MICRO-SERVICES

G. Molines

2017-2018

Université Nice Sophia Antipolis

POLYTECH NICE-SOPHIA

WHAT IS IT?

Monolith

- Eg: J2EE
- X dev teams in charge of a component
- Components aggregated (“packaged and deployed”) into single bundle, which gets deployed
  - Concept of “entreprise archive”
- Then integration-tested, then ready

SOA

- X dev teams in charge of a component
- Components deployed together into platform
- Then the whole is integration-tested, then declared ready

Micro-services

- X dev teams in charge of a component
  - Dev, test, deploy, operation, support, etc.
- Components deployed independently into platform, then declared ready

Notion of component

- Monolith
  - A back-end service, a chunk of UI
  - Integration done at build time
- SOA
  - Back-end service (UI is a client's concern)
  - Integration done at config time
- MS
  - Component = backend + storage + frontend
  - Integration done at run time

What is Micro-service architecture?

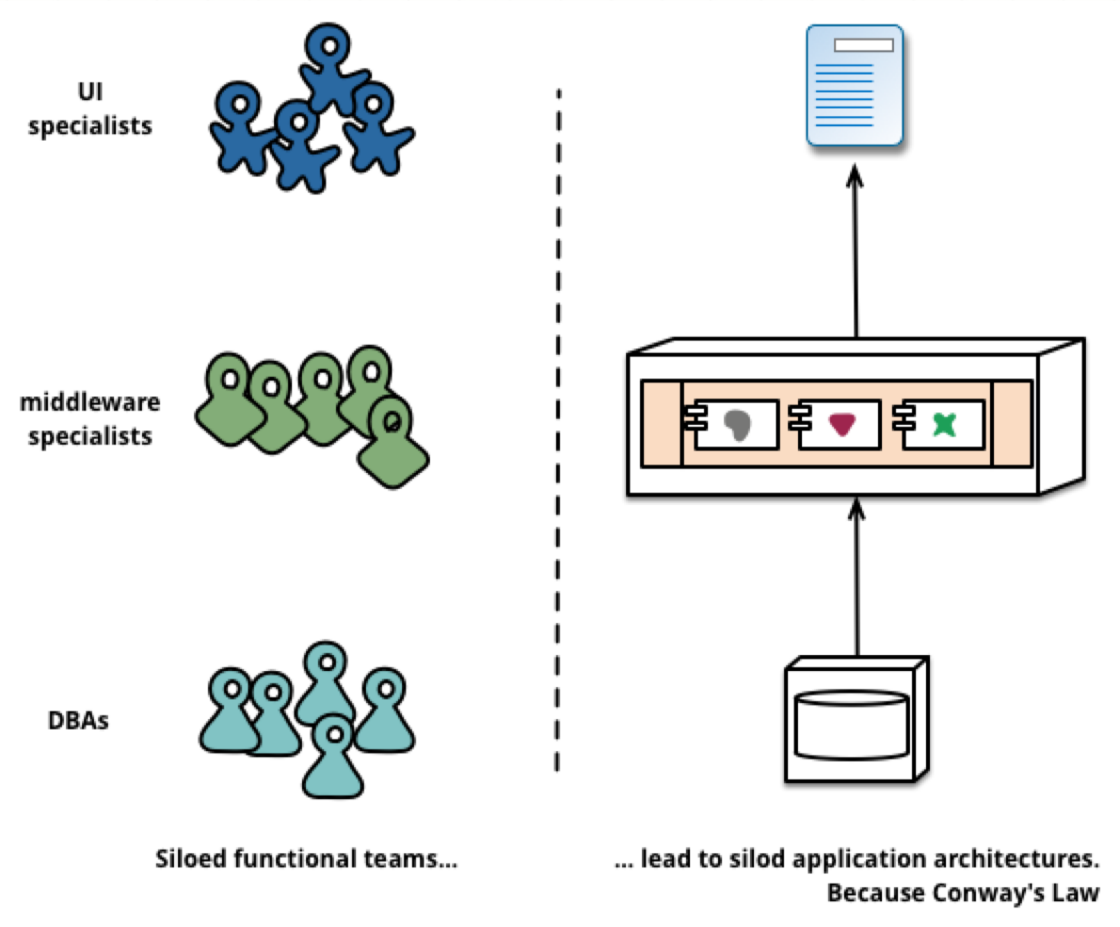
- An app = a suite of very small services
- Each service also includes its deployment capability
  - To some kind of lightweight container
  - With simple communication channels
- Independently deployable
- 1 service = 1 business capability

How?

Conway's law

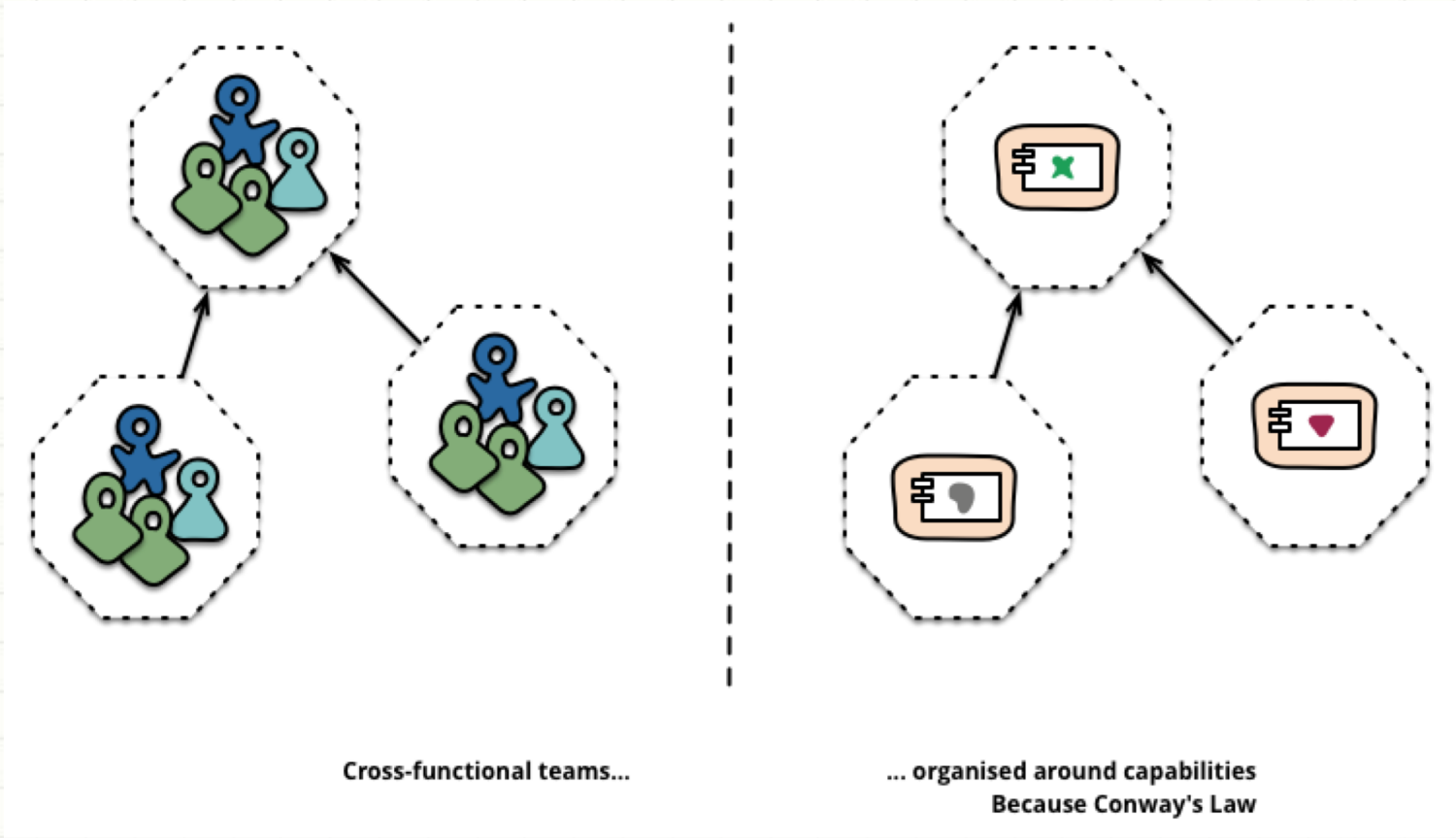
"Any organization that designs a system will inevitably produce a design whose structure is a copy of the organization's communication structure" Melvin Conway.

Conway’s law



© Martin Fowler

Conway’s law



© Martin Fowler

Considerations

- How big? Two pizzas rule
- DevOps style (you own both build and operations)
- Can be viewed as SOA 2.0
- Difference with classic SOA: independently deployable
- And also: a cool buzzword

Features

- Stateless
  - No sticky sessions
- Interface contract
  - Supports different languages

Transactions

- No distributed transaction support
  - Retry pattern
  - Circuit Breaker pattern
  - Rollback half-successful state yourself
- → painful….

Resiliency

- Netflix’ Chaos Monkey
- Co-existence of versions

Scaling

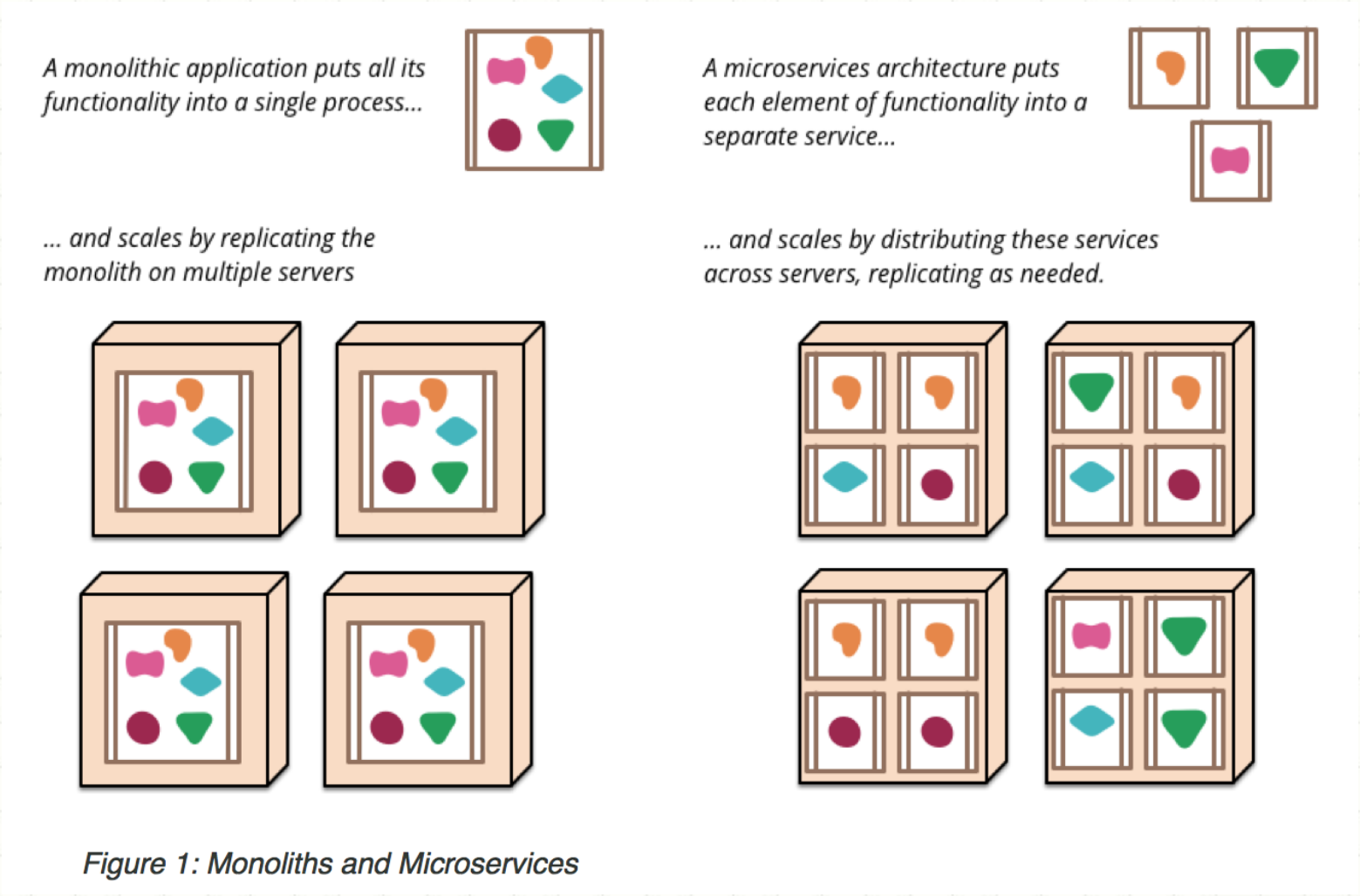


Figure 1: Monoliths and Microservices

Source: Martin Fowler

Pros / Cons of Monolith

- + Simple
- + Fast (in process calls)
- + Resource optimization
- + Scaling up

- - Modularity on paper
- - Team coordination
- - Atomic deployment
- - Hard to scale out

Pros / Cons of MicroServices

- + Easy to update
- + Easy to scale up and out
- + Automation, team ownership

- - management of data / storage
- - code your own transversal services

LIMITS

Anti-pattern - Macroservices

- Distinct functions in one service
- For the wrong reason: code reuse

Anti-pattern - Nanoservices

- Too thin
- High latency
- Complex integration
  - → is it really a service or a library?

Limits

- Team organization, new way to work
- Transition from perfectly working monolith solution
- Cost
- Why build scalability from the get-go?

QUESTIONS?