Architecture, Organization, Processes – and Humans

Stefan Tilkov, stefan.tilkov@innoq.com, @stilkov

Software Architecture Summit Berlin, 2017

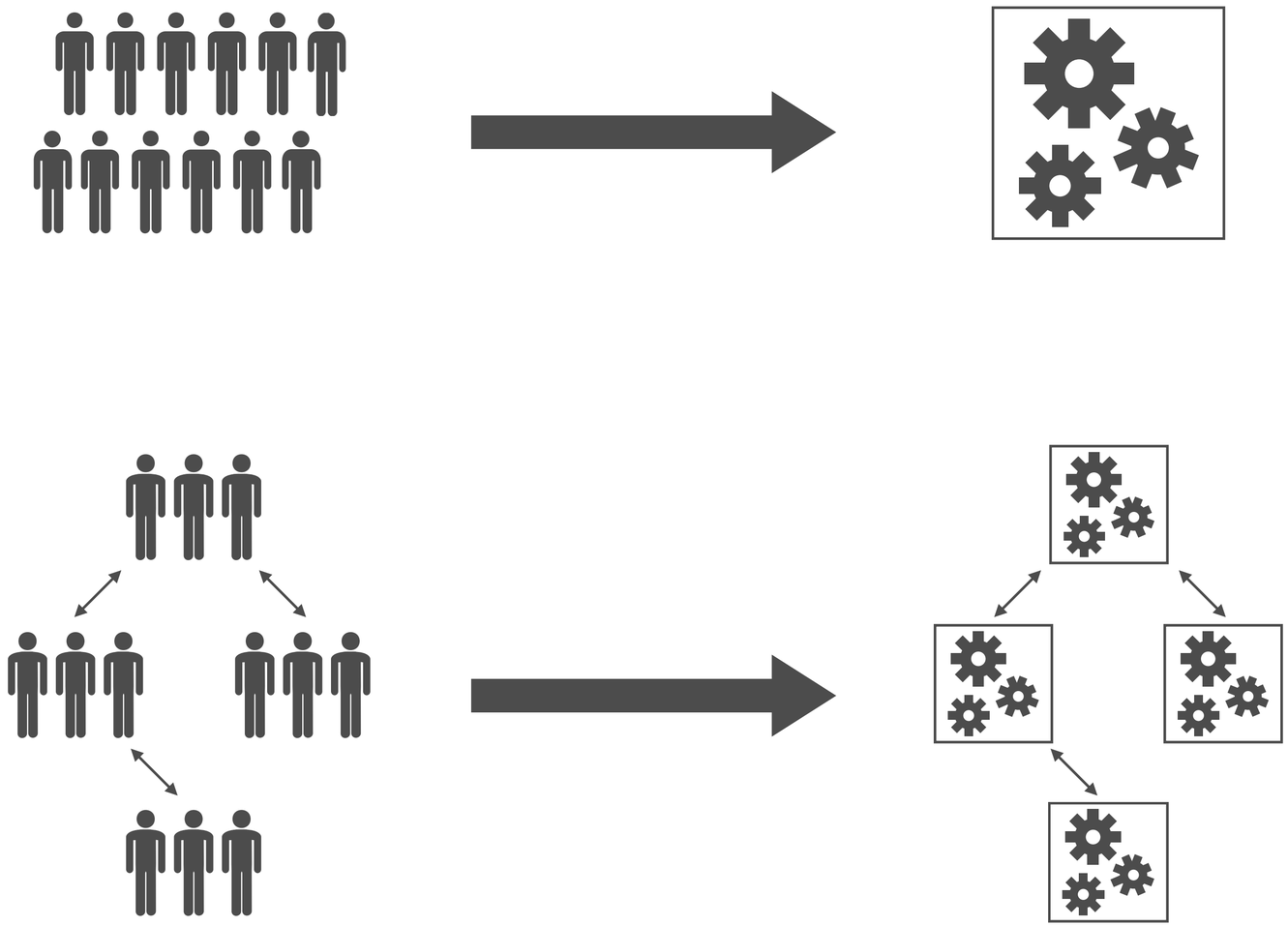
innoQ

Observations

Conway's Law: Organization ⇢ Architecture

"Organizations which design systems are constrained to produce systems which are copies of the communication structures of these organizations." – M.E. Conway

Conway’s Law illustrated



Conway Reversal 1: Organization ⇢ Architecture

Any particular architecture approach constraints organizational options – i.e. makes some organizational models simple and others hard to implement.

Conway Reversal 1: Organization  Architecture

Choosing a particular architecture can be a means of optimizing for a desired organizational structure.

The “Tilkov wants a law, too” slide

The quality of a system’s architecture is inversely proportional to the number of bottlenecks limiting its evolution, development, and operations

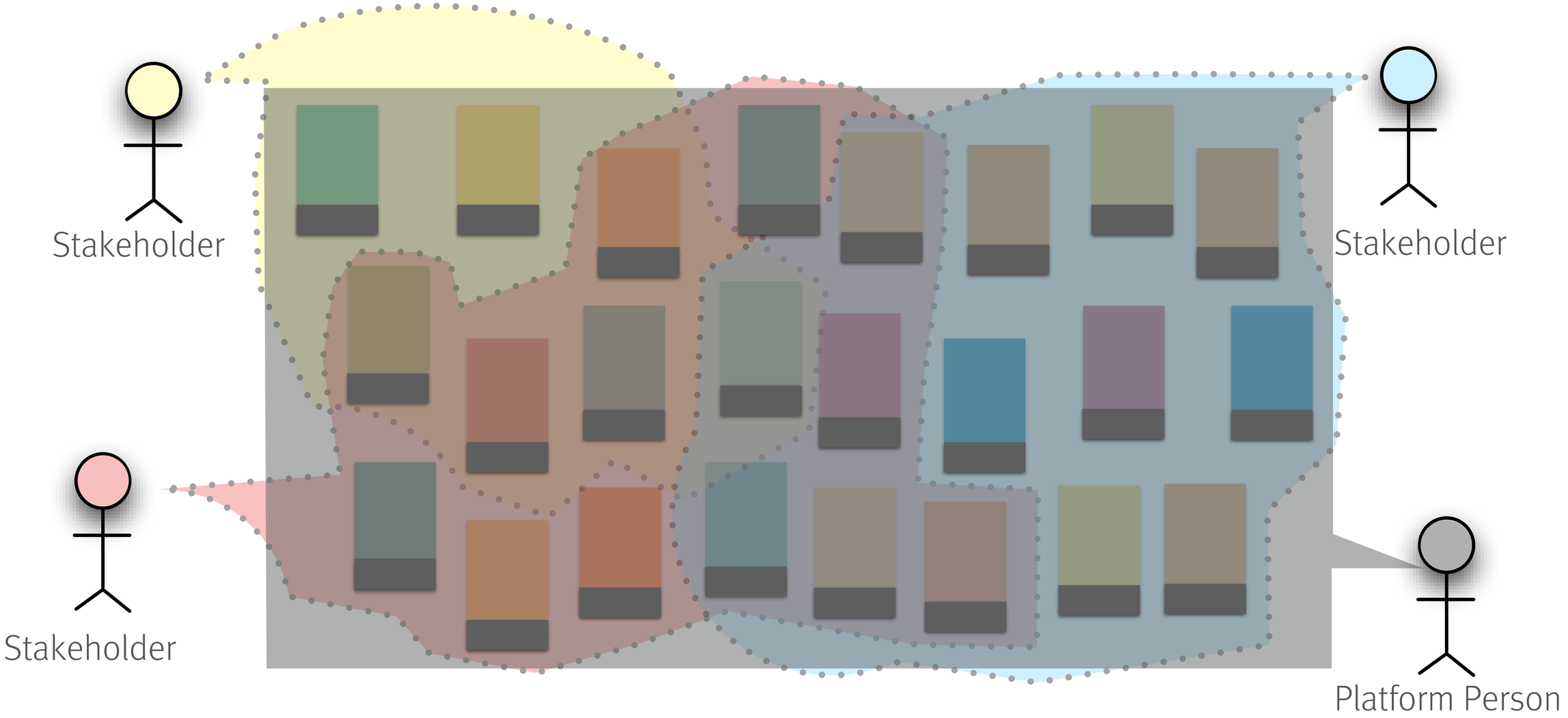
The “Tilkov wants a law, too” slide

(Attempt #2 in case the 1st one doesn’t catch on)

In a digital company, architecture, organization & processes can only evolve together

Antipatterns

Antipattern: Decoupling Illusion



Antipattern: Decoupling Illusion

Description

Technical separation into subsystems/services does not match business domain separation

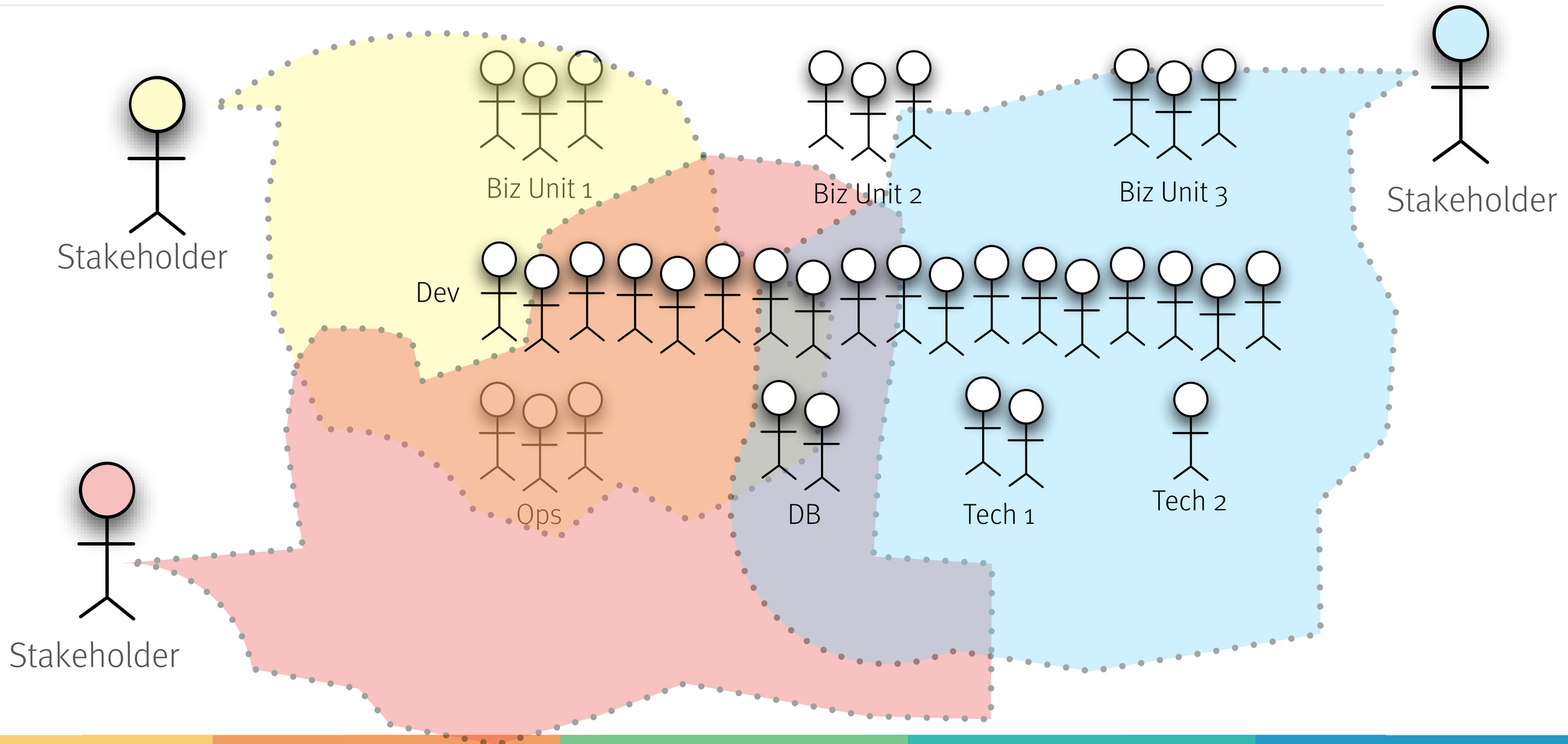
Reasons

- Technical drivers prioritized over business drivers
- Lack of awareness for stakeholder needs
- Reuse driver furthers single platform approach
- Microservices hype

Consequences

- Technical complexity
- Conflicting stakeholder needs require coordination
- Organizational bottlenecks due to centralized components with highly concurrent requests

Antipattern: Domain-last Approach



Antipattern: Domain-last Approach

Description

Major driver for organizational structure is roles and technical capabilities, not business domain

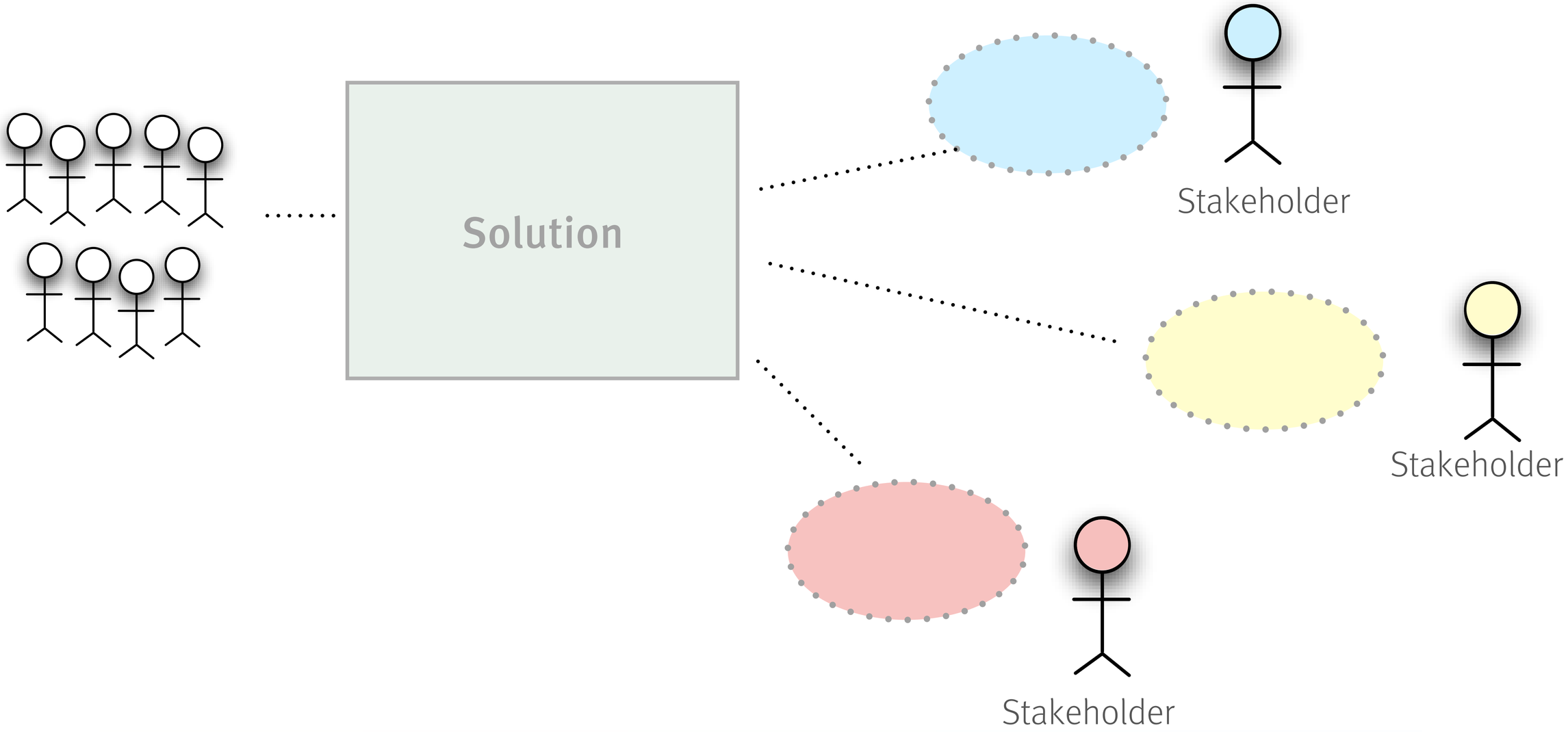
Reasons

- Matches classical company structure
- Division of labor in divisions, department, teams
- Projects as exceptions to change something that works

Consequences

- Inter-departmental politics over business needs
- Conflicting project and disciplinary hierarchies and stakeholders
- Blameshifting

Antipattern: Solution Centrism



Antipattern: Solution Centrism

Description

Implementation solution as unifying factor

Reasons

- Vendor influence
- Experience drives selection of technology
- Sunk cost fallacy

Consequences

- Inefficiency due to hammer/nail problem
- Bottleneck by definition
- Technology, not domain as unifying factor
- Developer frustration
- Skills shortage in market
- Hard to motivate people to train in proprietary tech

Antipattern: Uncreative Chaos

Antipattern: Uncreative Chaos

Description

Lack of architectural structure & repeatable process for architectural decisions

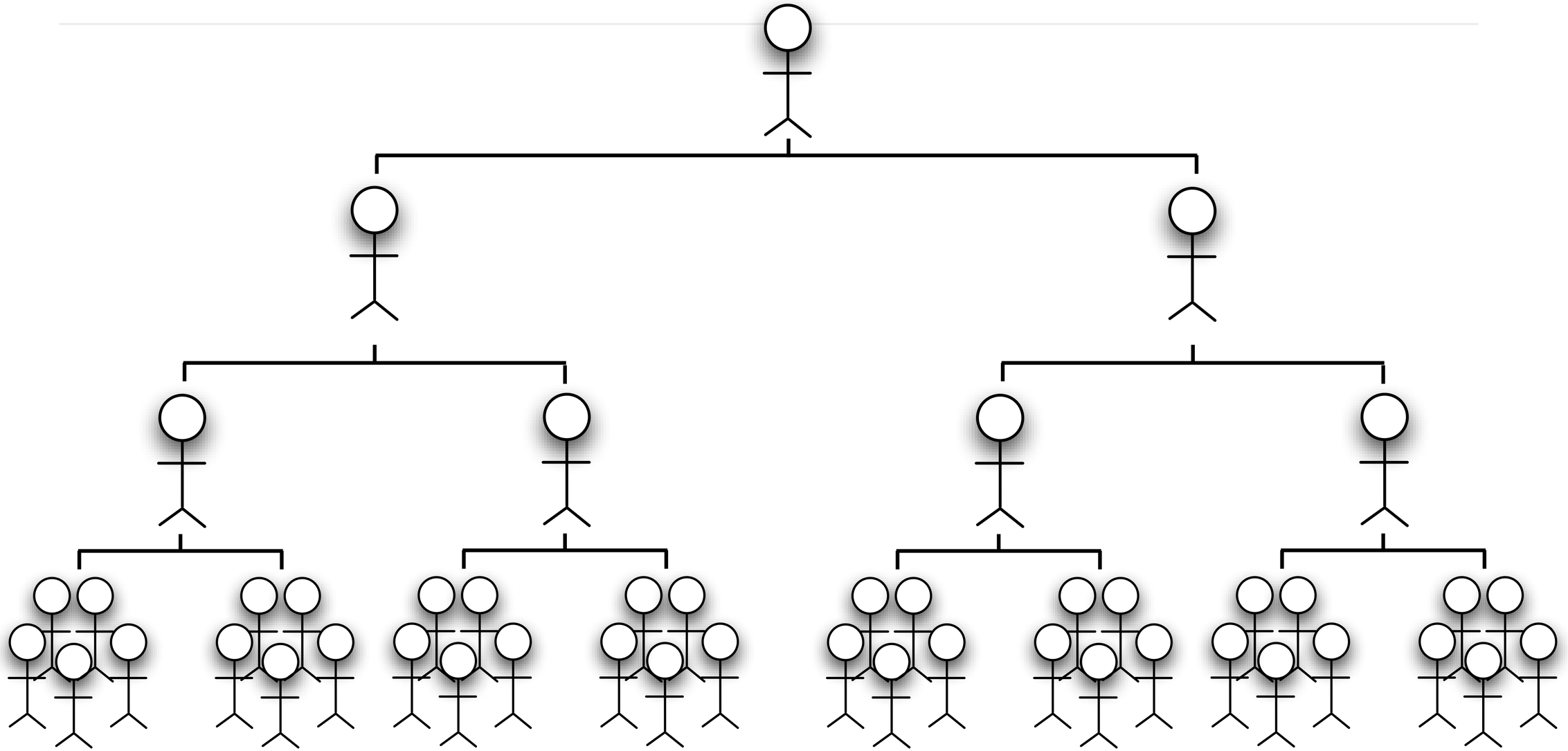
Reasons

- No (effective) centralized governance
- Non-technical senior management
- Focus on unnecessary standardization
- Strong business leaders, weak tech leaders

Consequences

- Redundancy in all aspects
- Frequent technology discussions between teams
- High integration costs and technical debt
- Slow delivery capability due to complexity
- Complex and expensive modernization

Antipattern: Authoritarian Regime



Antipattern: Authoritarian Regime

Description

Centralized decision making, strong standardization, homogeneous environment

Reasons

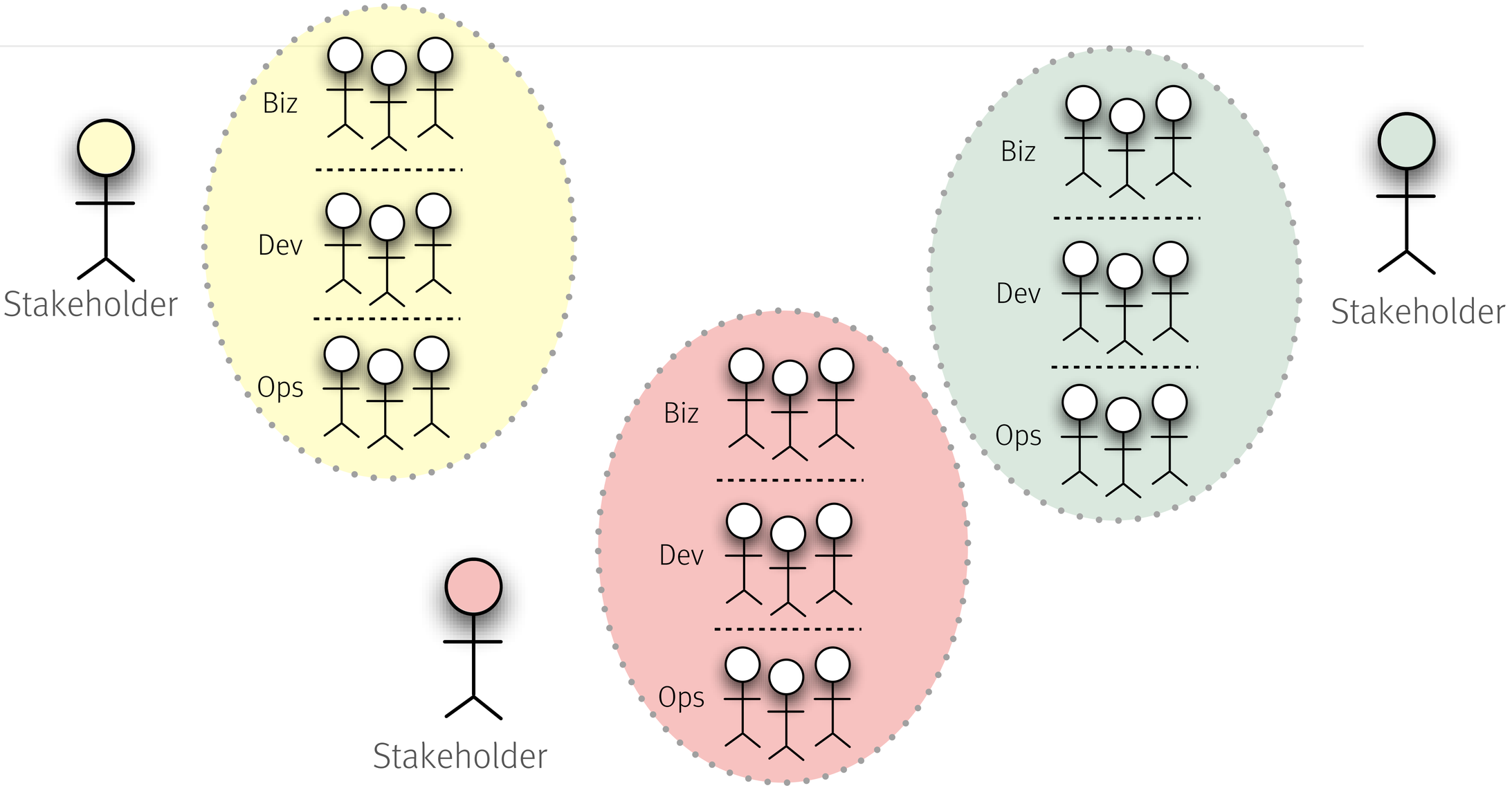
- Unpopular decisions (cost savings, product standardization, …)
- (Perceived or real) lack of skills in “lower levels”
- Possibly due to company culture

Consequences

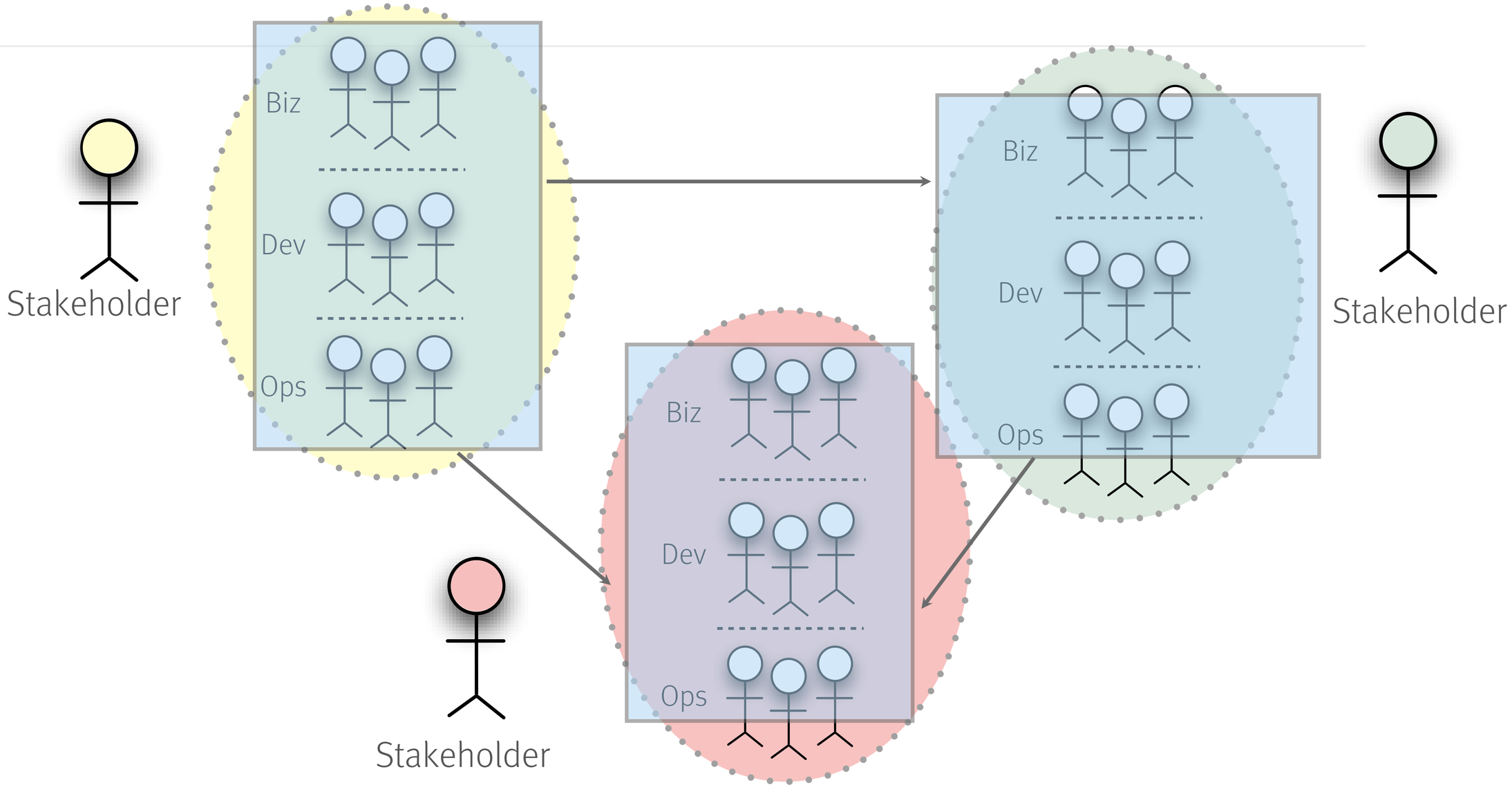
- Frustration and developer exodus
- Lack of innovation & speed because of bottlenecks
- Technology paralysis

Patterns

Pattern: Autonomous Cells



Pattern: Autonomous Cells



Pattern: Autonomous Cells

Description

Decentralized, domain-focused cells with maximum authority over all aspects of a set of capabilities

Approach

- Decisions are made locally on all aspects of a solution
- Success is measured via customer-oriented KPIs
- Cross-functional team with biz, dev, ops skills

Consequences

- Customer/end user focus
- Decentralized delivery capability
- Speed as #1 priority
- “Full-stack” requirement for developers and other roles
- Redundancy instead of centralization

Pattern: Evolutionary Architecture

Pattern: Evolutionary Architecture

Description

Architecture is constructed so it can evolve as much as possible over the course of (ideally indefinite) time

Approach

- Separation of large domain into “islands of change”
- Design for replacement, not for re-use
- Minimization of shared dependencies

Consequences

- Cell metaphor: Renewal over time
- Experimentation with different micro architecture approaches possible

Pattern: Regulated Market

Pattern: Regulated Market

Description

Let “the free market of ideas” decide what works best, but provide a framework of rules for interoperability

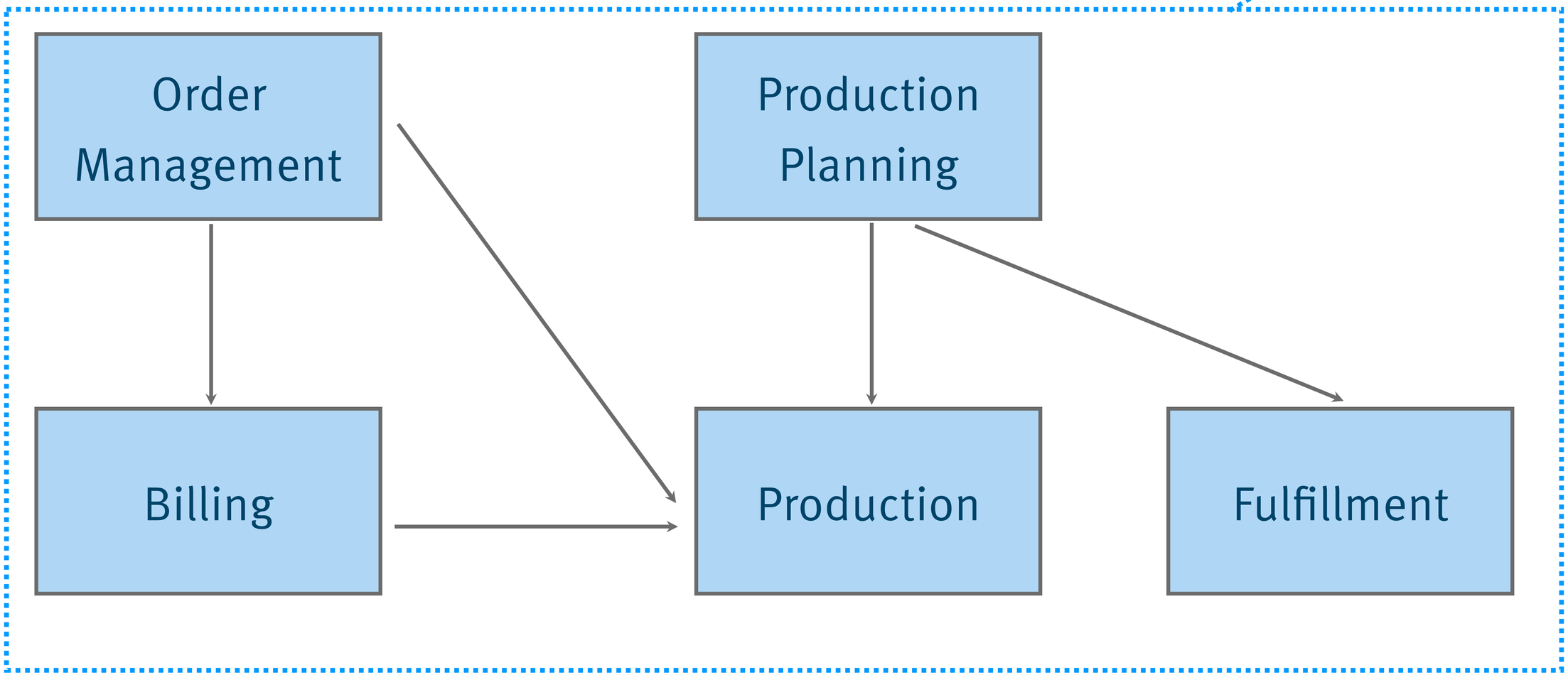
Approach

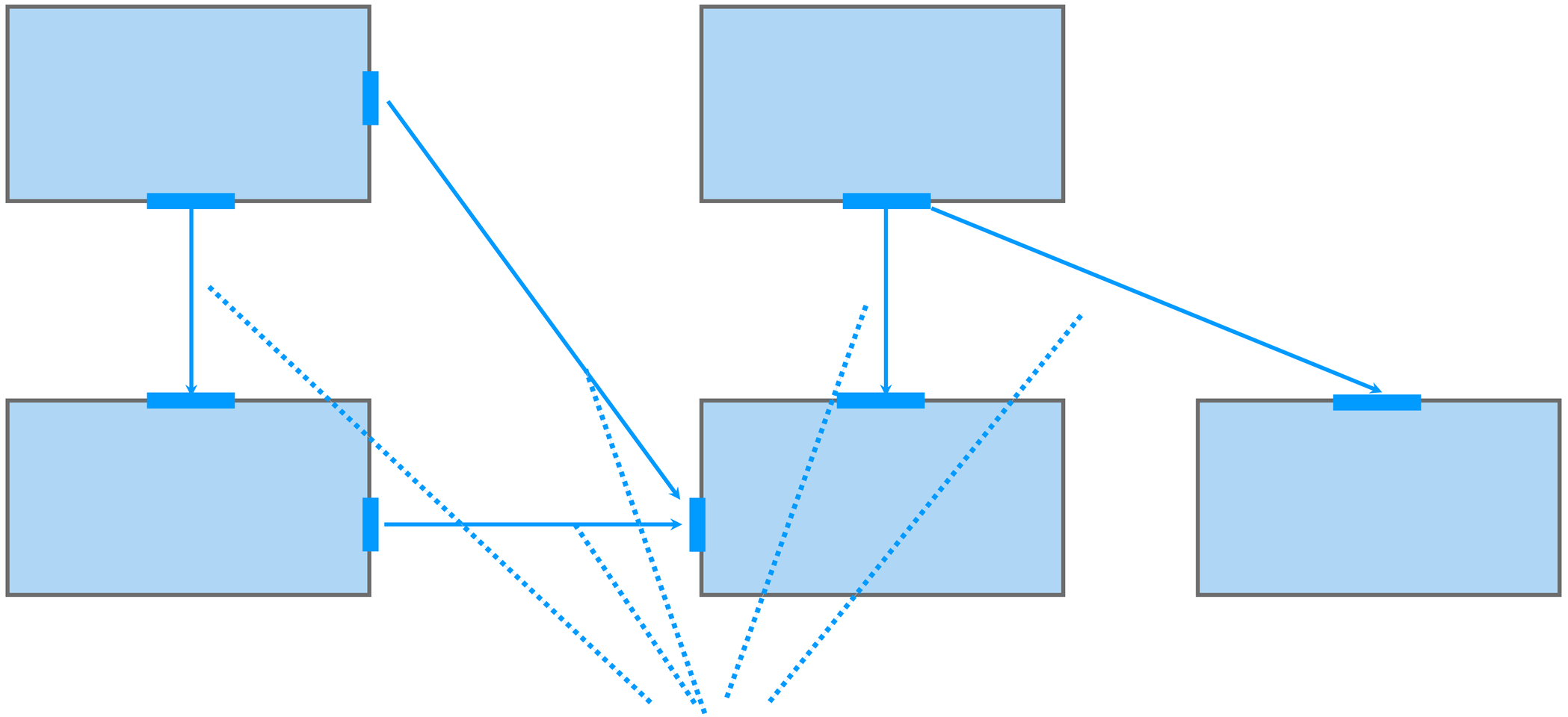
- Separate micro & macro architecture
- Strictly enforced rules for macro architecture
- Loose, minimal governance for micro architecture

Consequences

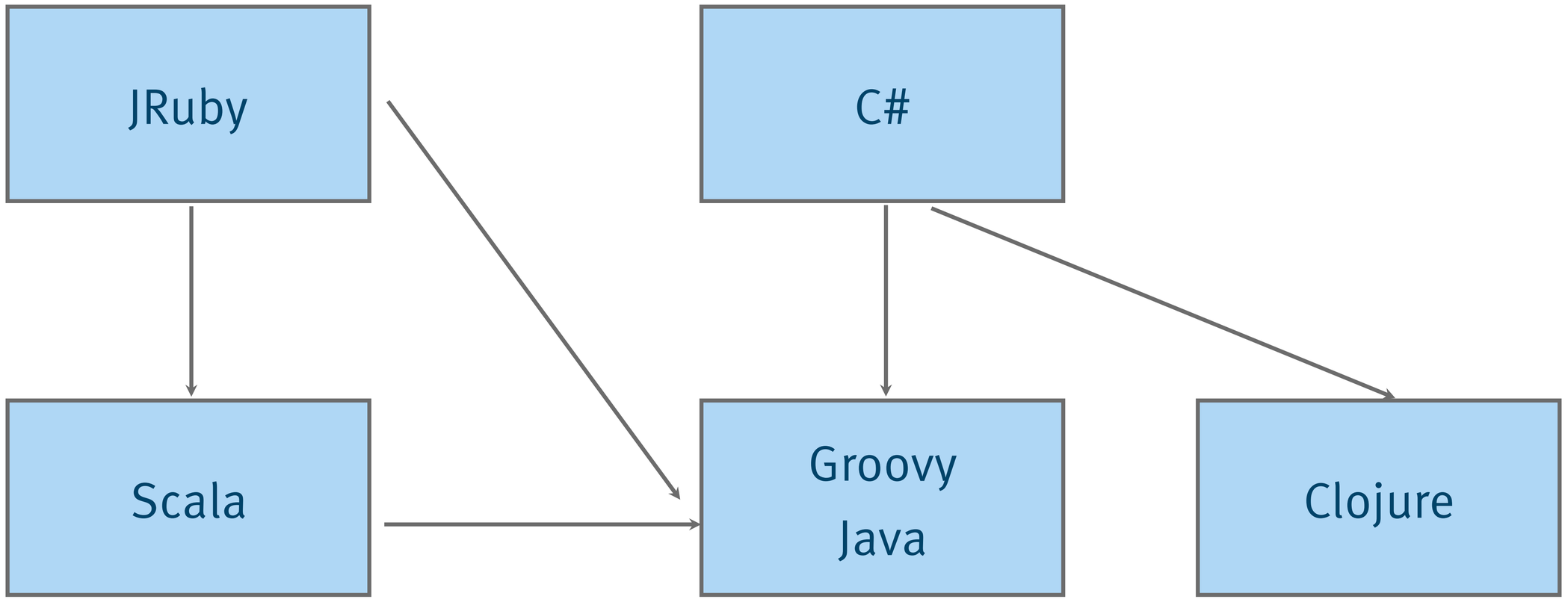
- Motivated developers
- Experimentation with different micro architecture approaches possible
- Best-of-breed approach
- Local optima

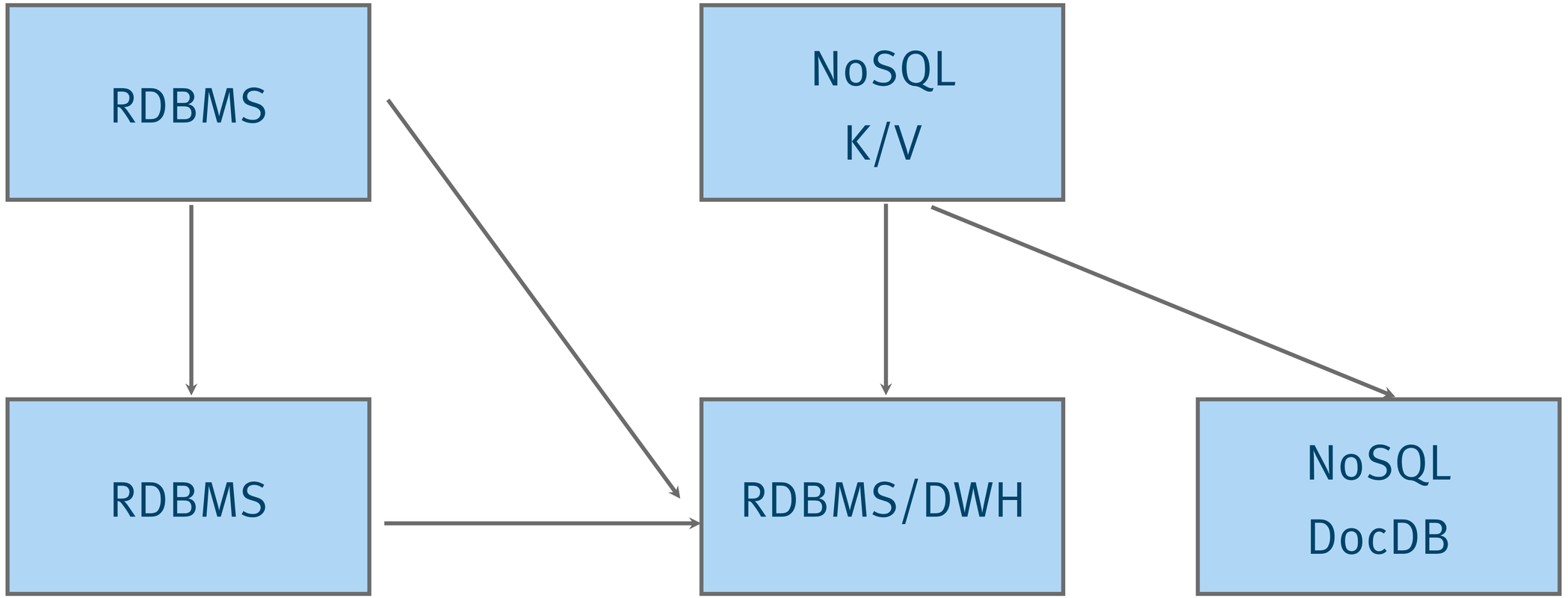
Domain architecture

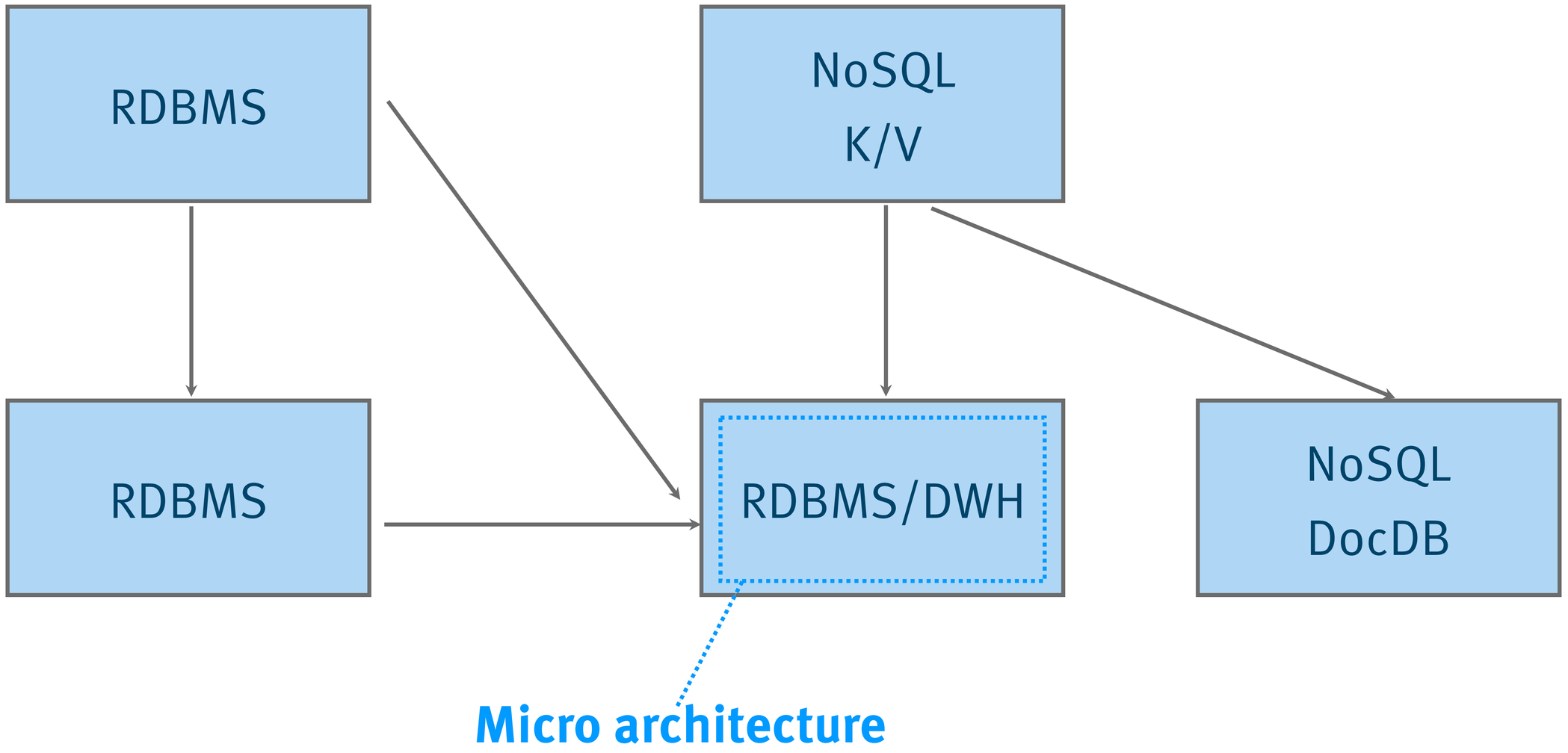




Macro (technical) architecture


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: Marketing-based Governance

Pattern: Marketing-based Governance

Description

Architectural approaches are evangelized instead of mandated

Approach

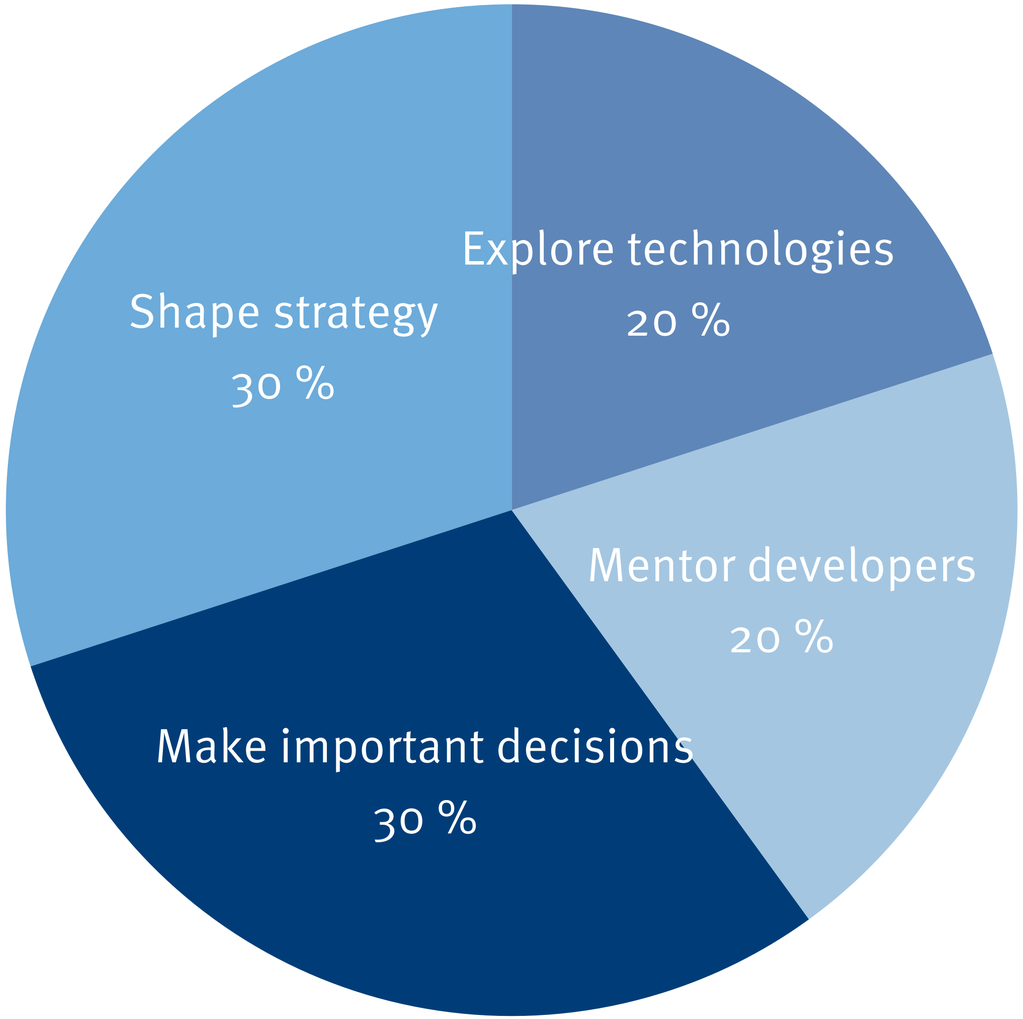
- Disseminate information via blogs, brown-bag sessions, public talks
- Architects as communicators
- Integration with public/ community work

Consequences

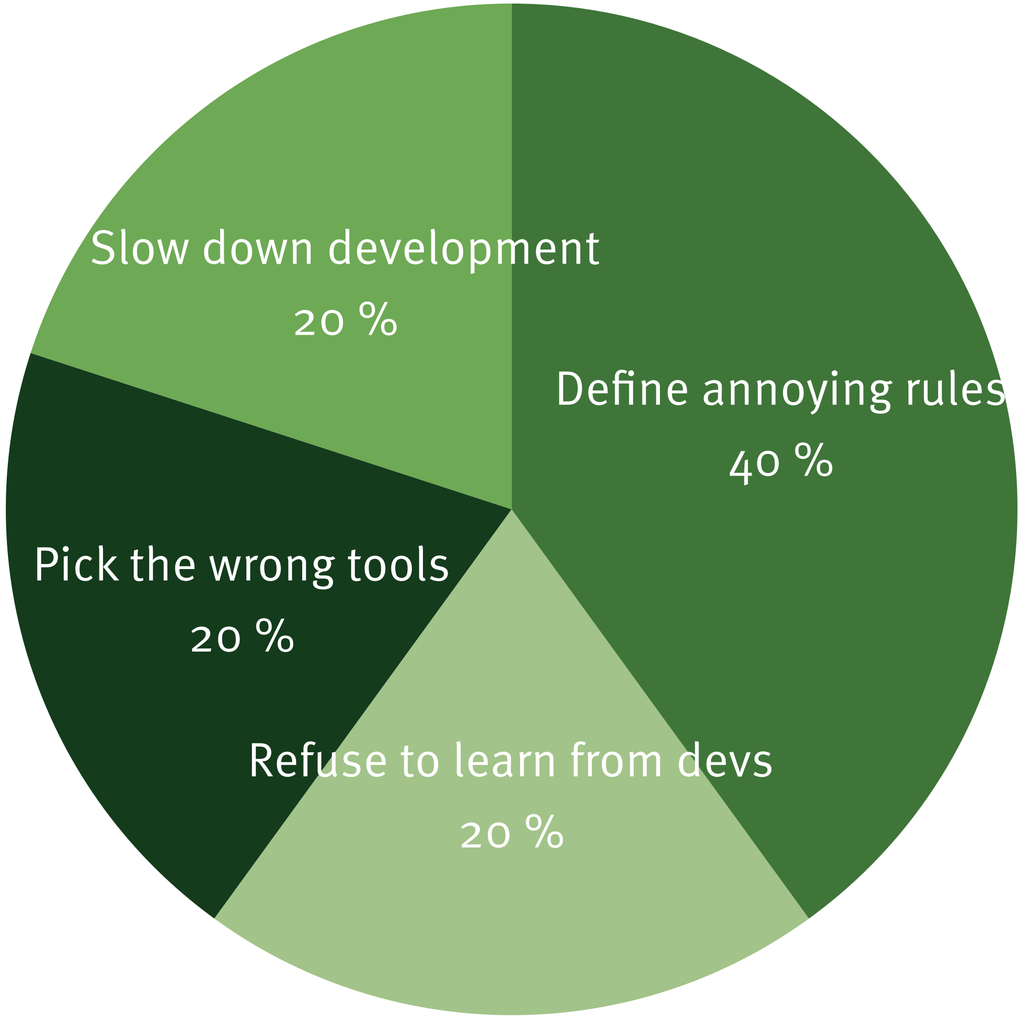
- More heterogeneity
- Similarity to industry
- Decisions made based on a solution’s merit
- Bottom-up modernization

What Architects Do

What architects want to do



What others think architects do



What architects *actually* do

Recommendations

1.

Acquire domain knowledge

2.

Partner with business stakeholders

3.
Create evolvable structures

4.
Get out of the way as quickly as possible

That’s all I have.

thanks for listening!

@stilkov

Stefan Tilkov

stefan.tilkov@innoq.com

Phone: +49 170 471 2625

innoQ

www.innoq.com

innoQ Deutschland GmbH

Krischerstr. 100
40789 Monheim am Rhein
Germany
Phone: +49 2173 3366-0

Ohlauer Straße 43
10999 Berlin
Germany
Phone: +49 2173 3366-0

Ludwigstr. 180E
63067 Offenbach
Germany
Phone: +49 2173 3366-0

Kreuzstraße 16
80331 München
Germany
Phone: +49 2173 3366-0

innoQ Schweiz GmbH

Gewerbestr. 11
CH-6330 Cham
Switzerland
Phone: +41 41 743 0116

Image Credit

https://pixabay.com/en/chaos-room-untidy-dirty-messy-627218/

https://commons.wikimedia.org/wiki/File:Wroclaw_Daily_Market.jpg

https://pixabay.com/en/smartphone-face-man-old-baby-1790833/

https://pixabay.com/en/marketing-customer-center-2483856/

About Stefan Tilkov

- CEO/Co-founder & principal consultant at innoQ
- Author and frequent conference speaker
- stefan.tilkov@innoq.com
- @stilkov

About innoQ

- Offices in Monheim (near Cologne), Berlin, Offenbach, Munich, Zurich
- ~125 employees
- Core competencies: software architecture consulting and software development
- Privately owned, vendor-independent
- Clients in finance, telecommunications, logistics, e-commerce; Fortune 500, SMBs, startups

www.innoq.com