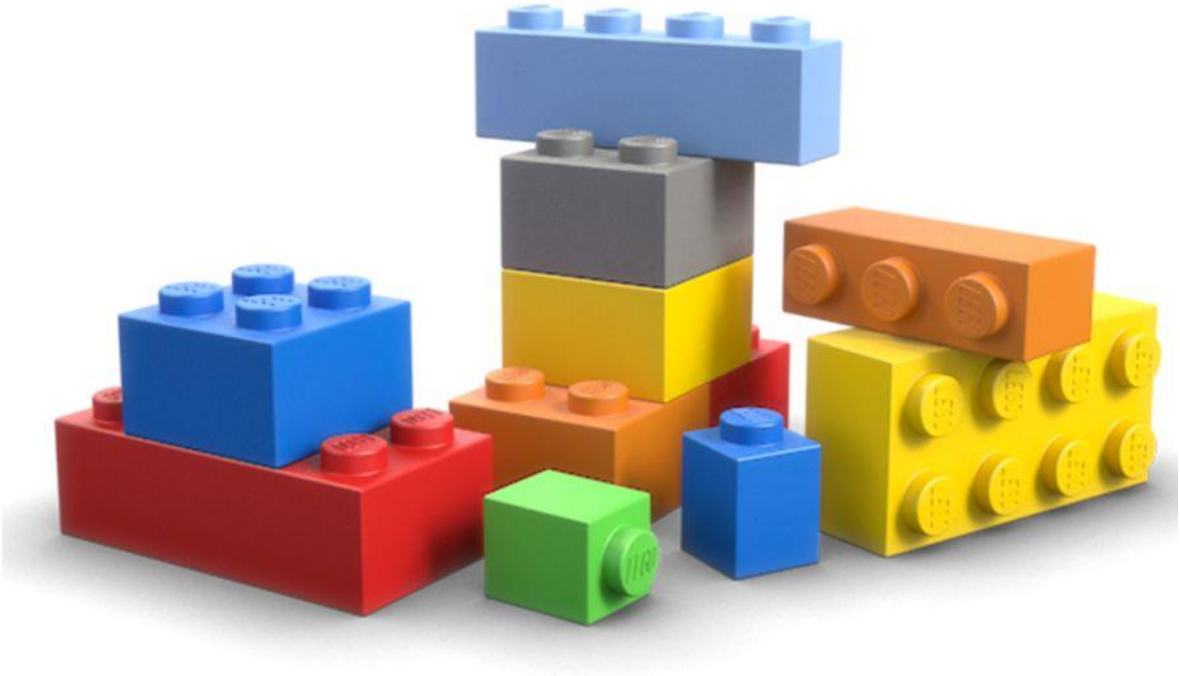
Published in Communications of the ACM
Volume 15 Issue 12, Dec. 1972



IT'S ALL ABOUT MODULARITY

Module

A **self-contained component** of a system, often **interchangeable**, which has a **well-defined interface** to the other components.



Source: https://en.wiktionary.org/wiki/module

DOES THE SIZE MATTER?

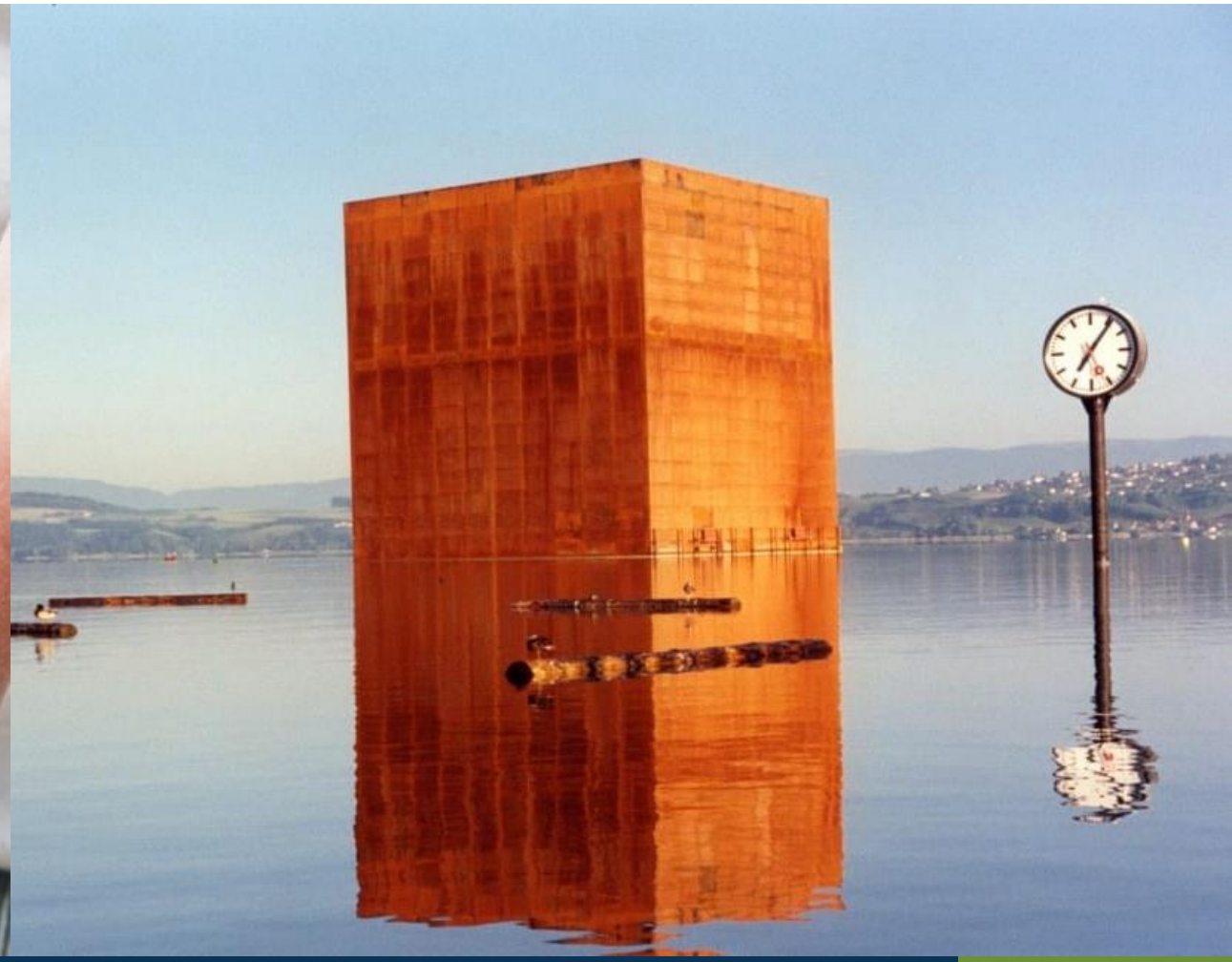
- The term **MICRO is misleading**
- **Scope matters**
  - Bounded Contexts
- **Design for Maintenance**
  - 3 to 10 developers per Microservice during project development
  - But how many developers will maintain the system?

WHAT ABOUT SERVICES?

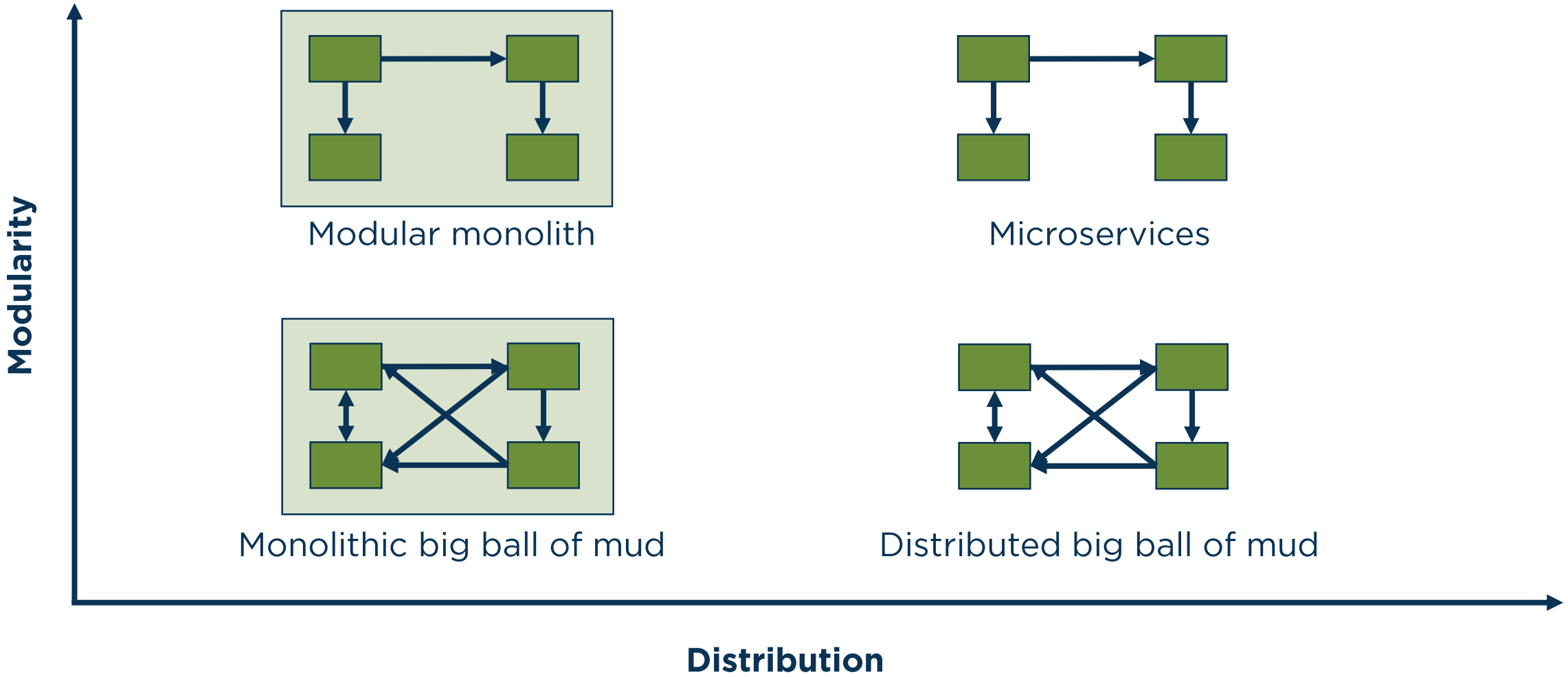
- **Service-Oriented Architecture (SOA)** is an **architectural style** that supports service-orientation
- **Service-orientation** is a **way of thinking** in terms of services and service-based development and the outcomes of services
- **Service**
  - Is a logical representation of a repeatable **business activity** that has a specified outcome (e.g., check customer credit, provide weather data)
  - Is **self-contained**
  - **May be composed** of other services
  - Is a **black box** to consumers of the service

Source: http://www.opengroup.org/soa/source-book/soa/p1.htm

ARE THEY REALLY THE SAME?

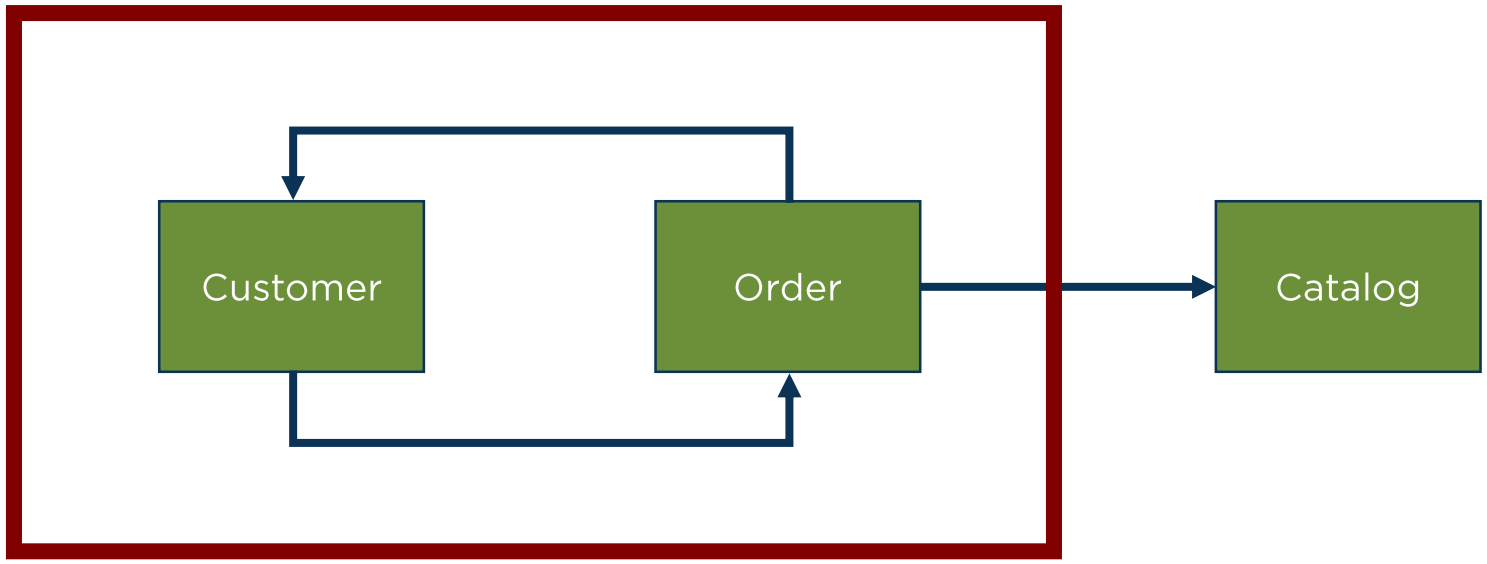


MODULARITY AND DISTRIBUTION



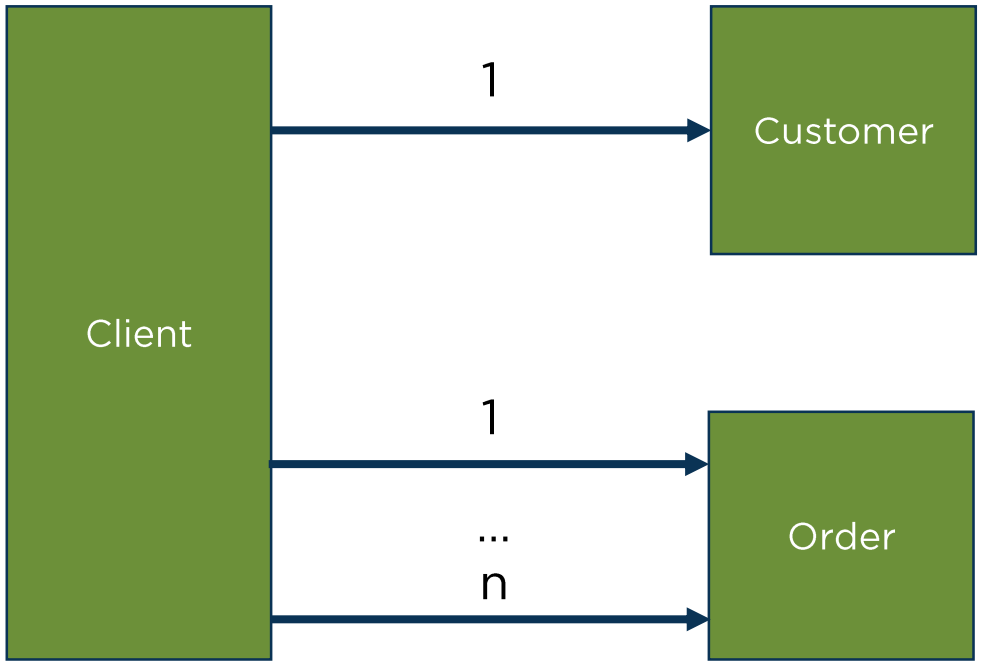
DISTRIBUTED BIG BALL OF MUD

If you can't build a modular monolith, what makes you think microservices are the answer? - Simon Brown



BEWARE OF THE N+1 SELECT PROBLEM

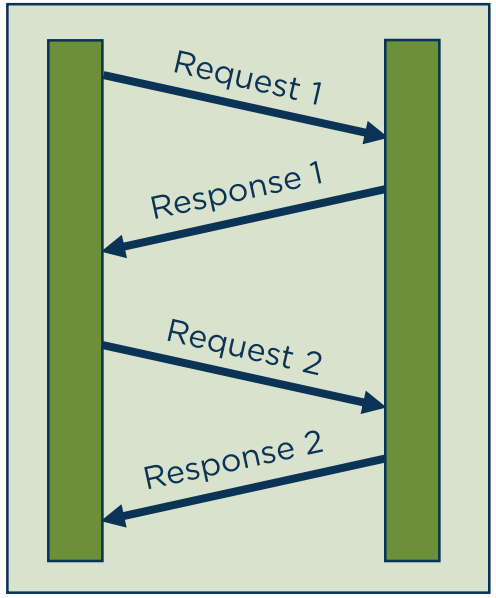
- Use Case: Select orders of customers
  - 1 request to get all customers
  - n requests to get customers orders



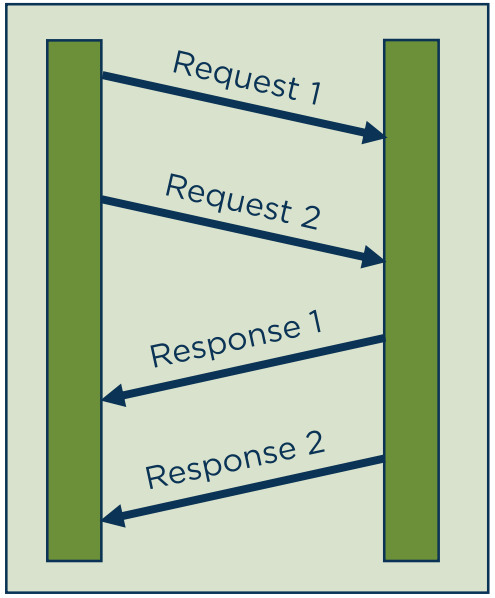
DISTRUBUTION AND COMMUNICATION

First Law of Distributed Object Design: Don't distribute your objects
- Martin Fowler

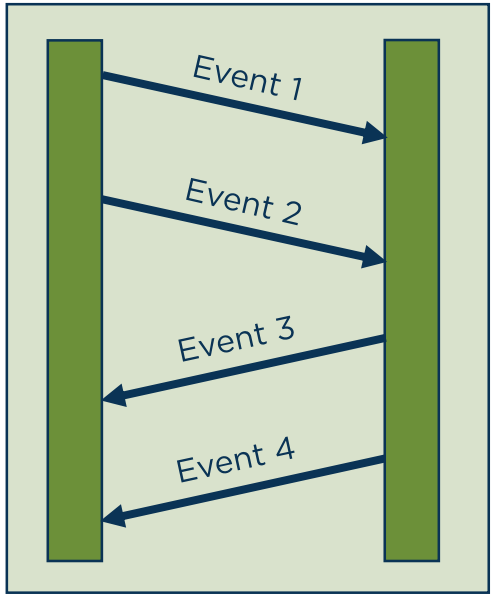
Synchronous



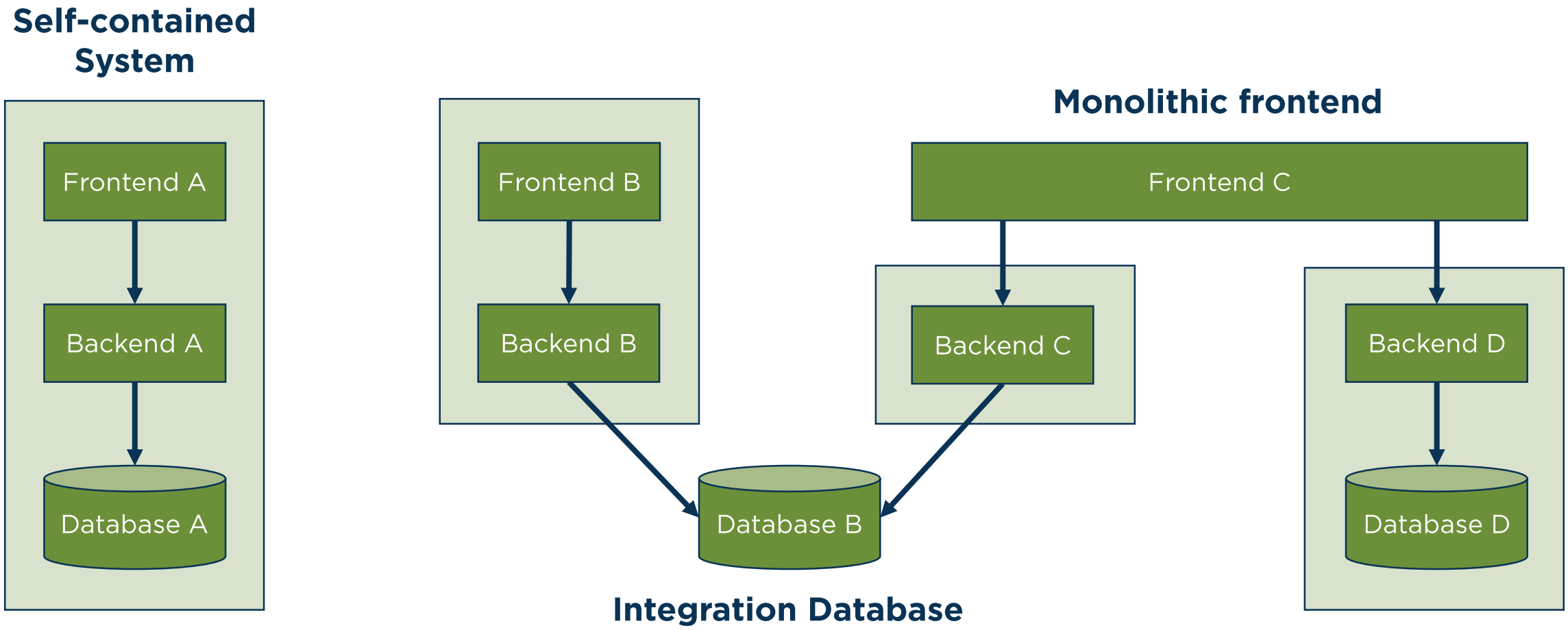
Asynchronous



Event based



SUBSYSTEMS (AKA GARTNERS MINISERVICES)

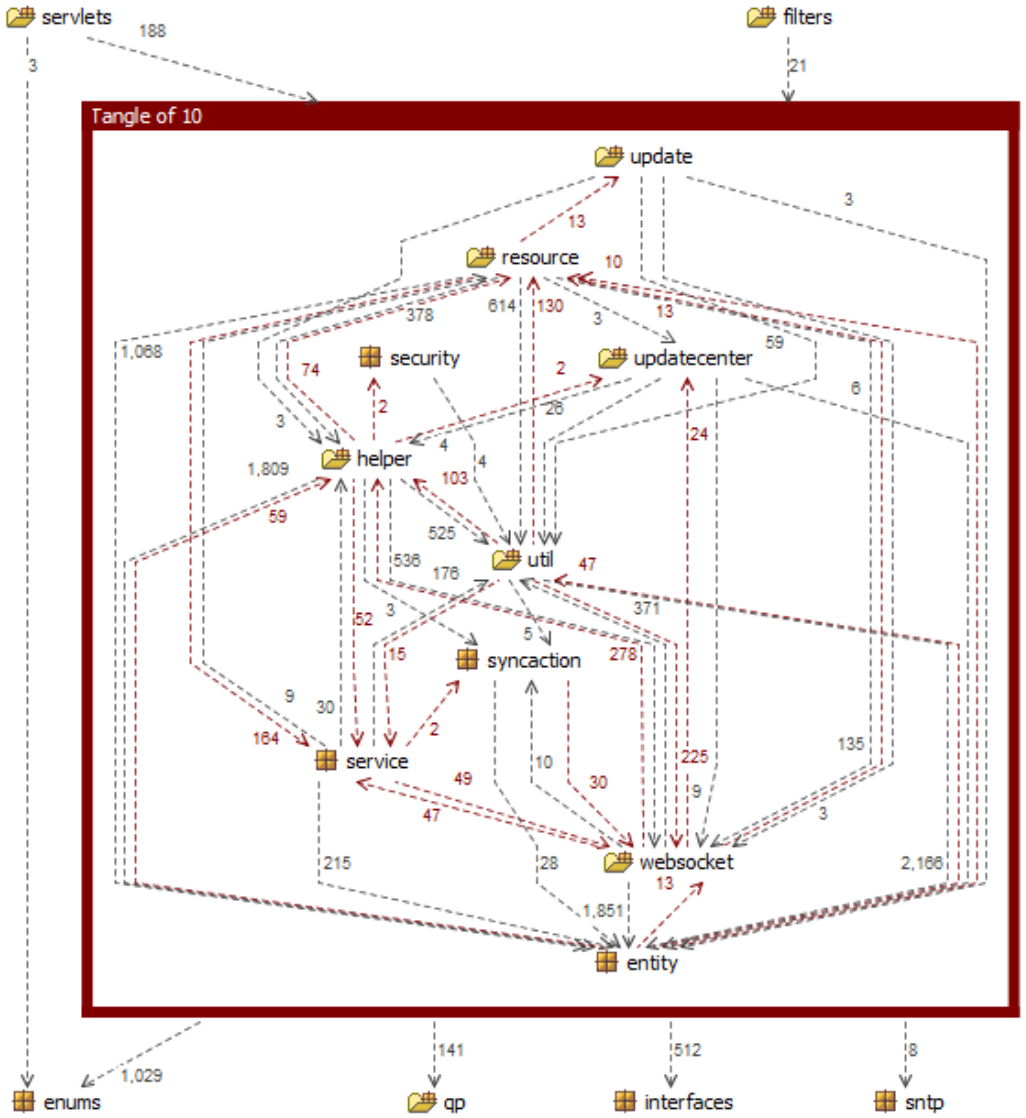


HOW TO BREAK THE MONOLITH?

A Real World Example

SYSTEM ISSUES

- **Cyclic** dependencies
- **No component** structure
- **Meaningless** names



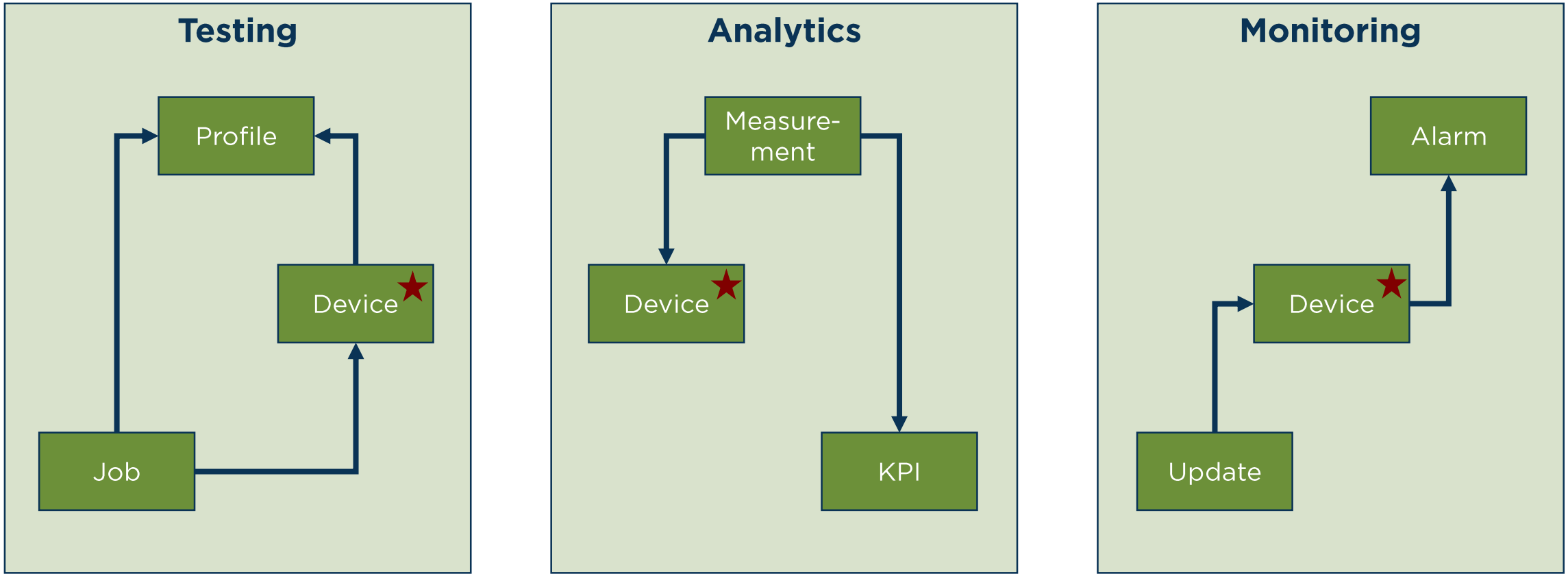
3 STEP REFACTORING

- Step 1: **Define target architecture**
  - Define bounded contexts
  - Introduce pattern language
  - Separate infrastructure code from business code

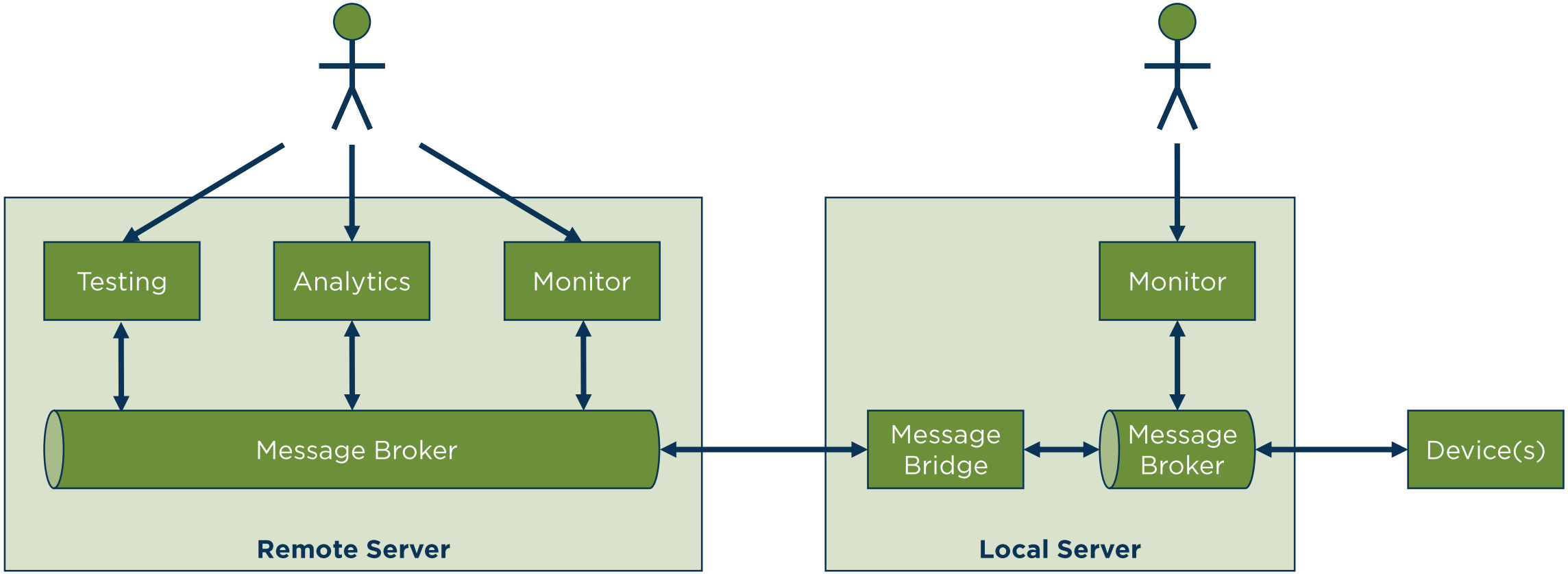
- Step 2: **Move existing code to newly defined subsystems**

- Step 3 (optional): **Distribute the subsystems**

DEFINE BOUNDED CONTEXTS

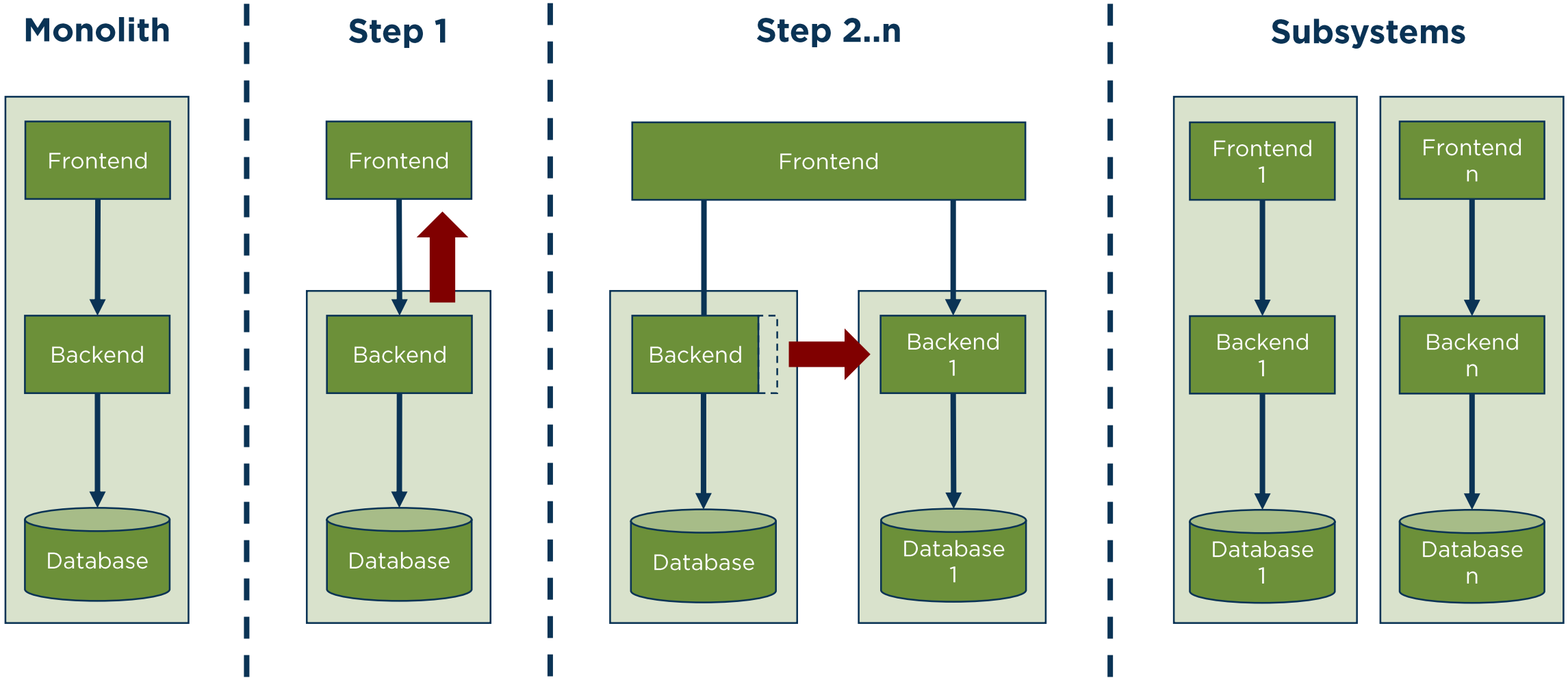


TARGET SYSTEMS ARCHITECTURE



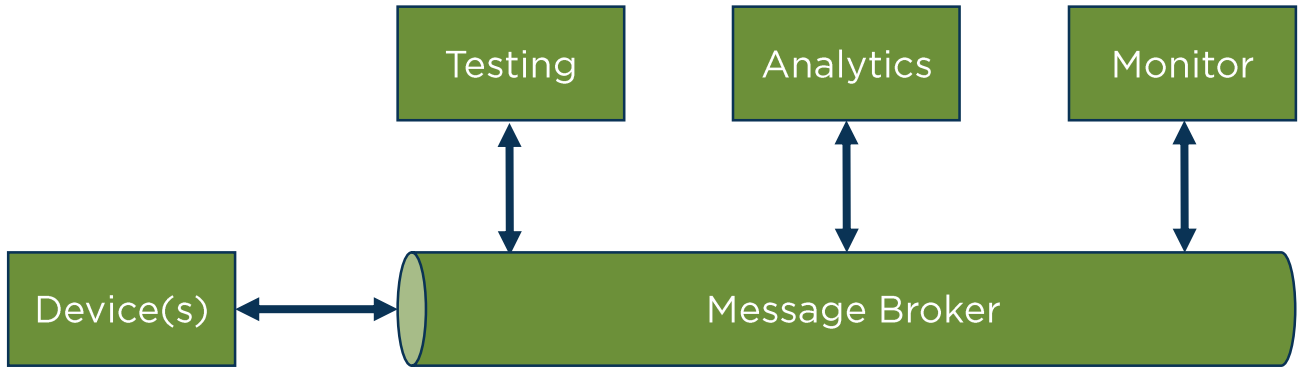
https://en.wikipedia.org/wiki/Strangler_fig

MIGRATION SCENARIO (AKA STRANGLER PATTERN)



DECOUPLING THROUGH MESSAGING

- Messaging **enables modularity**
- Messaging **enforces independence**
- **Events** not Request/Response



THE REAL VALUES

- A **microservices** architecture leads to **modularity**
- Developers are **forced** to follow the structure
- It's **easier to replace** a subsystem instead of the whole system in the future

Think twice before you build a microservice based big ball of mud!

Thank you!

72 Services LLC

Moosentli 7, 3235 Erlach

Switzerland

+41 32 544 8888

info@72.services

72.services