Microservices
Smaller is Better?

Eberhard Wolff
Freelance consultant & trainer
http://ewolff.com

micro:hg 2015

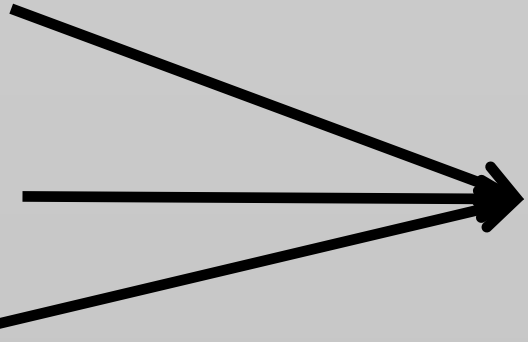
Why Microservices?

Why Microservices?

Strong modularization

Replaceability

Small units



Sustainable Development speed

Continuous Delivery

Deployment less risky

Free Choice of technology

Microservice

=

Micro?

Smaller modules better

10-100LOC

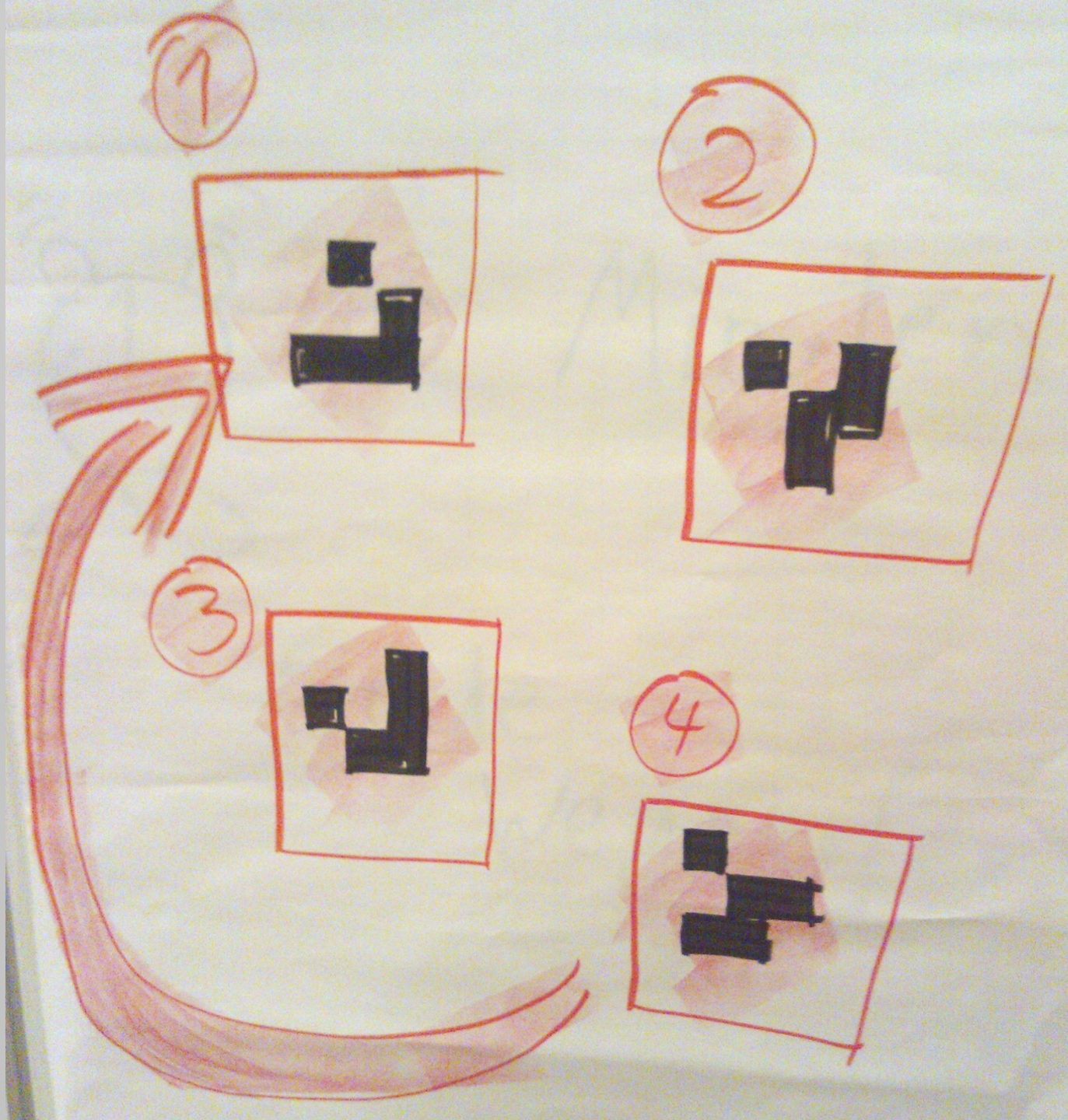
http://yobriefca.se/blog/2013/04/28/micro-service-architecture/

Smaller modules better

10-100LOC

http://vobriefca.se/blog/2013/04/28/micro-service-architecture/

Game of Life



Game of Life in one line of APL

```
life←{ ⍝ John Conway's "Game of Life".
     ↑1 ⍵∨.∧3 4=+/,¯1 0 1∘.⊖¯1 0 1∘.⌽⊂⍵
  ⍝ Expression for next generation.
  }
```

LOC is really a bad metric

dyalog.com

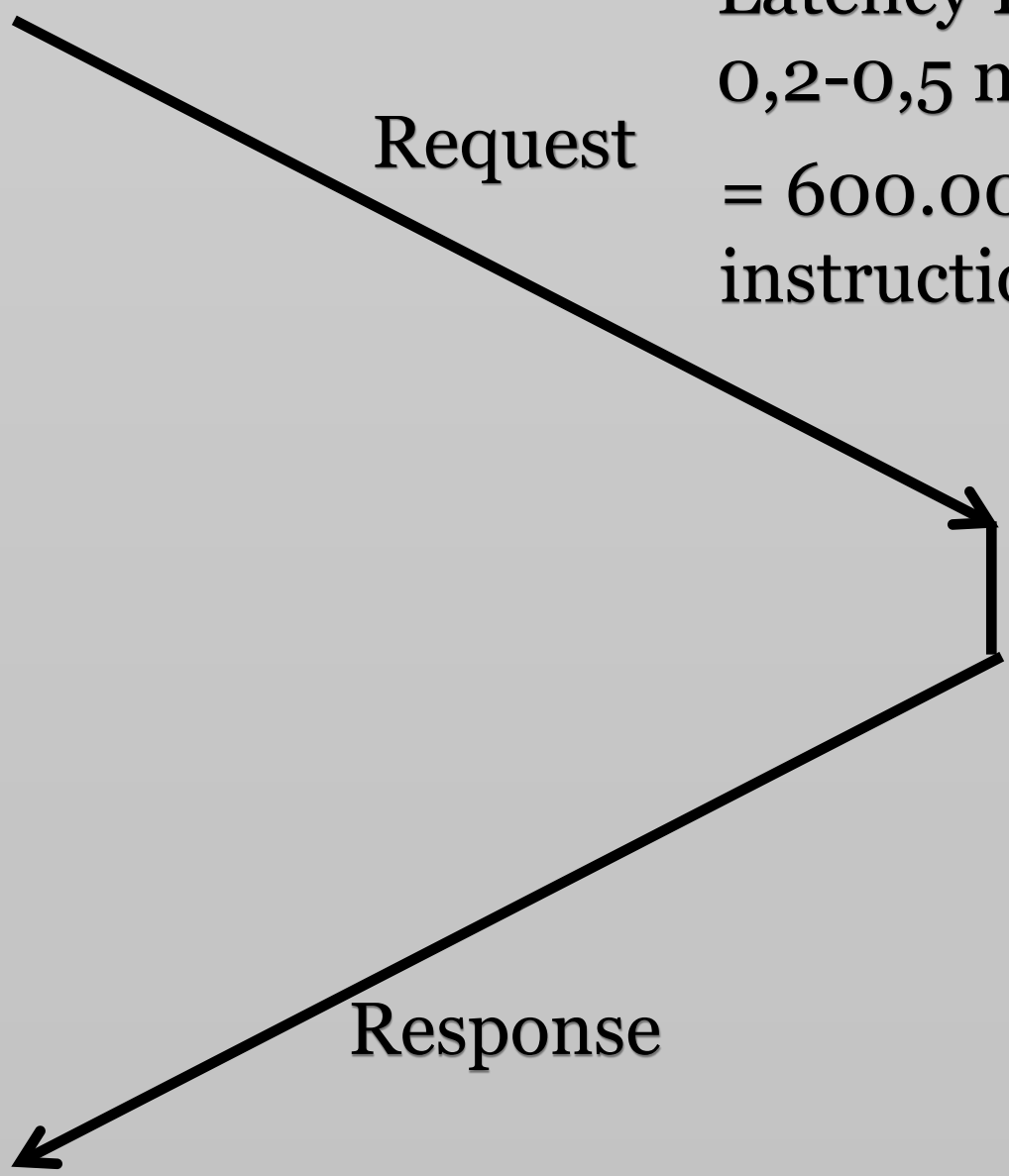
Larger?

- Microservices have an overhead
- Build & deployment pipeline
- Version control

1st Law of Distributed Objects

- Don’t Distribute Your Objects!
- Too much remote communication & overhead
- Lesson learned from CORBA etc

- http://martinfowler.com/bliki/FirstLaw.html



Latency Round Trip:
0,2-0,5 ms

= 600.000-1.500.000 instructions@3GHz

1st Law of Distributed Objects & Microservices

- Small Microservices = lot of communication
- Violates the 1st Law
- Seems to work, though

- http://martinfowler.com/articles/distributed-objects-microservices.html

Too small = too much communication

Again: Why Microservices?

The main reason

Business relevant

How to scale agile?

Implement more feature

Conways Law

microxchg 2015

The Law That I Must Not Name

Architecture
copies
communication structures
of the organization

Conway's Law as an Enabler

- Desired architecture = project structure
- Microservices belong to one team
- Team should be responsible for meaningful features
- Ideal: Independent features

Each team can build and deploy features independently!

Microservices must provide a sensible set of functionality

Conway's Law & Microservice Size

- Upper limit: What a (small) team can handle

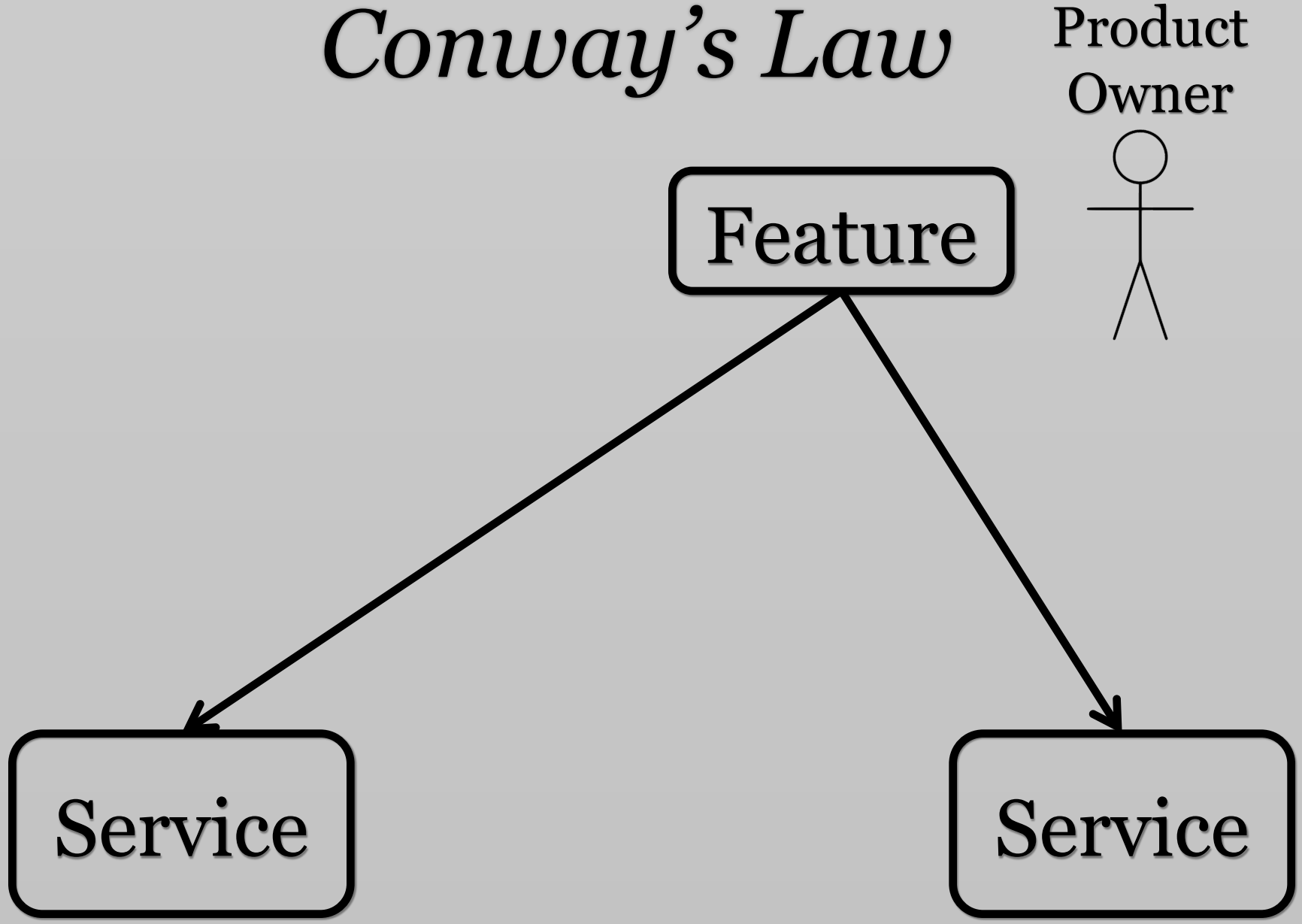
- ...and a meaningful set of features
- Probably not too small
- Lower limit: Depends on overhead / technology

Microservice = Micro?

- Size doesn't matter too much

- Teams must be able to work independently
- Small enough for one team
- Not too much overhead

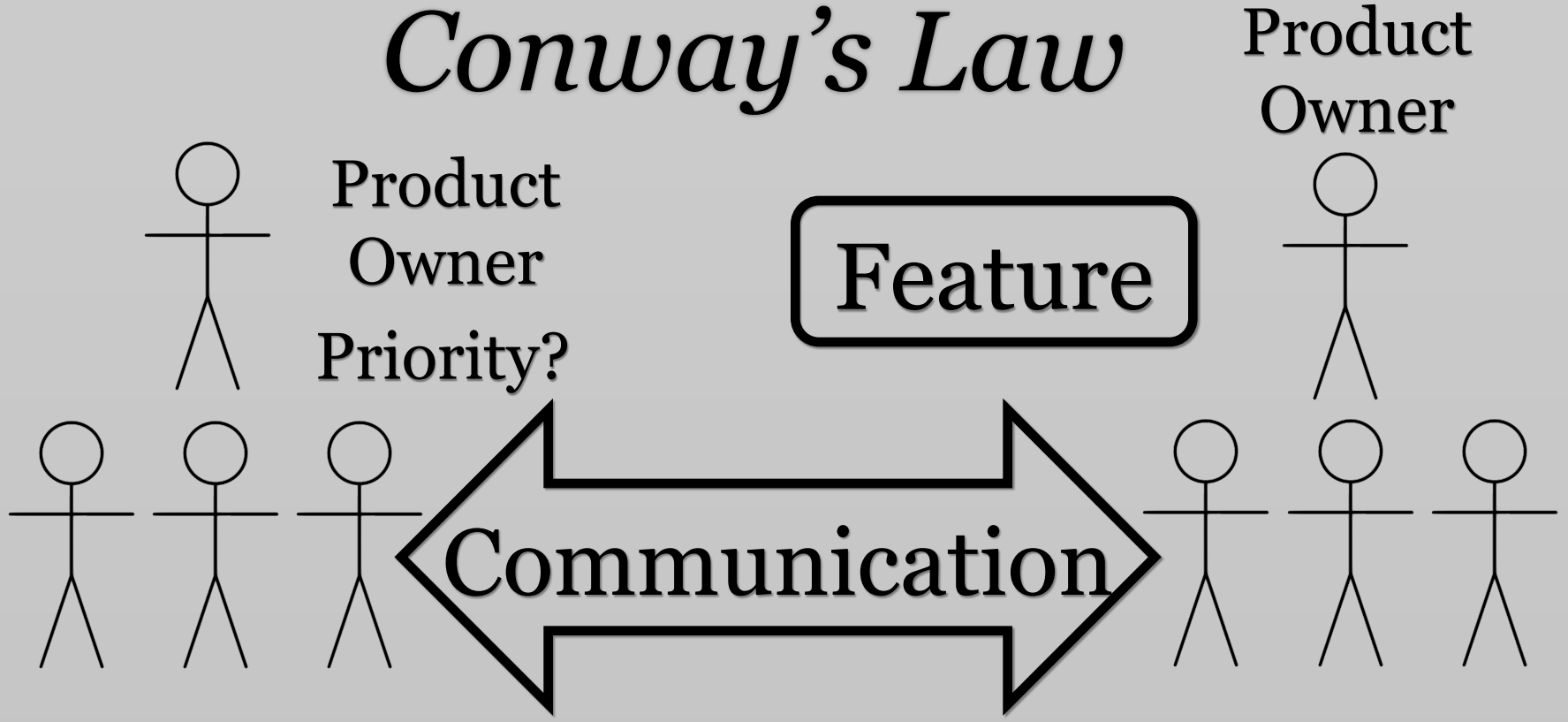
Conway's Law



Bad architecture?

Still can't be avoided!

Conway's Law



Service

Slow

Service

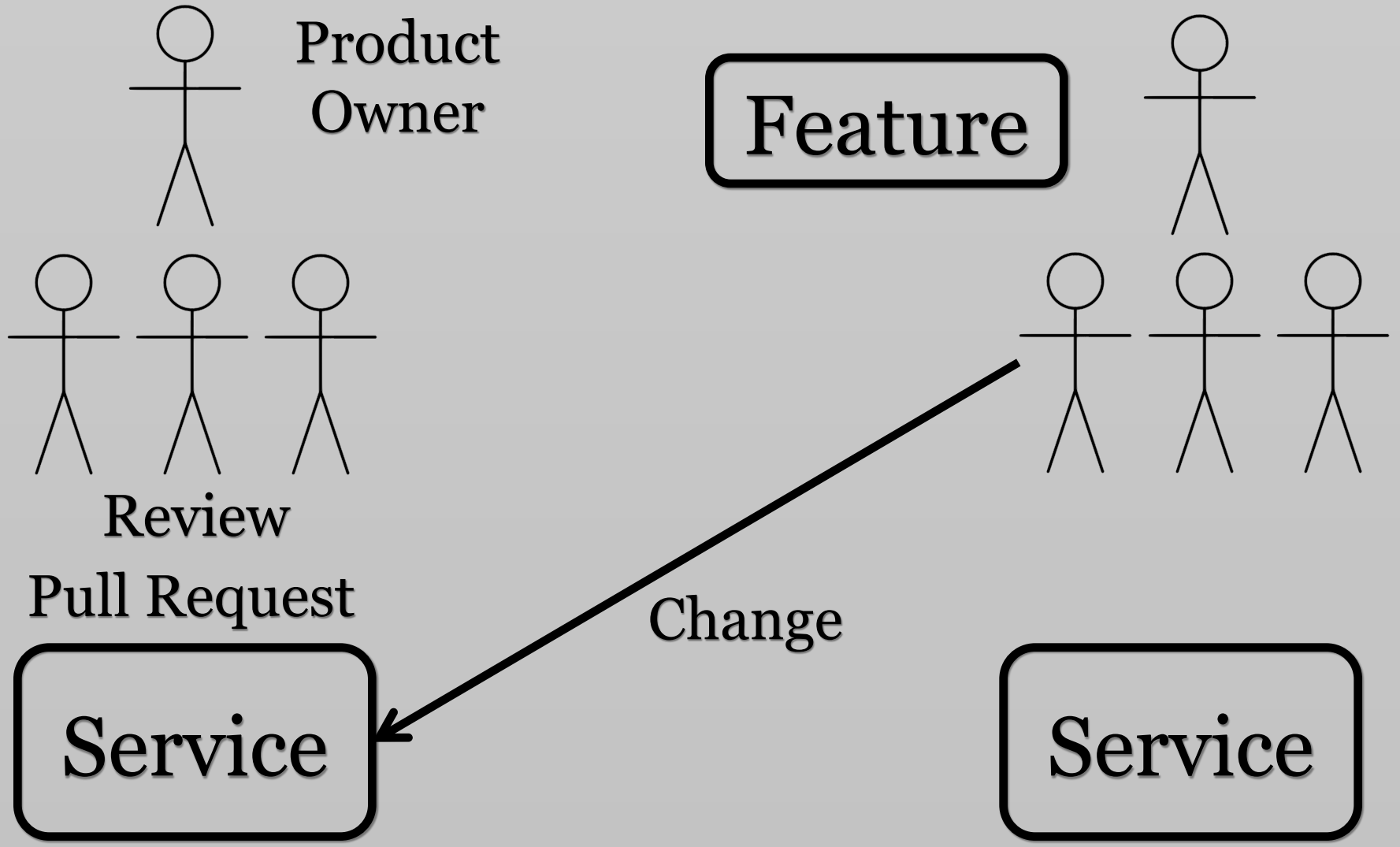
Conway's Law

- Software Interface = Team Communication
- Too much communication if you get the architecture wrong.

- Slows down the process

Collective Code Ownership

Conway's Law


Product Owner
Product Owner
Feature
Review
Pull Request
Change
Service
Service

Microservice & Collective Code Ownership

- Team might change any service
- Reviews can still be done
- …or use Pull Requests

- More devs can work on a services
- Resilience against personnel turnover
- Faster

Microservice & Collective Code Ownership

- Team must understand bigger parts of the code
- Technology freedom?

Refactoring

Refactoring

Different libraries

Different language

Possibly a rewrite

Limited Technology Set

- Easier Refactoring
- Easier to follow standards
  for deployment, monitoring etc
- Easier to implement e.g. resilience

- Netflix uses a lot of
  Java

Refactoring

Releases must be coordinated

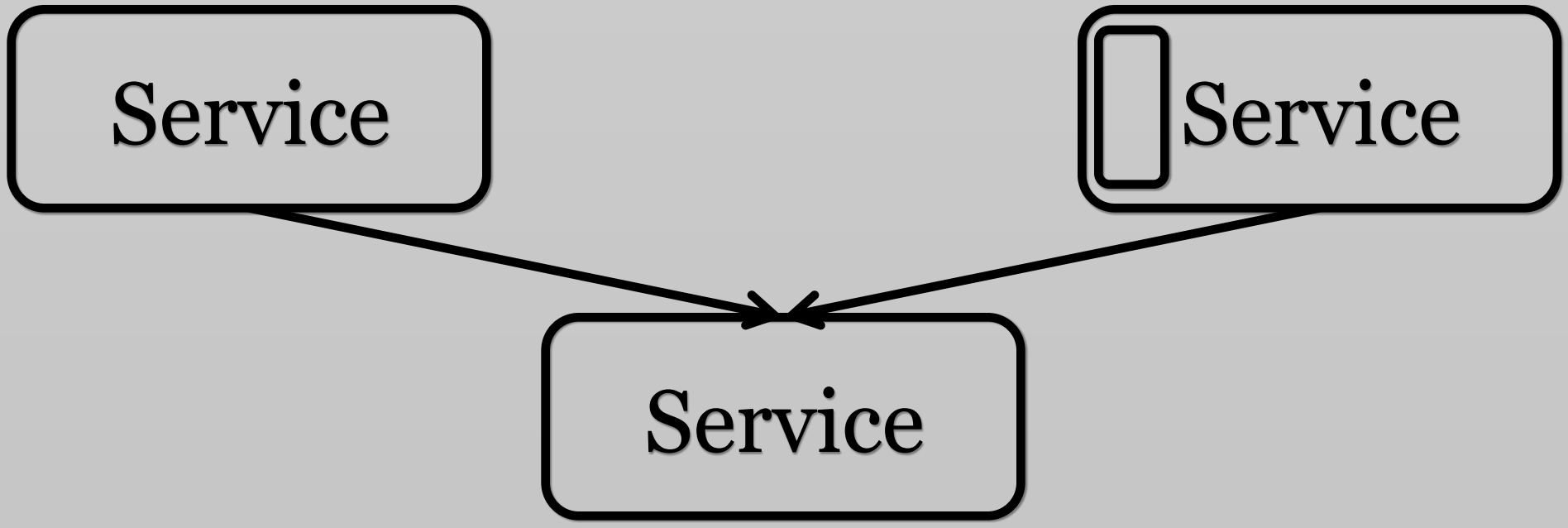
...and we want independent releases

Enforces same language / platform

Hard to implement really reusable code

Like: really, really hard

Refactoring



Remote communication

Slower calls

Unreliable network

Not great

But best option

Number of Services will increase

Refactoring across Services hard

Start BIG

- Conway's law: Upper size = what a team can handle
- Refactoring inside a service easier
- …needed for agile environments
- …where Microservices are used
- Number will increase anyway
- Tear apart easier than join

If You Start Small…

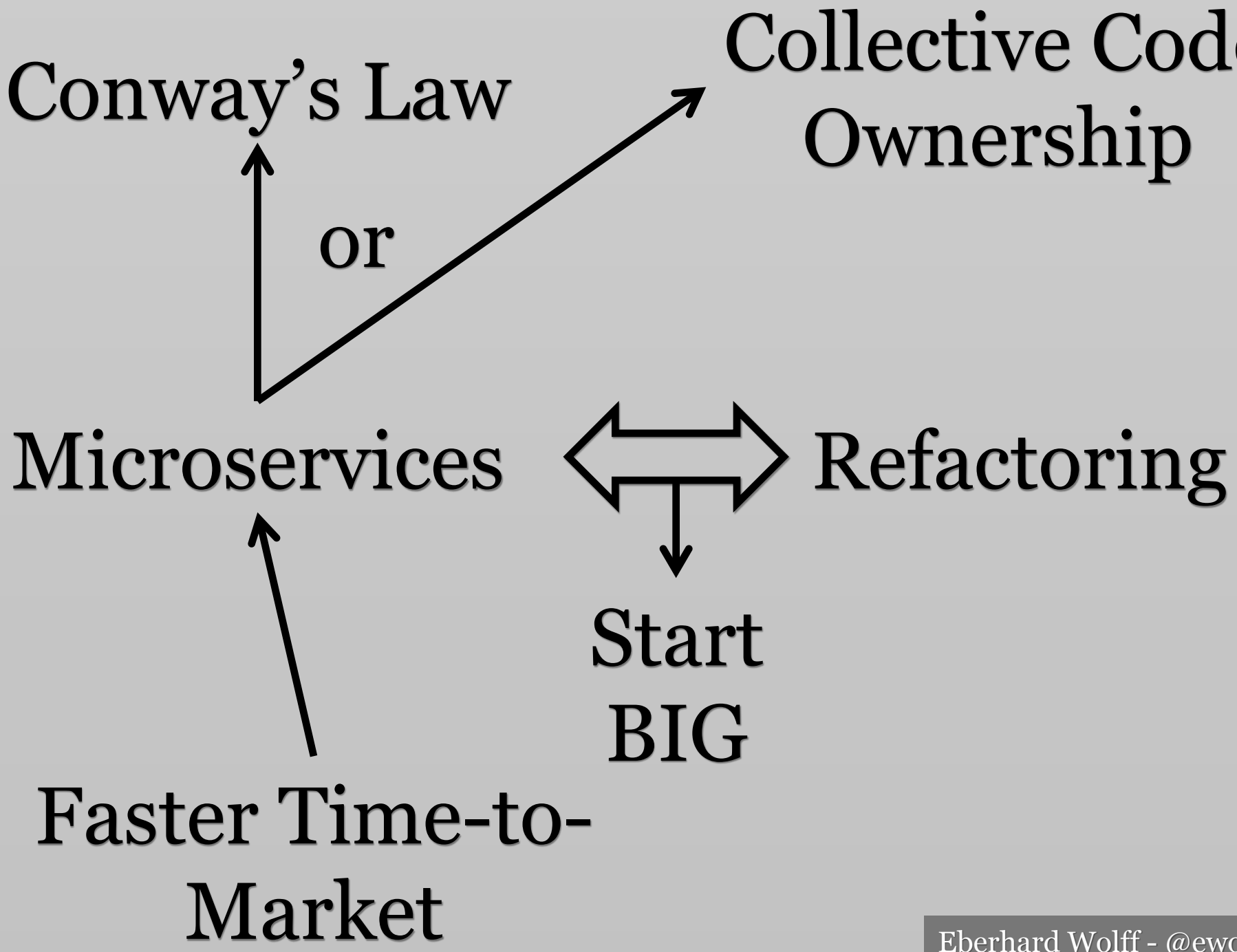
- You will get the architecture wrong
- Functionality hard to move
- Services not too large at the beginning anyway
- Fast deployment still possible

Systems can not be engineered

Systems grow.

Guide growth.

Sum Up


Conway's Law
or
Collective Code Ownership
Microservices
Refactoring
Start BIG
Faster Time-to-Market

Thank You!