Software Architecture Summit, 2019

Architecture, Centralization, Autonomy

Stefan Tilkov
stefan.tilkov@innoq.com
@stilkov

INNOQ

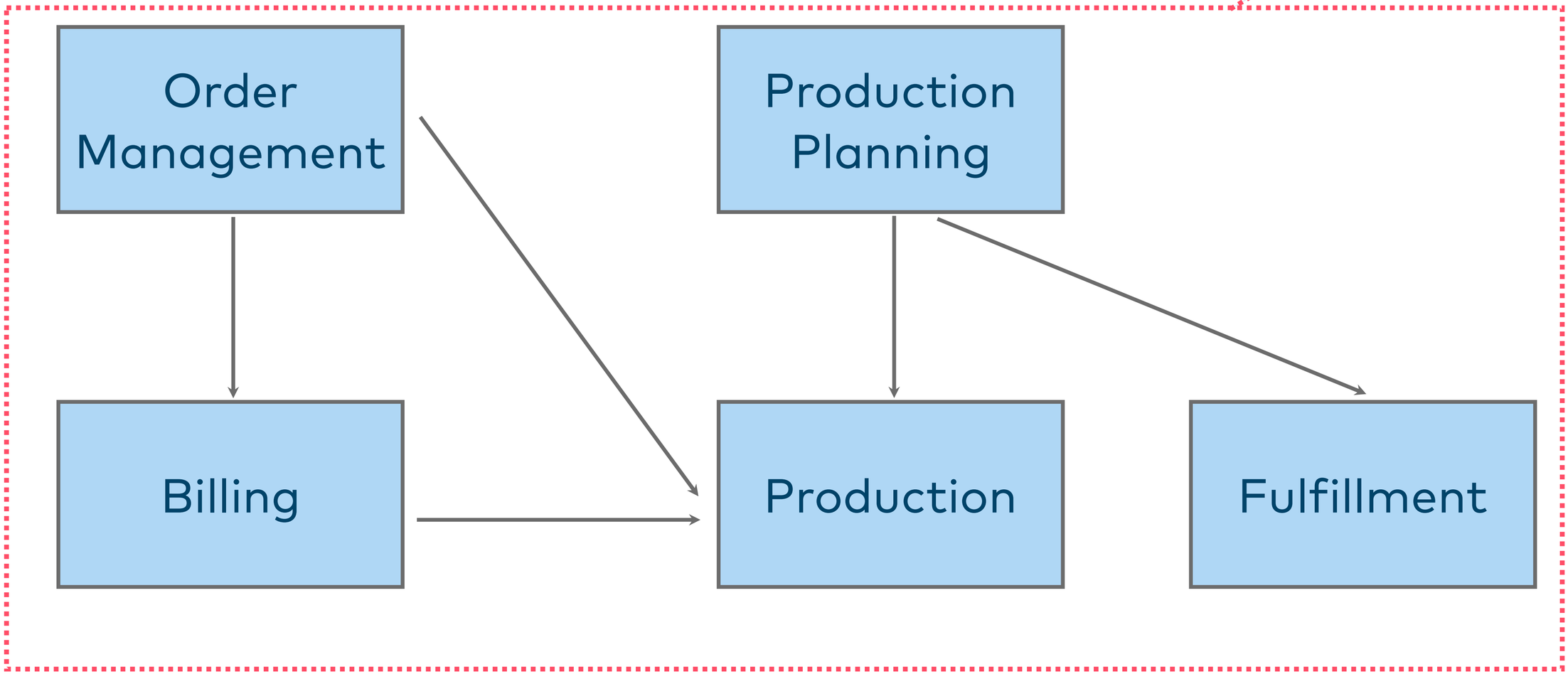
(Software) Architecture Definitions

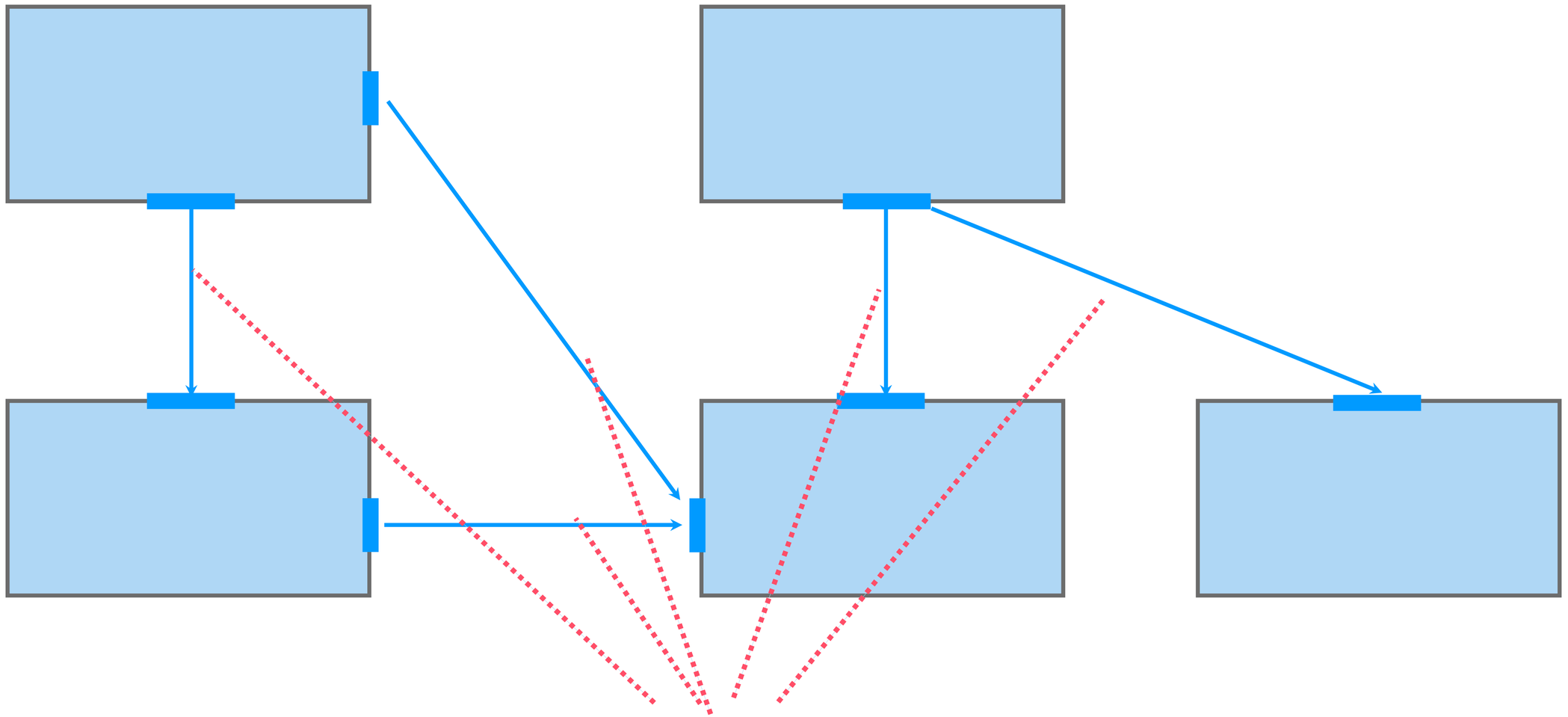
A system's elements, their relationships, and the rules and principles that govern their design and evolution

Whatever the architect considers important enough to merit their attention

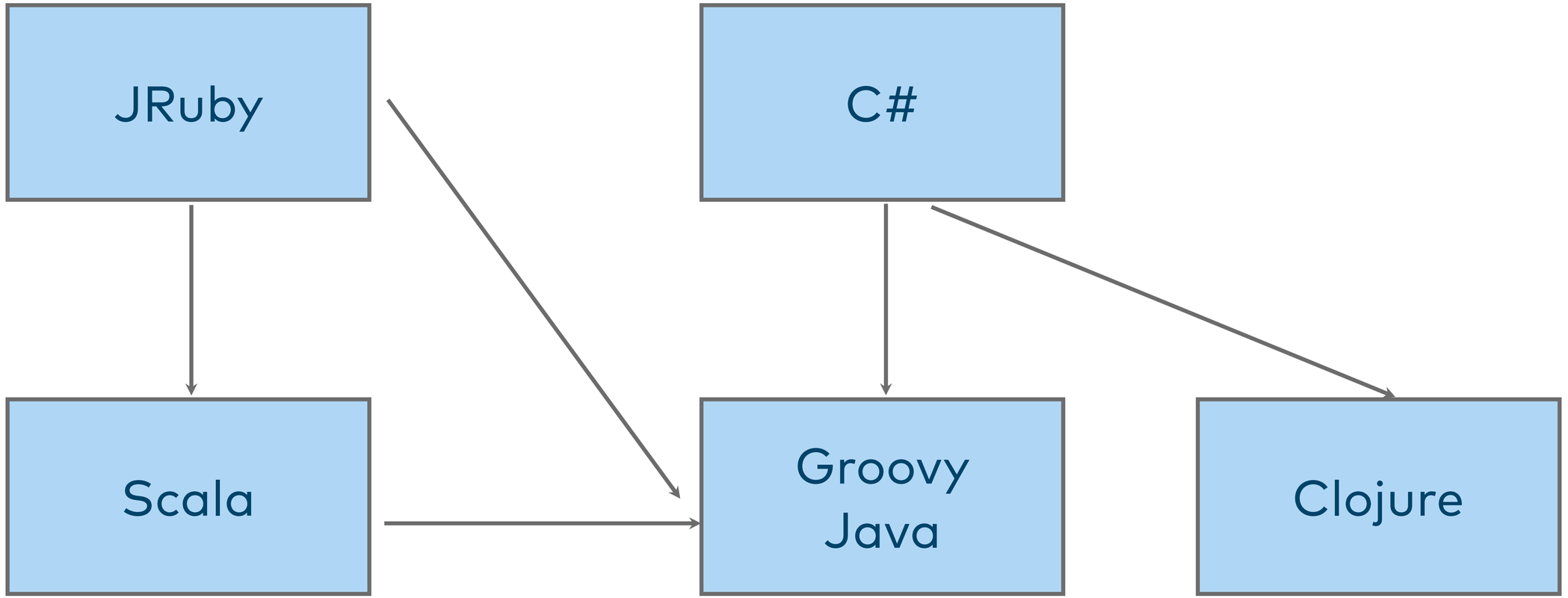
Decisions that you want to be correct because they are costly to change

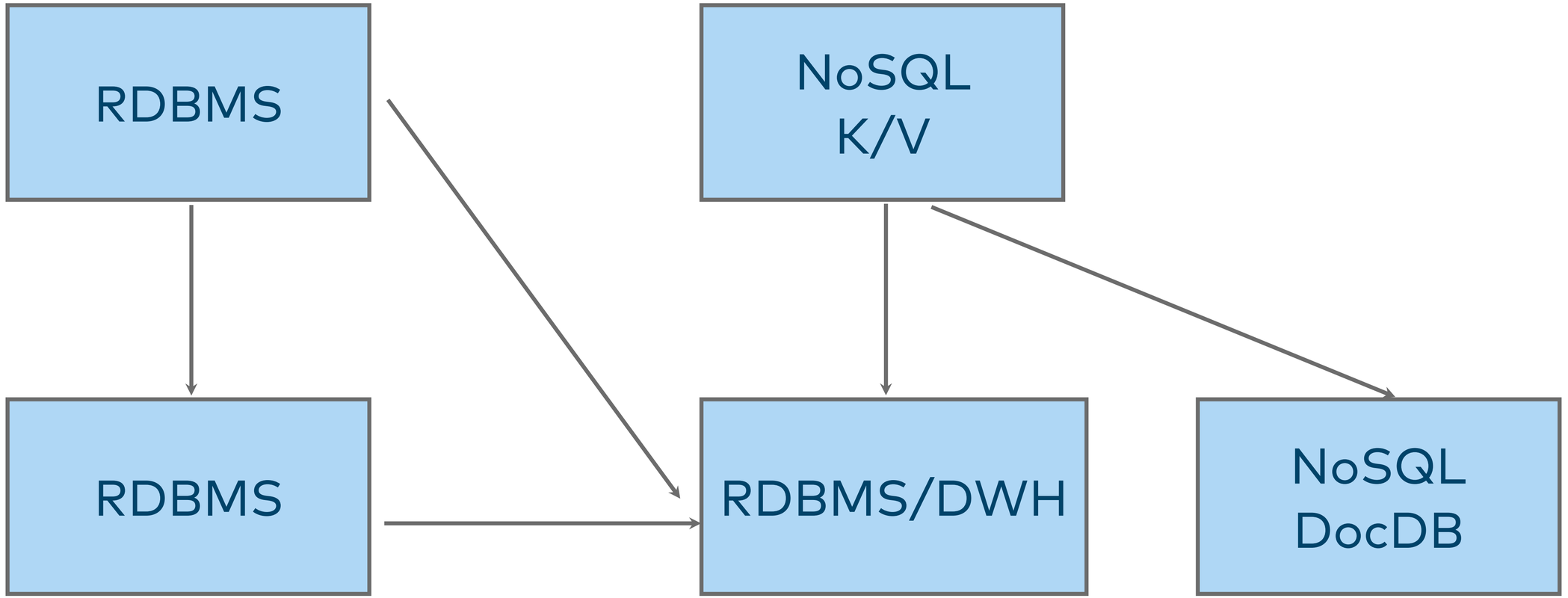
Domain architecture

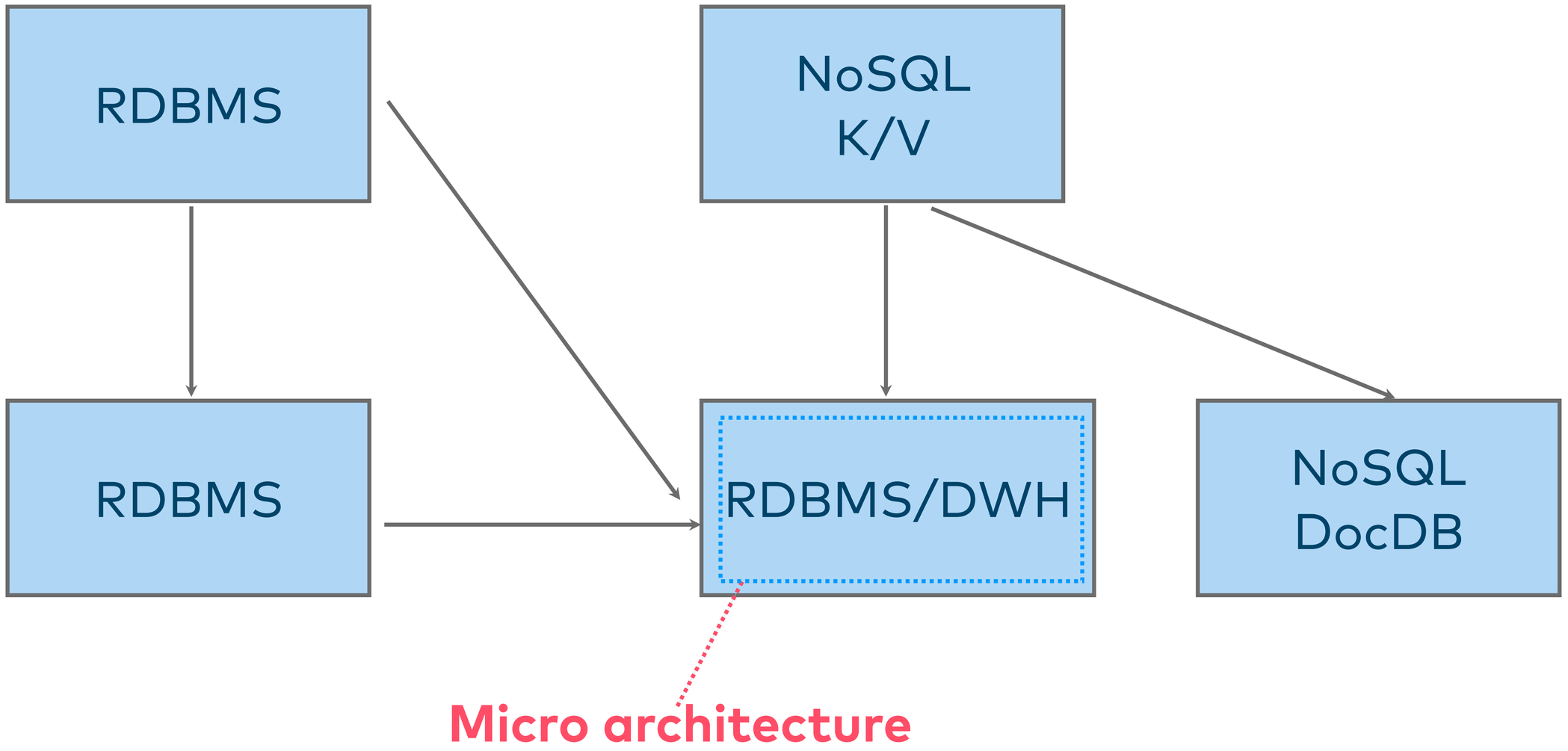




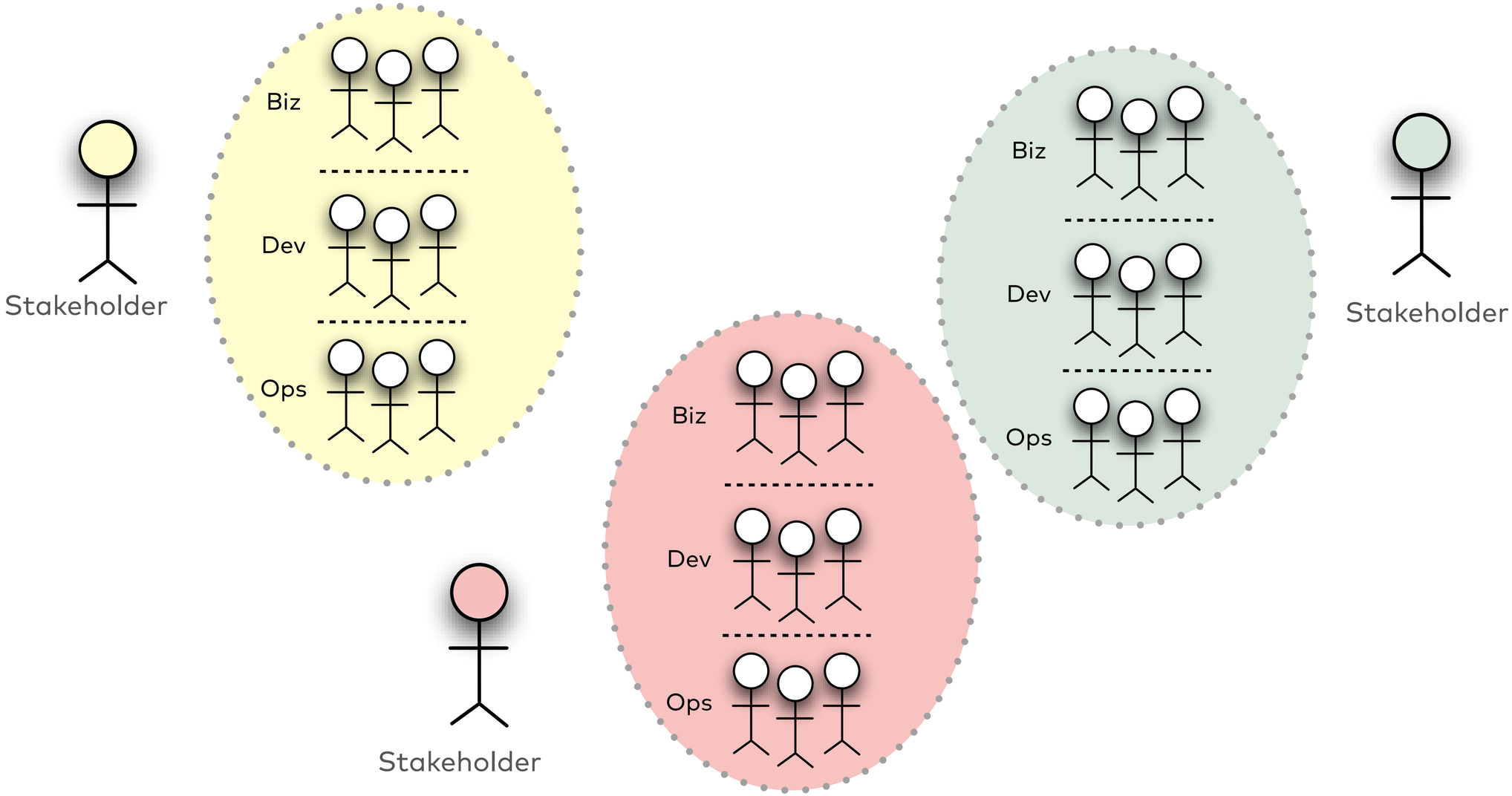
Macro (technical) architecture


JRuby
C#
Scala
Groovy
Java
Clojure

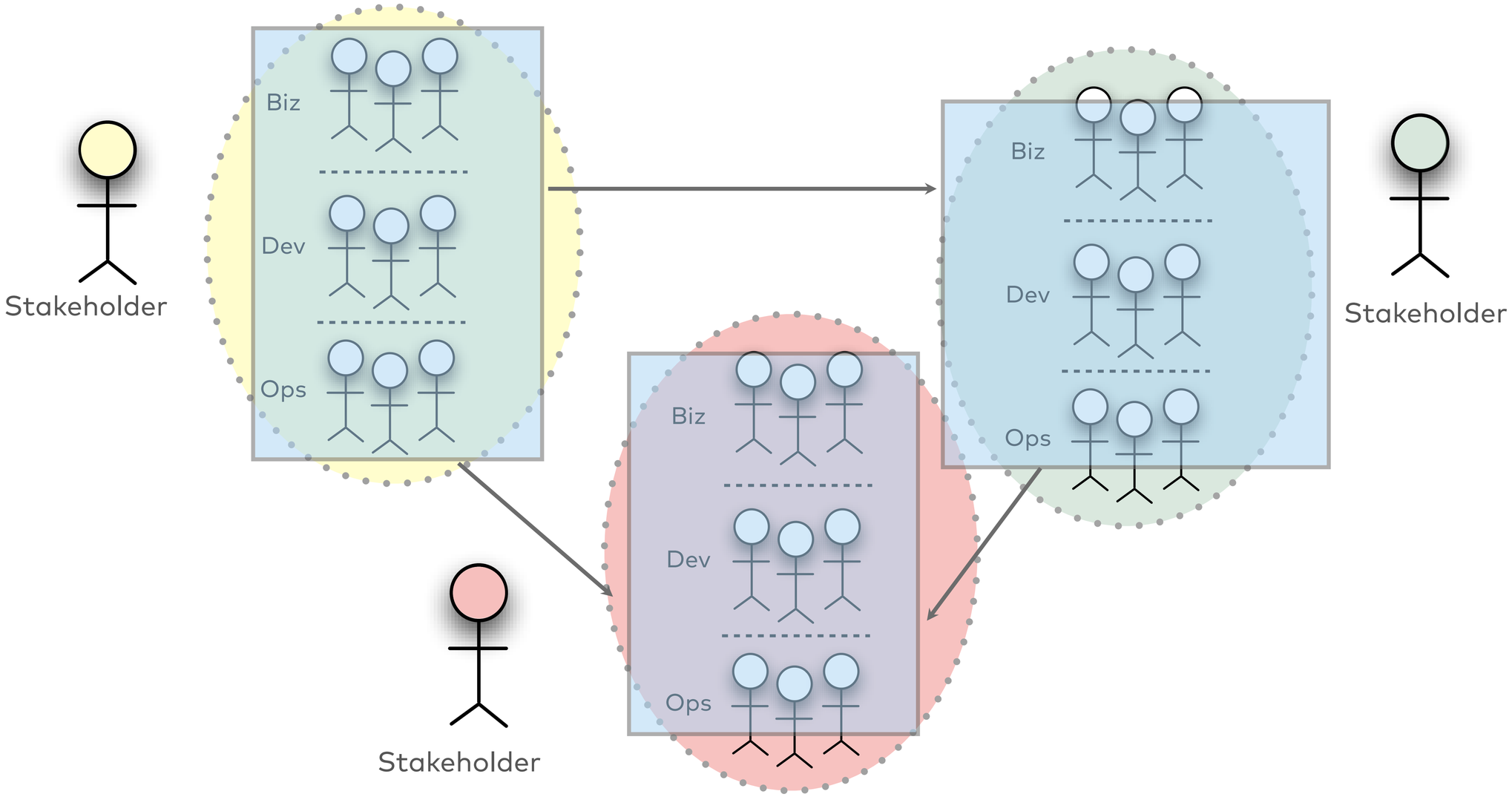

RDBMS
NoSQL
K/V
RDBMS
RDBMS/DWH
NoSQL
DocDB


RDBMS
NoSQL
K/V
RDBMS
RDBMS/DWH
NoSQL
DocDB
Micro architecture

Pattern: Autonomous Cells



Pattern: Autonomous Cells



Why you should centralize everything

Why you should centralize nothing at all

Why autonomous teams rule

Why autonomous teams fail

If your goal is to support autonomous teams, architecture is an essential ingredient

Subsidiarity

Autonomy

Just as it is gravely wrong to take from individuals what they can accomplish by their own initiative and industry and give it to the community, so also it is an injustice and at the same time a grave evil and disturbance of right order to assign to a greater and higher association what lesser and subordinate organizations can do. [...]

The supreme authority of the State ought, therefore, to let subordinate groups handle matters and concerns of lesser importance, which would otherwise dissipate its efforts greatly.

Thereby the State will more freely, powerfully, and effectively do all those things that belong to it alone because it alone can do them: directing, watching, urging, restraining, as occasion requires and necessity demands.

Pope Pius XI, Encyclical Quadragesimo anno, 1931

Centralization

@stilkov

Pattern: Regulated Market

Context:

- …

Observation(s):

- …

Lesson(s) learned:

- …

Context:

- E-Commerce/Online shop (Retail)
- 100-120 developers, ~10 teams

Observation(s):

- Lack of front-end expertise led to central UI/design team, bottleneck for development, deployment, operations, evolution

Lesson(s) learned:

- Local optimization needs can trigger centralization
- Full stack teams require full stack capabilities

A general lack of specific skills, combined with a select few who have it, will sabotage any attempt at decentralizing anything requiring it

Context:

- E-Commerce/Online shop (Retail)
- 100-120 developers, ~10 teams

Observation(s):

- Extremely inefficient UI integration runtime due to lack of standardization
- Vast differences in API style, formats, documentation

Lesson(s) learned:

- Complete lack of guidance creates unproductive diversity

You cannot decide to not have an architecture; if you don't actively create it, be prepared to deal with the one that emerges

There's a fine line between diversity (that adds value) and chaos (that doesn't)

Context:

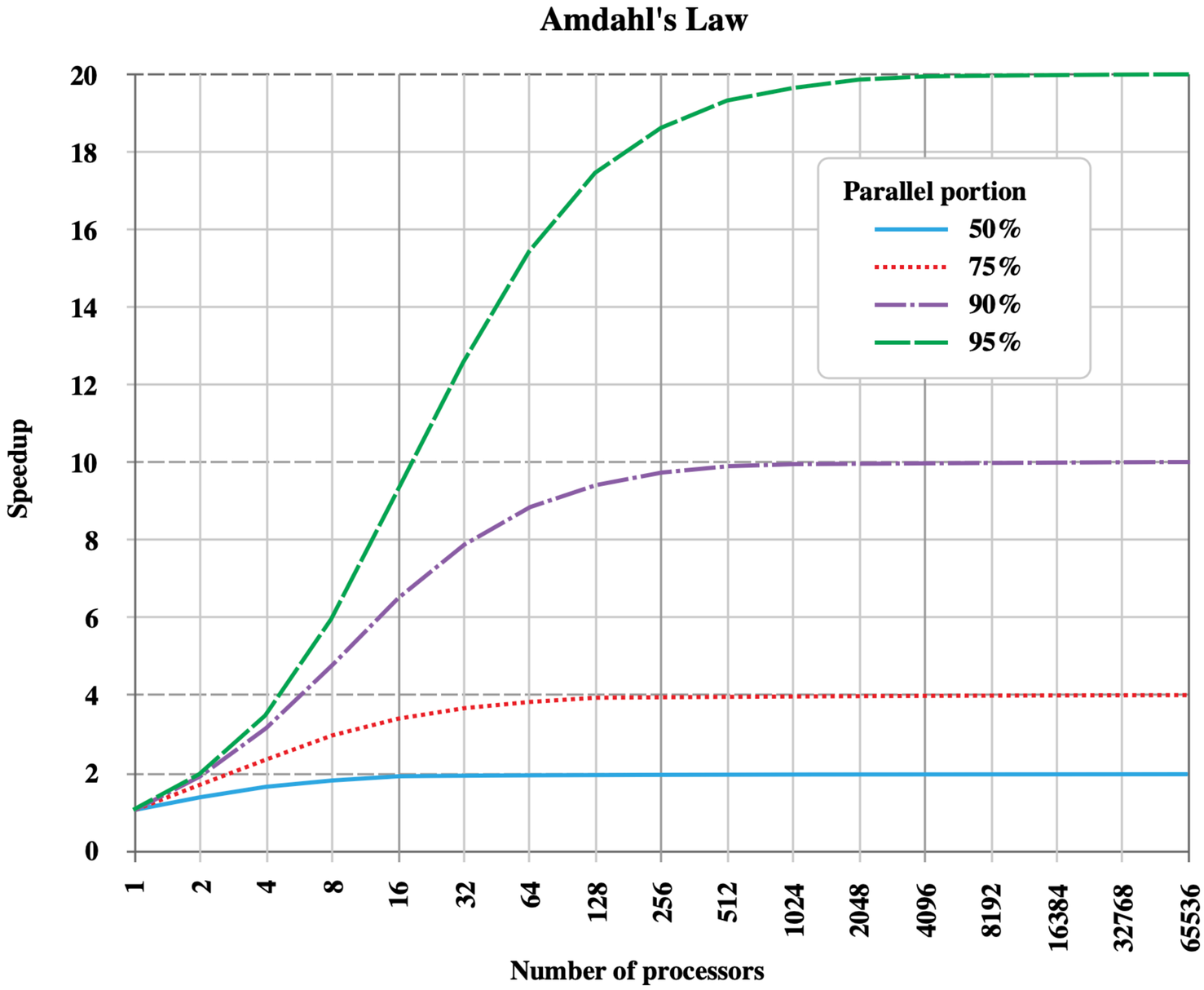
- Insurance customer portal
- 10-15 developers, 1 team

Observation(s):

- Potential for independent decisions in separated systems (almost) never exploited
- Engineering effort spent on coordination

Lesson(s) learned:

- Premature modularization can lead to increased effort without matching benefits



https://en.wikipedia.org/wiki/Amdahl's_law#/media/File:AmdahlsLaw.svg

Amdahl's law for teams

- Threshold set by non-parallelizable part of work
- Adding more teams will not help you if you've reached the threshold

Law of diminishing returns

- Coordination effort increases with # of people/teams
- Returns from re-use possibly far outweighed by extra effort

Context:

- E-Commerce/Online shop (Retail)
- 100-120 developers, ~10 teams

Observation(s):

- Common standard micro architecture at start of project
- Gradual increase in degrees of freedom
- Increase in actual diversity of tools, languages, architecture

Lesson(s) learned:

- Increased maturity allows for less dogma/fewer rules

Start with a common internal (micro) architecture, but allow for separate evolution according to specific needs

Pattern: Marketing-based Governance

Context:

- Global logistics company
- m projects, n teams

Observation(s):

- Inside-out development of rich, multi-faceted, highly functional platform, sophisticated tool support for developing platform applications
- Teams resist perceived proprietary, complex, useless platform
- Ultimate decommissioning of platform after MM€ investment

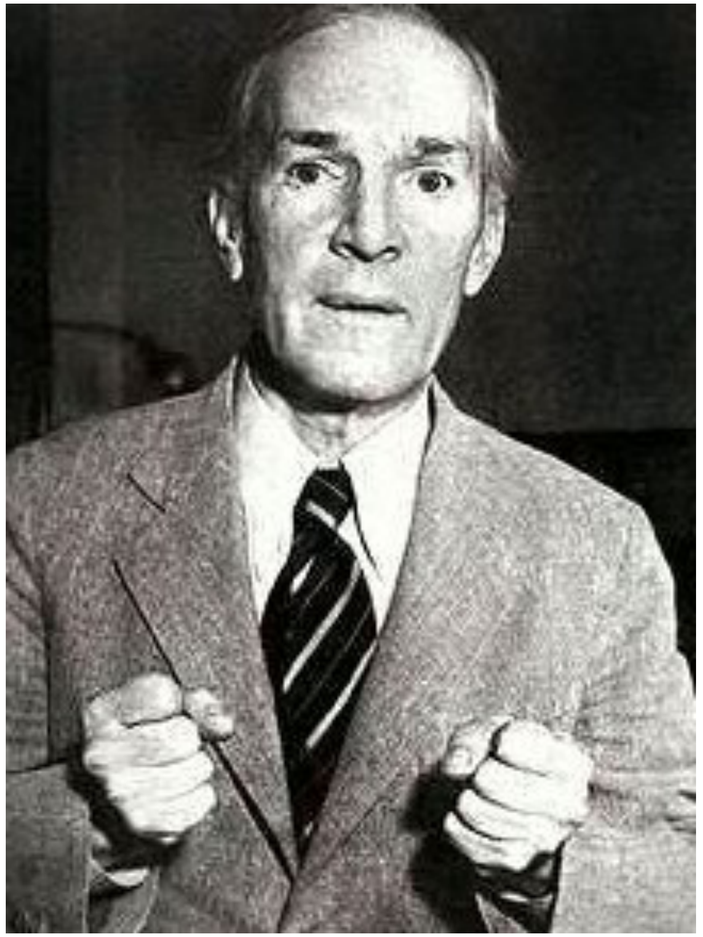
Lesson(s) learned:

- Platform development as high risk activity

Change Resistance

It's difficult to get a man to understand something when his salary depends on his not understanding it.

Upton Sinclair, 1934



Sunk Cost Fallacy

Eating your own dog food is an excellent idea.
If you're a dog.

Context:

- Company-wide digitization effort
- 150-300 developers, 10-15 teams

Observation(s):

- Common standard platform and team to support other teams
- Standardized CI/CD pipeline & runtime platform
- Severe inefficiencies due to one-size-fits-all platform (esp. DB)
- Continuous fighting between teams and platform engineering

Lesson(s) learned:

- Platform teams can take on a significant life of their own

Closed organizational systems will do everything they can to maintain themselves

Closing your system to external influences is a great way to ensure it will suck, eventually

Context:

- E-Commerce marketplace
- 25-75 developers, 5-10 teams

Observation(s):

- Strategic decision to outsource platform to external party (public cloud provider)
- 100% “all-in” strategy (no worries about vendor lock-in)

Lesson(s) learned:

- Significantly decreased emotional attachment to platform
- Underestimated need for platform expertise

Don’t fall in love with your own tools or libraries, maintain a strictly professional relationship

Dreyfus model of skill acquisition

Stage / Quality	Novice	Advanced Beginner	Competence	Proficient	Expert
Recollection	Non-Situational	Situational	Situational	Situational	Situational
Recognition	Decomposed	Decomposed	Holistic	Holistic	Holistic
Decision	Analytical	Analytical	Analytical	Intuitive	Intuitive
Awareness	Monitoring	Monitoring	Monitoring	Monitoring	Absorbed

@stilkov

The more experienced you are at (active and passive) architectural governance, the less you can do of it

Growing architectural maturity means less guidance and rules are needed

Takeaways

1.
Autonomy is the goal
(unless you waste effort without benefit)

2.
Control is tempting
(unless you're the one being controlled)

3.
Letting go is the hardest part
(unless everyone sees benefits)

4.

Decentralization must be managed

(to the degree that's needed to keep it)

5.
Standardization helps
(if it's only mandatory as an exception)

That's all I have.
Thanks for listening!

Stefan Tilkov
@stilkov
stefan.tilkov@innoq.com
Phone: +49 170 471 2625

www.innoq.com

SERVICES

Strategy & technology consulting
Digital business models
Software architecture & development
Digital platforms & infrastructures
Knowledge transfer, coaching & trainings

FACTS

~150 employees
Privately owned
Vendor-independent

OFFICES

Monheim
Berlin
Offenbach
Munich
Hamburg
Zurich

CLIENTS

Finance
Telecommunications
Logistics
E-commerce
Fortune 500
SMBs
Startups

@stilkov