Boolan
高端IT互联网教育平台

2017
SA-Summit

全球软件架构技术大会

演讲稿合辑

www.sa-summit.org

“全球大师”系列高端在线课程

《CTO必修课》

风靡美国技术界的CTO必修课，揭开技术管理迷雾

主讲老师：

Martin Abbott

全球软件架构大师
《架构即未来》作者

Martin 从 eBay 创业初期直到发展为全球500强企业，担任 eBay 高级技术副总裁以及首席技术执行官。在加入eBay之前，Martin 曾在 Gateway 以及 Motorola 担任技术、管理等重要职位。

面向技术人员和技术管理人员: CTO、技术VP、首席架构师、技术总监、开发主管、研发经理。

- 提升自身领导力及架构能力
- 解决技术团队及架构现存问题
- 提升团队凝聚力及创新力

10+小时课程时长 丰富准则、工具、经验案例。

- 凝聚Martin Abbott 20年一线实操精粹，历经350家企业实践检验。
- 历经数十年精心提炼打磨，专业在线课程团队全心策划制作。

2018 震撼推出！课程体验名额限量招募中。
扫码填写登记表。

akf PARTNERS

SCALING TECHNOLOGY AND ORGANIZATIONS

MARTY ABBOTT
CEO
AKF PARTNERS

AKF Positioning Statement

For companies dissatisfied with or challenged to meet the demands of their growth, we are growth problem solvers. Unlike technology consulting firms, our approach identifies the root causes from all sources including architecture, process, organization, product and strategy.

Most consulting firms fix the symptoms of your problem. We treat both the symptoms and the causes. We are the world's first growth consulting firm.

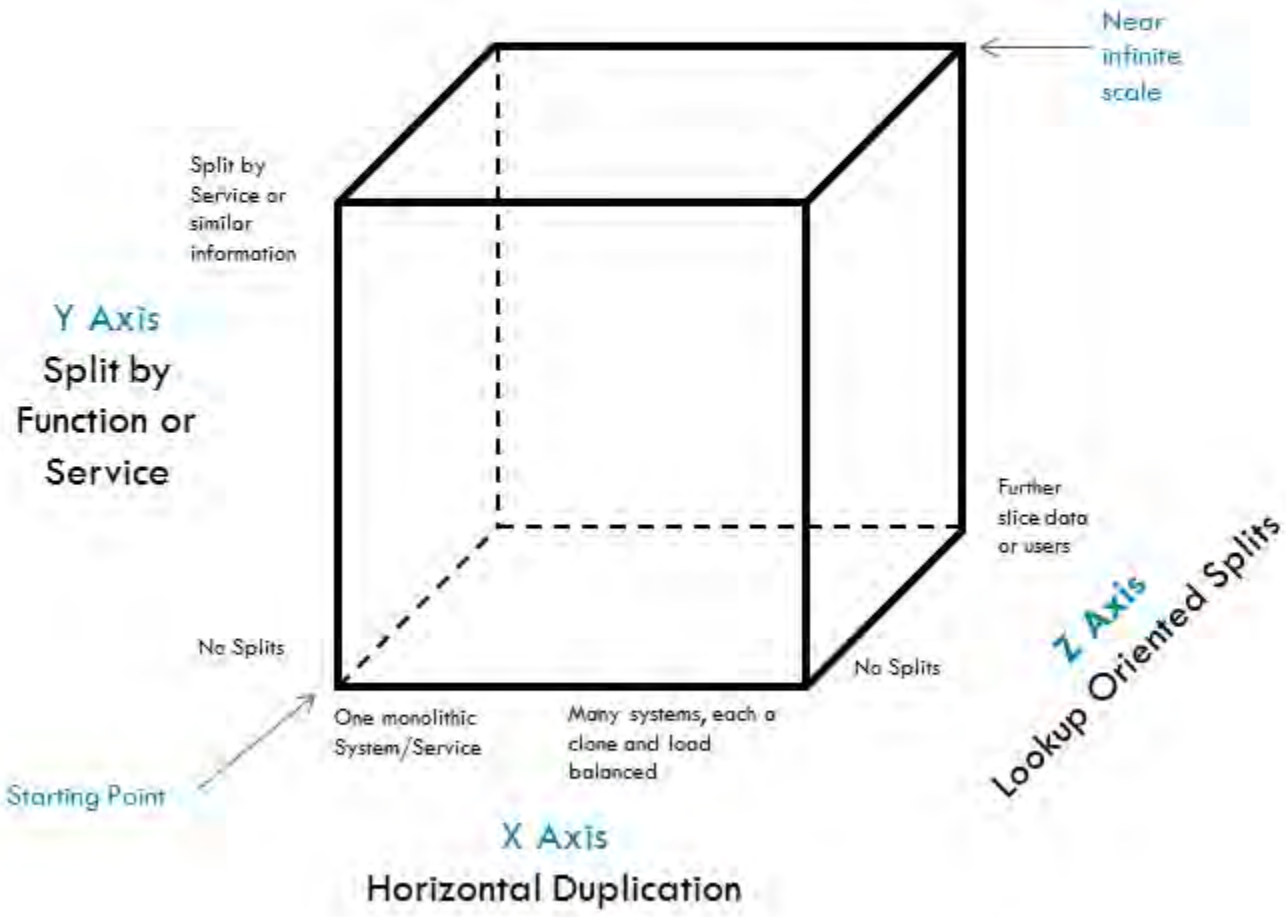
We differentiate ourselves through superior leadership and fast time to client returns. We bridge the gap between technology, product and strategy.

Where we exist

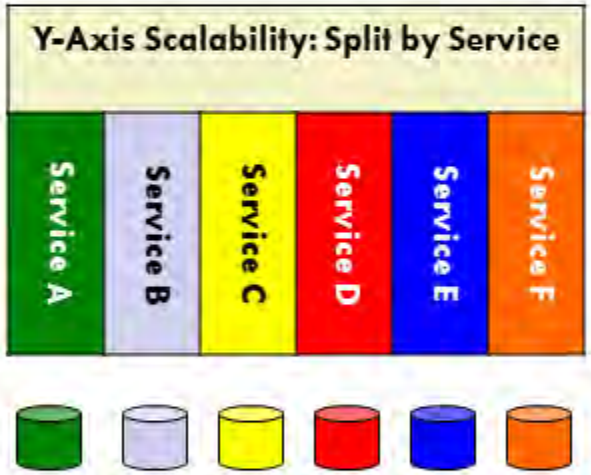
akf PARTNERS

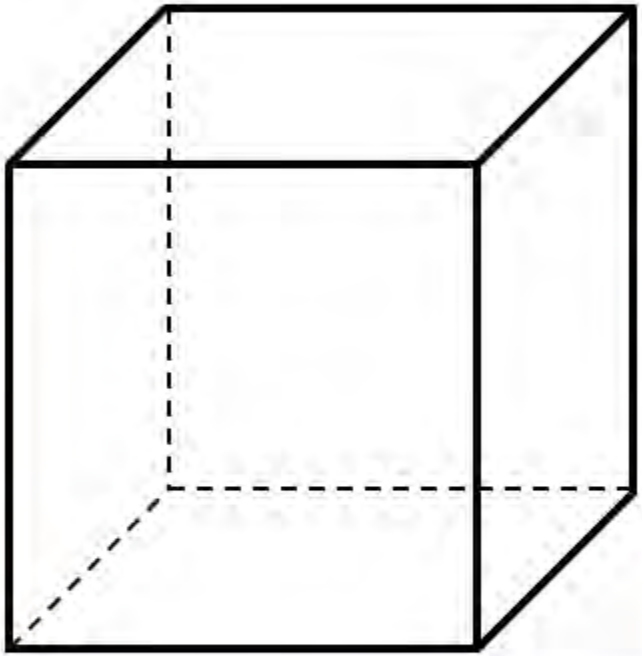
AKF SCALE CUBE

AKF Scale Cube



AKF Scale Cube

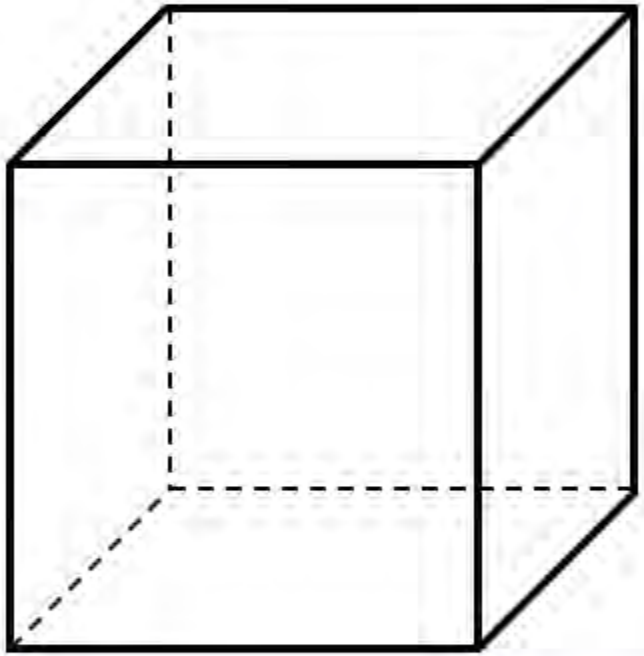




X-Axis Scalability: Replicate & LB	
Web Tier	Replicate Web Servers & Load Balance
App Tier	Store Session in browser or separated Object Cache to horizontally scale app tier independent of web tier
DB Tier	Use Read-Replicas for read-only use cases like reporting, search, etc.

AKF Scale Cube

Pros	
Easy	Simple to implement
Fast	Typically very fast to implement
Transaction Scalability	Scales transactions well

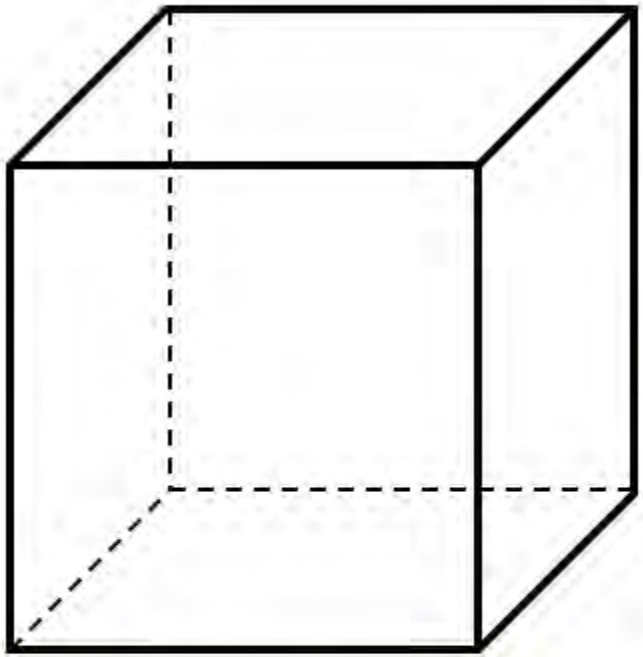


Cons	
Storage	Can be costly
Cost Scalability	Multiple data sets
Cacheability	Doesn't address caching

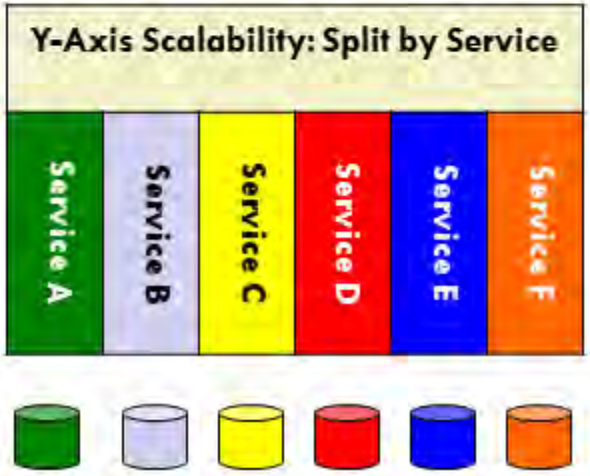
X-Axis Scalability: Replicate & LB	
Web Tier	Replicate Web Servers & Load Balance
App Tier	Store Session in browser or separated Object Cache to horizontally scale app tier independent of web tier
DB Tier	Use Read-Replicas for read-only use cases like reporting, search, etc.

AKF Scale Cube

Pros	
Transaction Scalability	Scales transactions well
Cost	Independent scale of services
Org Scale	Allows for Org Scale
Cache	Increases cache hit rate
Availability	Can have fault isolation

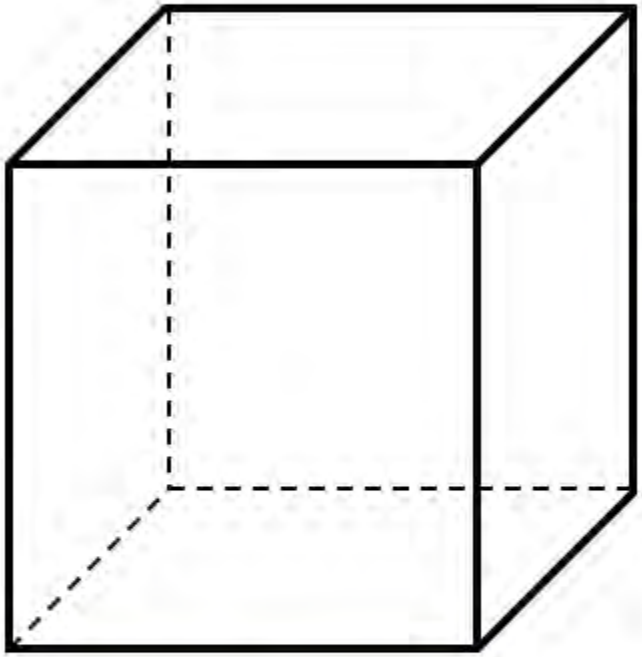


Cons	
Difficult	Can take time to architect
Complexity	More services

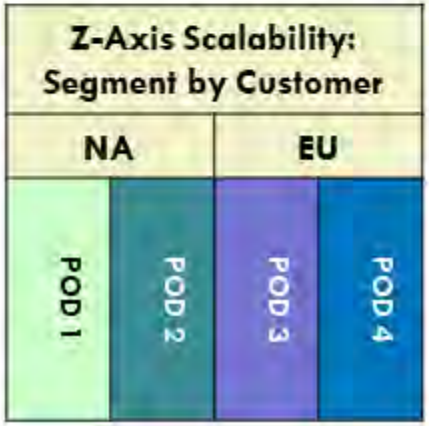


AKF Scale Cube

Pros	
Transaction Scalability	Scales transactions well
Latency	Reduces response times
Cache	Increases cache hit rate
Availability	Can have fault isolation



Cons	
Difficult	Can take time to architect
Complexity	More things to manage

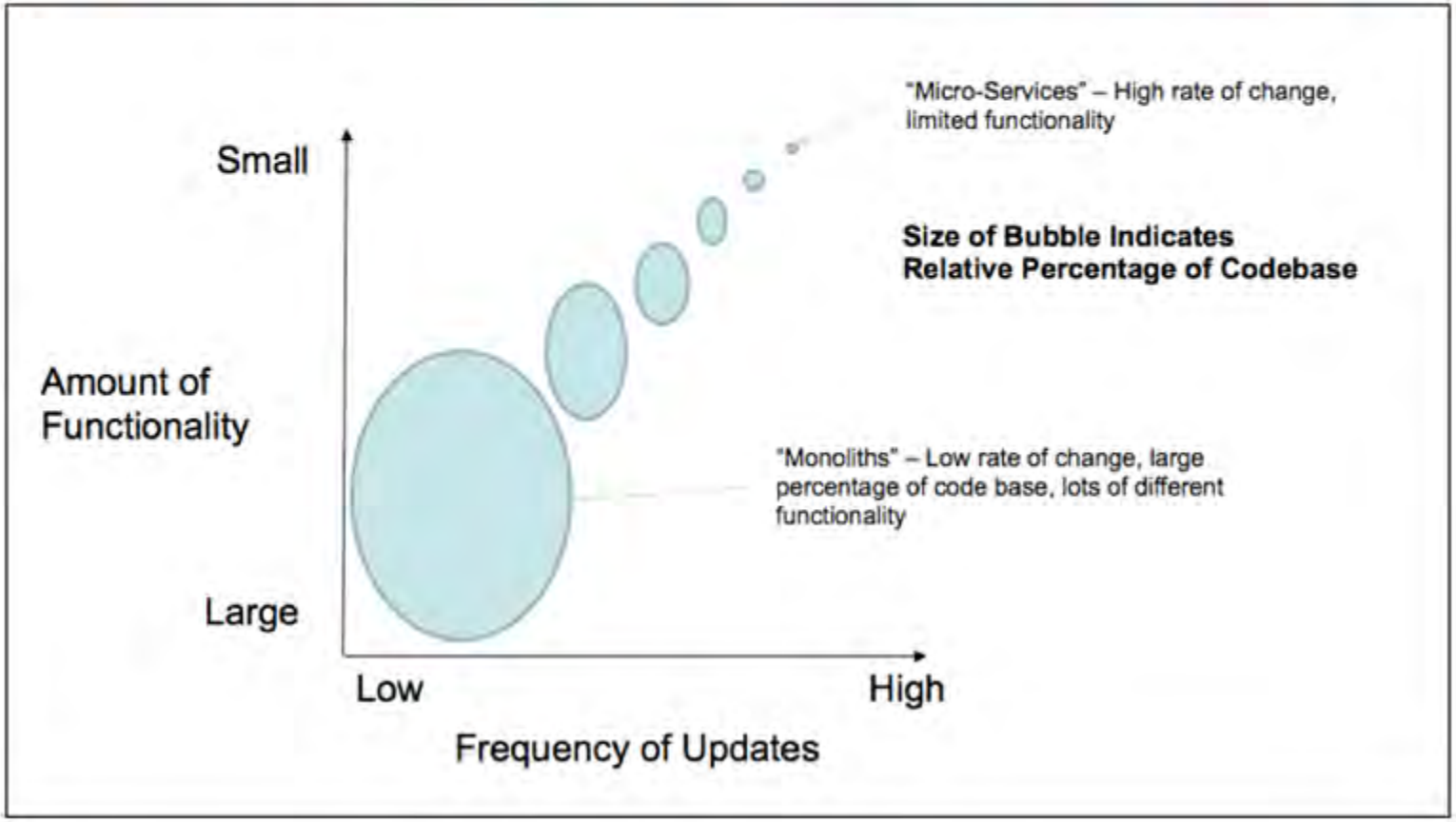


When splitting services, for the Y axis, how far should one go?

Determining Service Size

akf PARTNERS

The diagram represents the relationship between functionality, frequency of changes, and the relative percentage of the codebase across your system.



Determining Service Size

akf PARTNERS

Common Questions and Considerations When Splitting

How Granular Should The Split Services Be? How Many Swimlanes Should We Split Out? Which Services Should Be Grouped Together?

There is a point where splitting too many of your services will yield little return or even cost you.

Developer Throughput

- ✓ Frequency of Change?
- ✓ Degree of Reuse?
- ✓ Team Size
- ✓ Specialized Skills?

Availability and Fault Tolerance

- ✓ Desired Reliability and Dependency on Other Functions?
- ✓ Criticality to the Business?
- ✓ Risk of Failure?

Cost

- ✓ Effort To Split Code?
- ✓ Shared Persistent Storage Tier?
- ✓ Network Configuration?

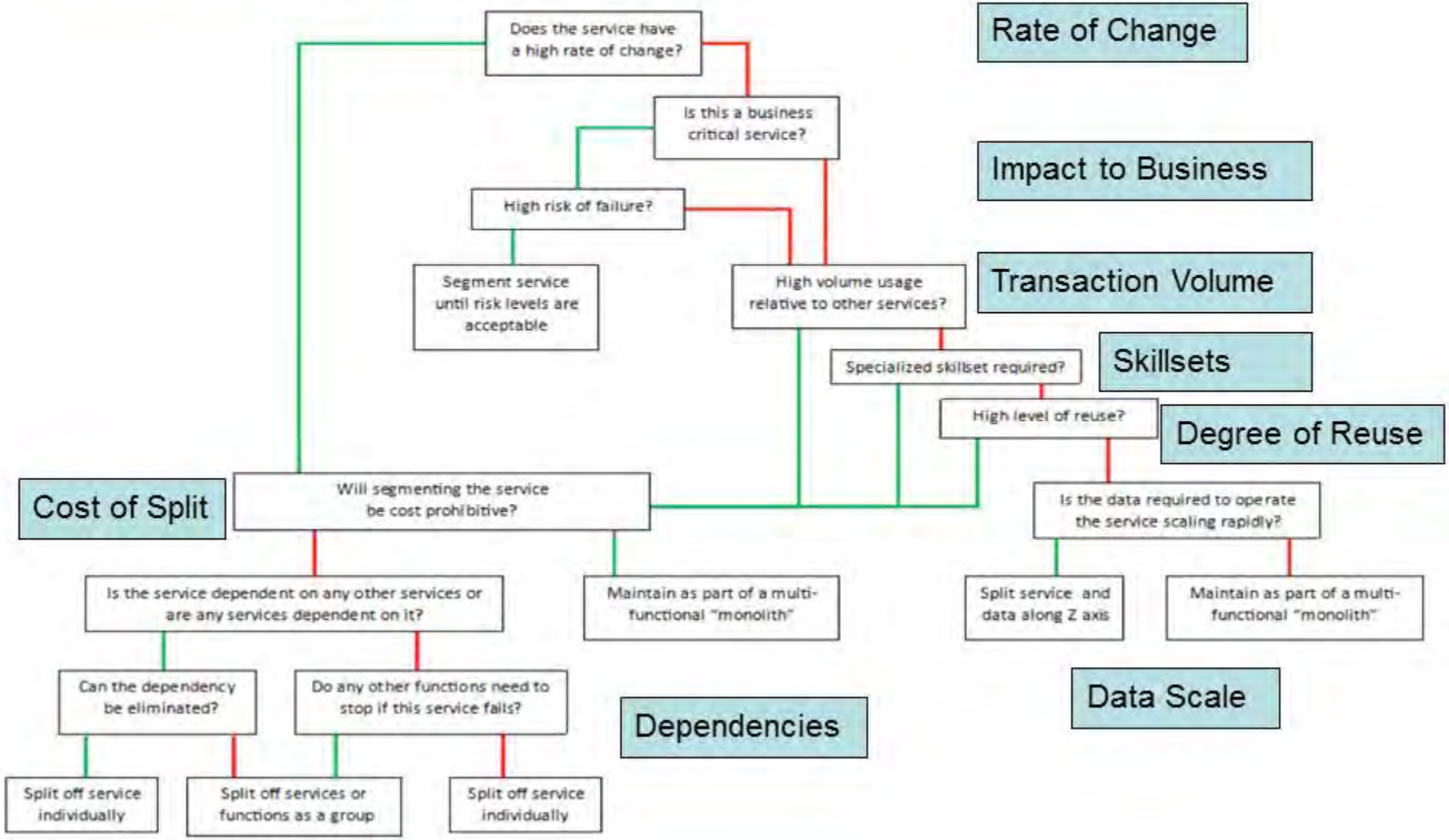
Scalability

- ✓ Scalability of Data?
- ✓ Scalability of Services?
- ✓ Dependency on Other Service's Data?

Determining Service Size

akf PARTNERS

Decision Tree for Applying Splits to your Design



akf PARTNERS

ARCHITECTURE PRINCIPLES FOR SCALE AND AVAILABILITY – TOP 10

Architecture Principles

The Top 10

Design for Rollback. Ensure you can roll back any release of functionality.

Design to Be Monitored. Think about monitoring during design, not after.

Isolate Faults. Practice fault isolative design – implement circuit breakers to keep failures from propagating.

Scale Out Not Up. Never rely on bigger, faster systems.

Asynchronous Design. Communicate synchronously only when absolutely necessary.

Stateless Systems. Use state only when the business return justifies it.

Build Small, Release Small, Fail Fast. Build everything small and in iterations to grow.

Automation Over People. Build everything to be automated – never rely on people to do something that a robot can do.

Buy When Non-Core. If you aren't the best at building it and it doesn't offer competitive differentiation, buy it.

Design for at Least Two Axes. Think one step ahead of your scale needs.

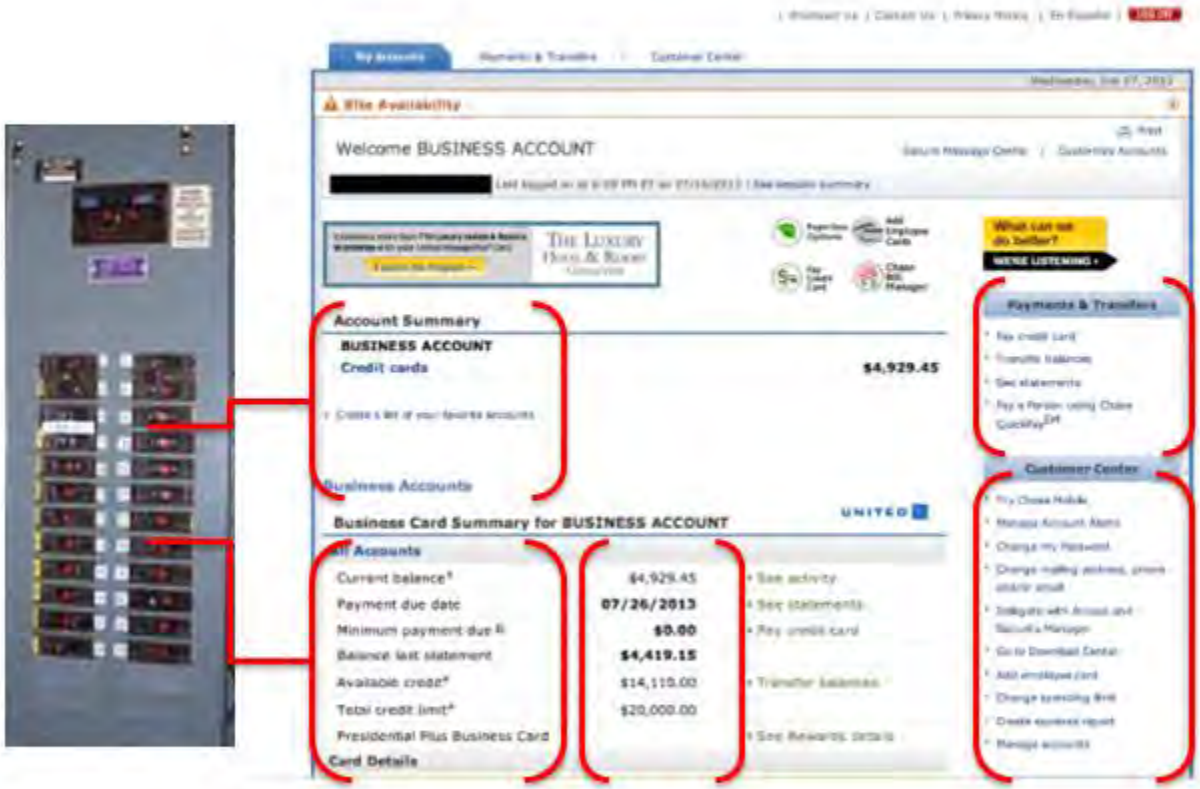
Fault Isolation and Y Axis

Everything Fails.

We Must Implement Circuit Breakers to Contain Failures.

Y Axis Scalability:

Functionally Split Services & Fault Isolate. Portions of a page delivered from separate services.



Each bracket represents a component of the page that fails or succeeds independently. A failure of that element does not bring down the entire page.

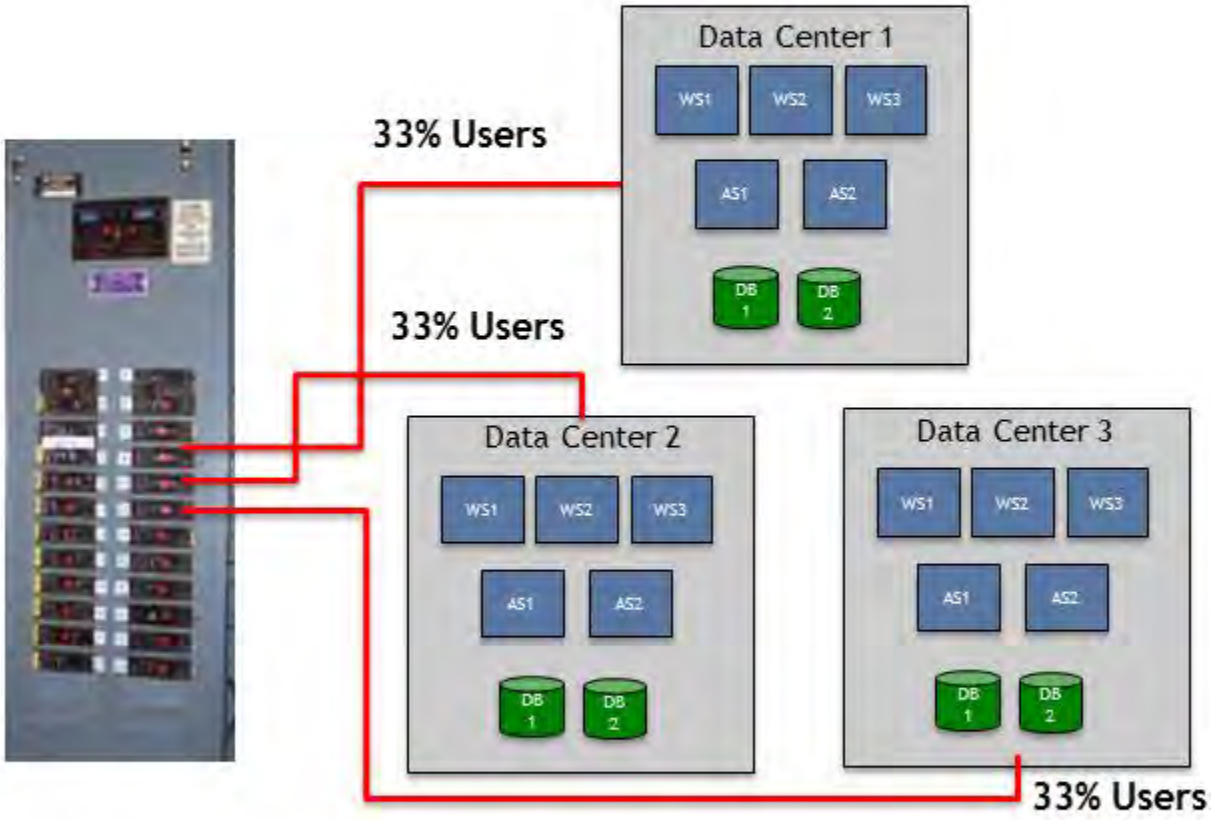
Fault Isolation and Z Axis

Everything Fails.

We Must Implement Circuit Breakers to Contain Failures.

Z Axis Scalability:

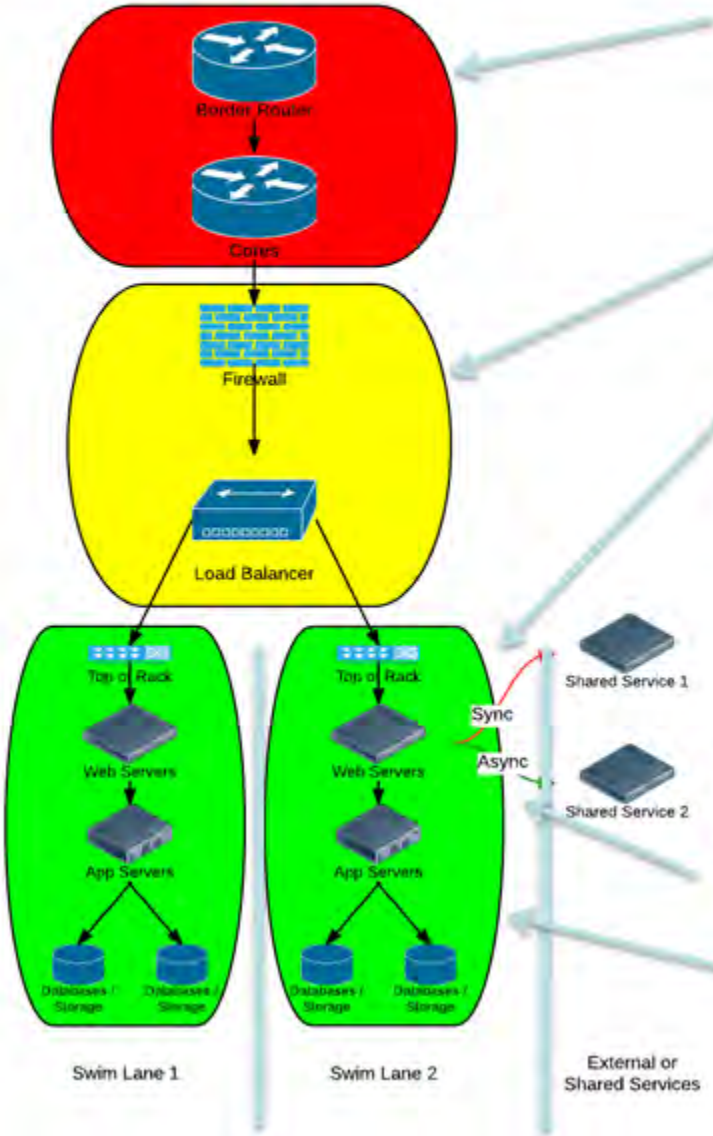
Split Customers and assign them to data centers for fault isolation.



Separate customers into partitions. Make changes independently.

What constitutes a fault isolated swim lane?

akf PARTNERS



Not Isolated – generally cost prohibitive to justify full isolation.

Isolate if Practical – firewalls and load balancers are common failure points, but are expensive. Ideally each swim lane will have its own firewalls and load balancers.

Synchronous calls to outside services cause problems when they are slow or unavailable. If you must make a synchronous call, consider building a copy of the service within the swim lane.

Asynchronous calls to outside apps and services allow for fault isolation – an app or service is designed for fault isolation when they can "fire and forget" or deal with the outside service being slow or unavailable.

Always isolate – Top of Rack Switches, Compute, and Storage should always be fault isolated within a swim lane.

akf PARTNERS

SCALING ORGANIZATIONS

DESIGNING FOR EMPOWERMENT, ACCOUNTABILITY AND INNOVATION

40+ Years of Research

Management and Technology researchers, theorists, professors and practitioners have known for more than 40 years that developing software based solutions in functional organizations is hazardous.

And yet, we have largely done nothing about it.

That is, until recently.

Drucker: Why is it that large companies can't seem to innovate (one exception)?

Drucker: Most innovation and most employment in the US from the SMB space (<500 employees).

Innovation and Entrepreneurship, 1986

Mel Conway (1968): **Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure.**

Conway's Law (Popularized by Fred Brooks)

Brooks' Corollary (1975): Because the design that occurs first is almost never the best possible, the prevailing system concept may need to change. Therefore, flexibility of organization is important to effective design.

Mythical Man Month

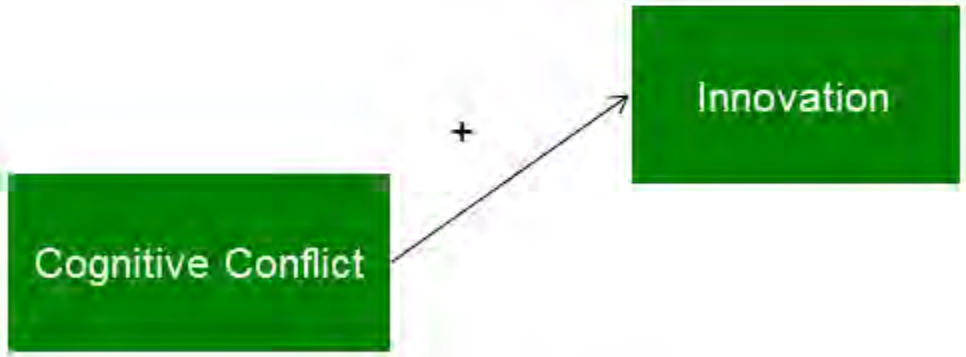
Steve Blank: **Innovation comes from opportunity, chaos and rapid experimentation.**

Why Innovation Dies

How does the structure of your organization impact its ability to innovate?

Cognitive Conflict

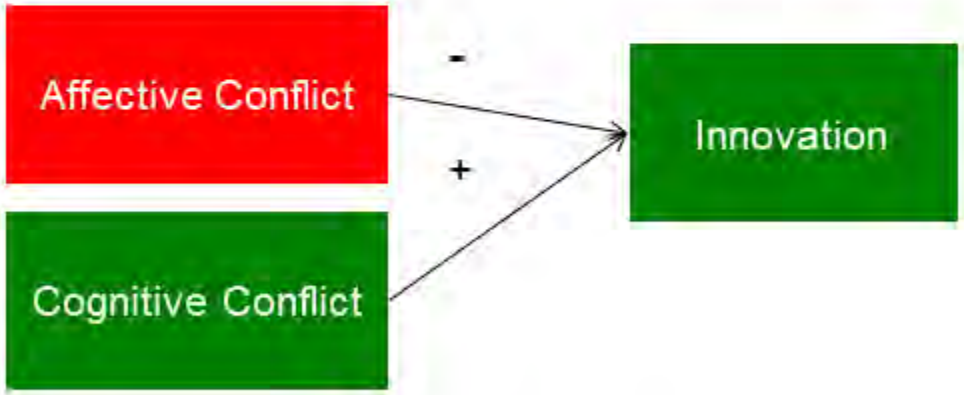
As cognitive conflict increases (what and why), so does innovation



**De Dreu and Weingart 2003

Affective Conflict

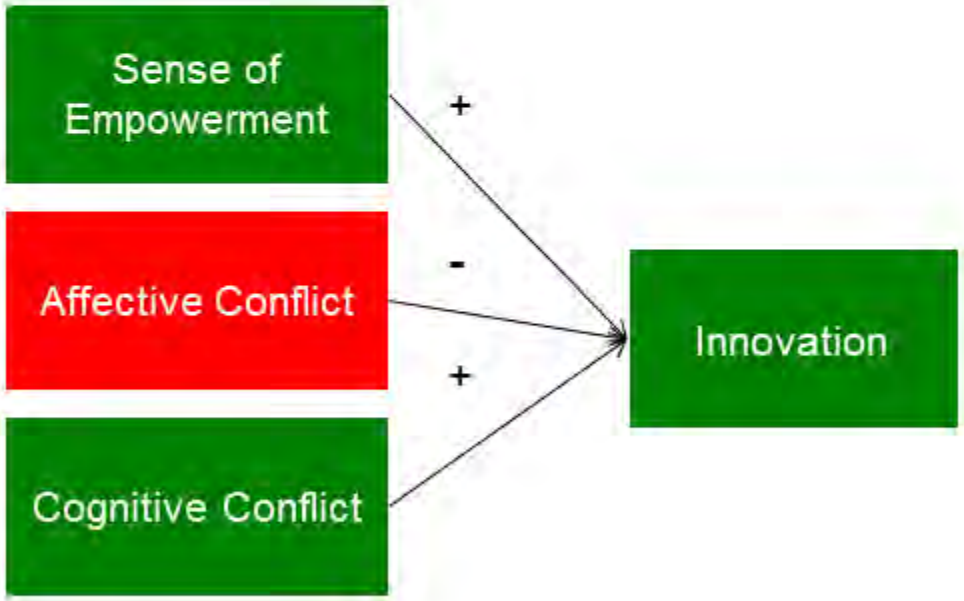
As affective conflict increases (who and how), innovation decreases



**Amason and Mooney, 1999; De Dreu and Weingart 2003

Empowerment

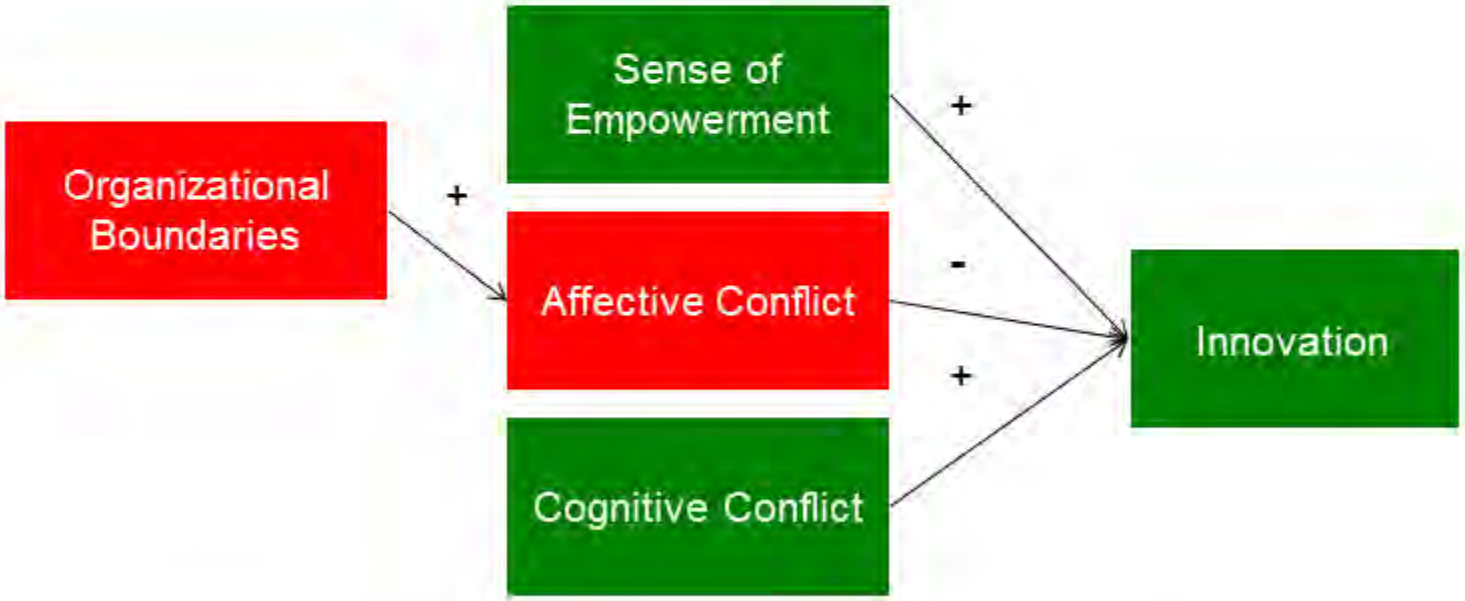
Feelings of empowerment increase innovation



**Spreitzer, De Janasz, 1999

Org Boundaries

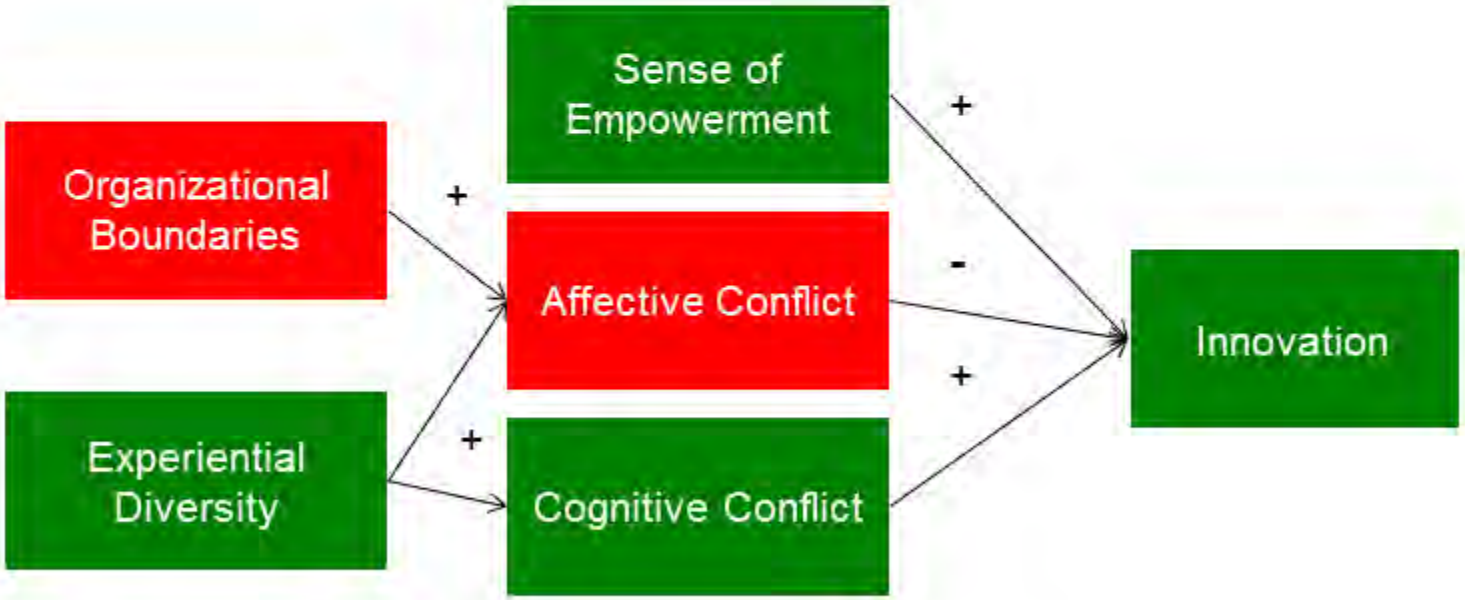
Organizational boundaries across which collaboration must happen increase affective conflict



** Tajfel 1974; Hogg and Terry 2000; Cummings and Kiesler 2005

Diverse Experience

Both cognitive and affective conflict driven by diversity in experiences

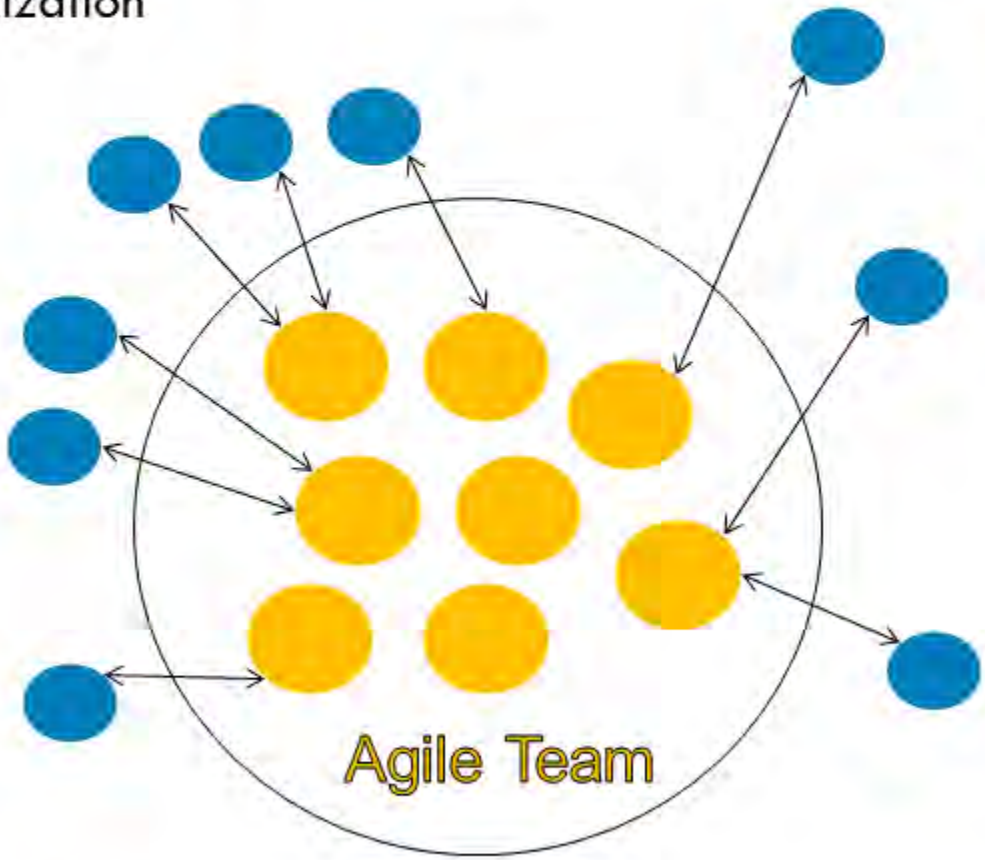


** Burt 2001; Bassett Jones 2005; Abbott, Lyytinen et al. 2010

Network Diversity

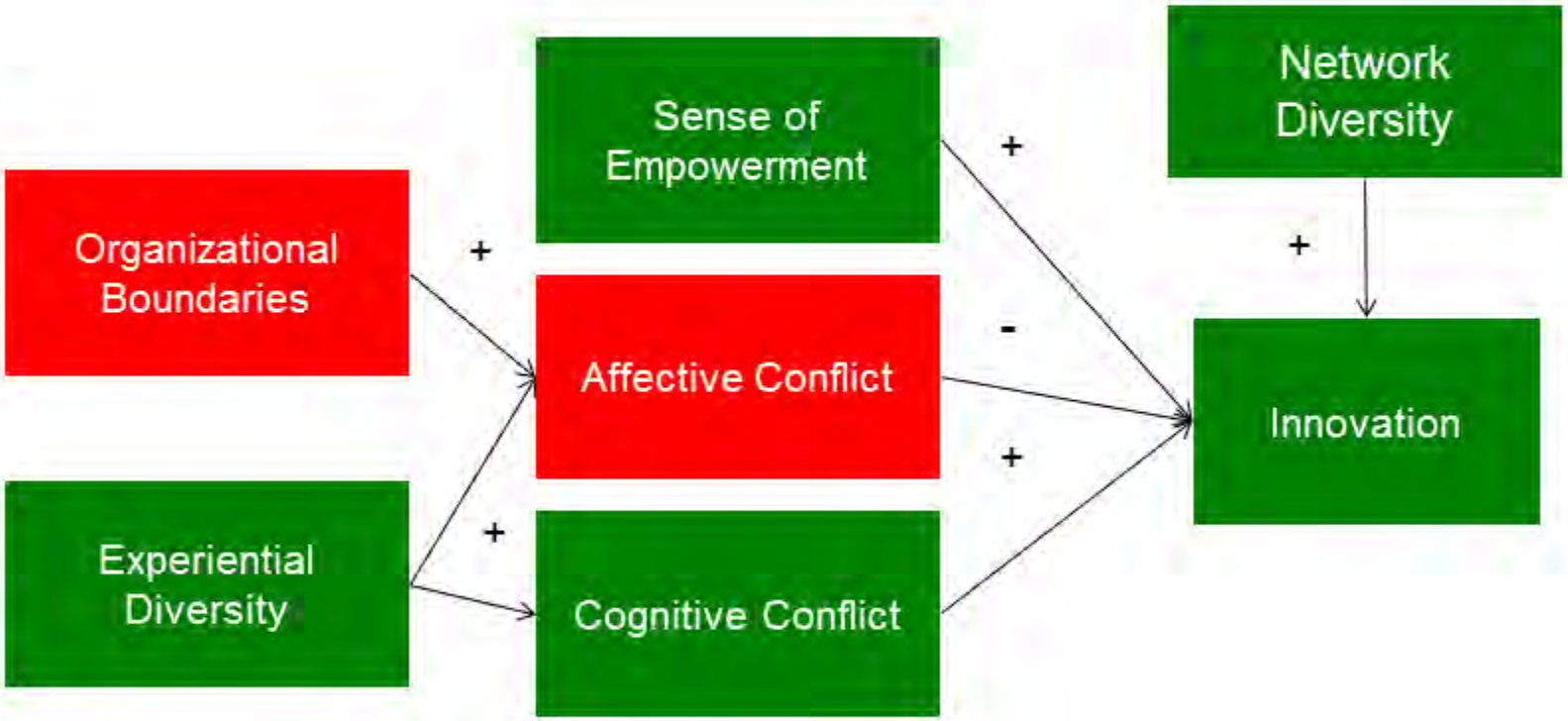
Best when high number of **non-overlapping** network links (loose acquaintances) outside of the organization

Network Diversity



** Burt 2001; Bassett Jones 2005; Abbott, Lyytinen et al. 2010

Driving Innovation

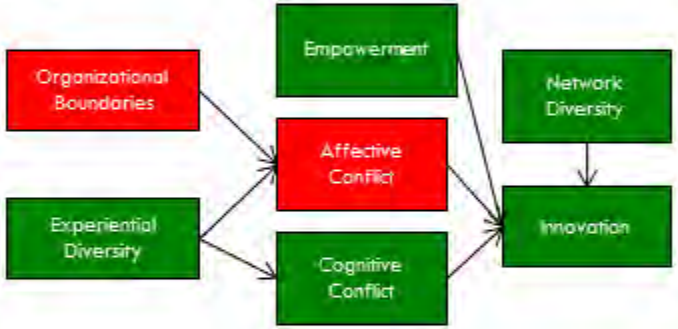


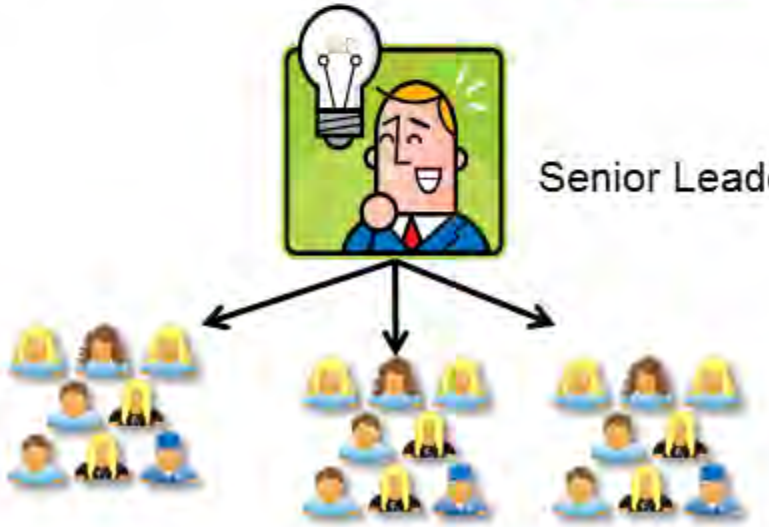
Barriers to Innovation

Top Down Innovation

This type of innovation should be avoided in favor of team driven innovation. Empowering teams with KPIs/Goals as well as with all the necessary/diverse skillsets to achieve the goals will help drive innovation.

- **No Empowerment/Ownership**
- **Does not leverage network**
- **Does not leverage experience**





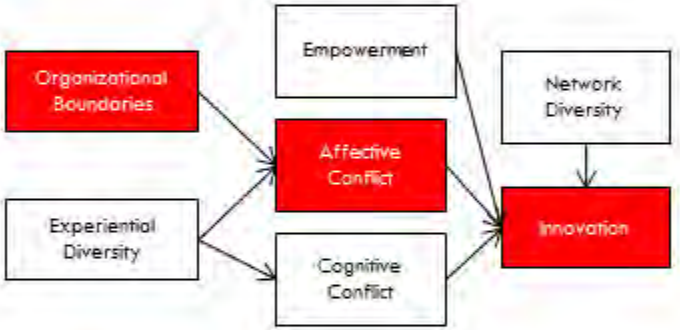
Barriers to Innovation

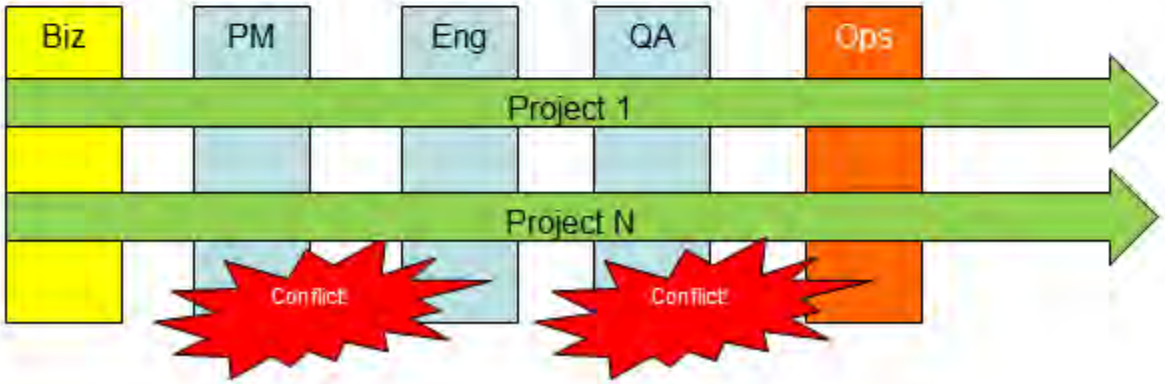
akf PARTNERS

Functional Organizations

This type of organization structure promotes skill-based efficiency over innovation and empowerment.

- Organizational boundaries create conflict
- "Who and How" dominate instead of "What and Why" which slows innovation.





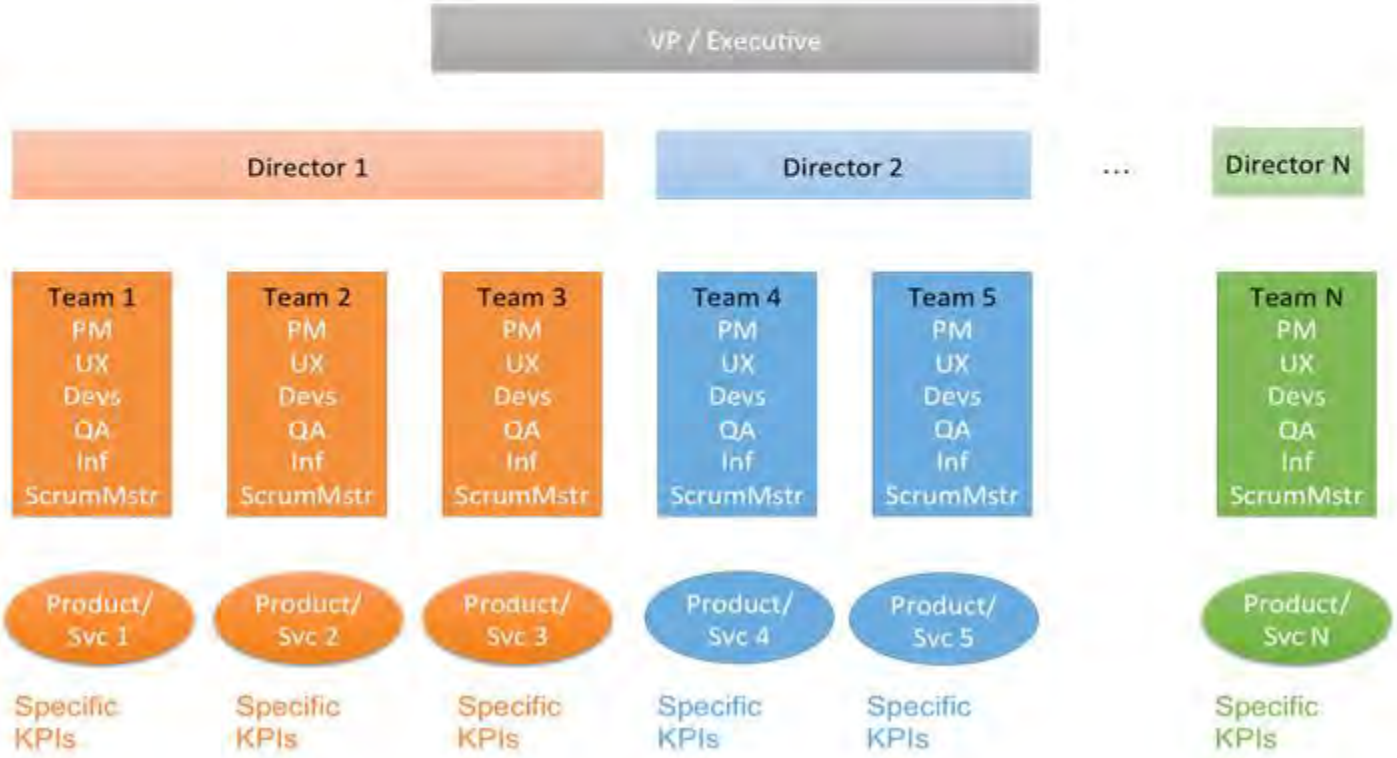
akf PARTNERS

MULTIDISCIPLINARY AGILE TEAMS

Organization Options

Multidisciplinary Agile Teams

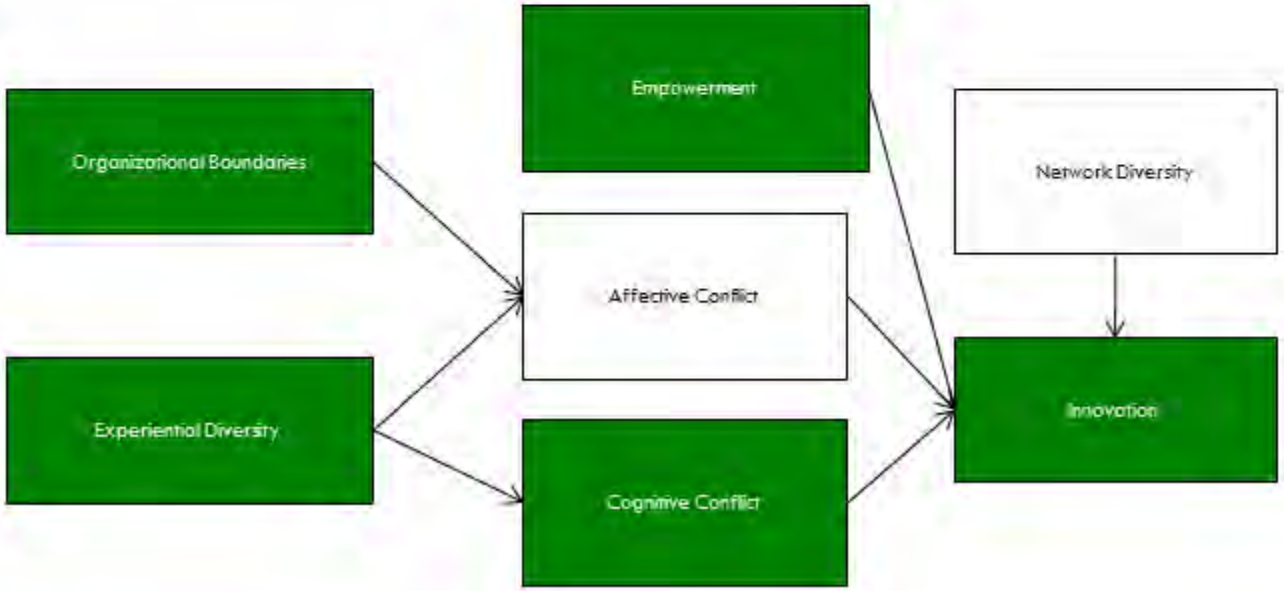
This type of organization encourages empowerment and innovation through diversity of experience in teams and lack of unnecessary organization boundaries.



Organization Options

Multidisciplinary Agile Teams

- Teams have all the skills necessary to achieve their goals and feel empowered
- Cognitive or "good" conflict increases



Organization Options

There are multiple options for organizations that we have seen in practice or worked with clients to leverage for their organization:

- "NoOps" Model which promotes automation heavily
- "Full Stack" Model which promotes end-to-end capability within an Agile team to delivery a service
- "Squads, Chapters & Guilds" Model which promotes autonomy between teams as if they were mini startups while encouraging communication and knowledge sharing across the various groups

No one model normally fits our clients exactly and there are pros and cons to each that need to be considered when thinking of the optimal structure.

Model selection is not an "only choose one" decision . You can utilize elements of all 3 to find the right organization structure for your team.

"NoOps" Model

- The "NoOps" model is most often associated with the Netflix streaming service and its near 100% use of the AWS Cloud.
- High Scalability and fast Time to Market are primary goals of this model. Cost management is secondary.
- Budget and responsibility of figuring out how to use public cloud IaaS was given directly to the Development organization after failed attempts to quickly scale out data centers during periods of extreme growth.
- Relies heavily on near 100% automation for all critical development and releases processes.
- Netflix has a developer-oriented culture starting with the CEO Reed Hastings who was a developer.

CORE Team (Cloud Operations Reliability Engineering)
Responsible for:
- Monitoring
- Application Performance Management

"NoOps" Model

akf PARTNERS

"A handful of DevOps engineers …are coding and running the build tools and bakery, and updating the base AMI from time to time. Several hundred development engineers use these tools to build code, run it in a test account in AWS, then deploy it to production themselves. ***They never have to have a meeting with ITops, or file a ticket asking someone from ITops to make a change to a production system, or request extra capacity in advance****... This is part of what we call NoOps. The developers used to spend hours a week in meetings with Ops discussing what they needed, figuring out capacity forecasts and writing tickets to request changes for the datacenter. Now they spend seconds doing it themselves in the cloud…*

There is no central control, the teams are responsible for figuring out their own dependencies and managing AWS security groups that restrict who can talk to who."

– Adrian Cockcroft
http://perfcap.blogspot.com/2012/03/ops-devops-and-noops-at-netflix.html

CORE Team (Cloud Operations Reliability Engineering)
Responsible for:
- Monitoring
- Application Performance Management

"Full Stack" Model

- Minimizing risk and moving fast are primary goals of this model.
- Teams responsible for services have all of the necessary skill to be innovative and release frequently. They have knowledge of UI/UX, Mid-Tier services, Databases, Hosting environments, Networks, etc. (i.e. "Full Stack" development teams)
- Facebook also has a sizeable Release Engineering team that assists with releases multiple times daily and weekly.
- Facebook Infrastructure Engineering is a separate organization that is responsible for data centers, capacity planning, etc.
- Facebook has a developer-oriented culture starting with the CEO who was a developer.

Release Engineering
Responsible for:
- Controlling releases
- Daily pushes
- Weekly pushes

Infrastructure Engineering
- Data Centers
- Server build out
- Capacity
- Performance
- Incident Management

"Full Stack" Model

akf PARTNERS

"None of the previous principles work ***without*** *operations and engineering teams that work together seamlessly, and can handle problems together as a team. The person responsible for something must have control over it. At Facebook we push code to the site every day, and the person who wrote that code is there to take responsibility for it.*

.. We have three people working on photos. Each of those three people knows photos inside and out, and can make decisions about it. So when something needs to change in photos they get it done quickly and correctly."

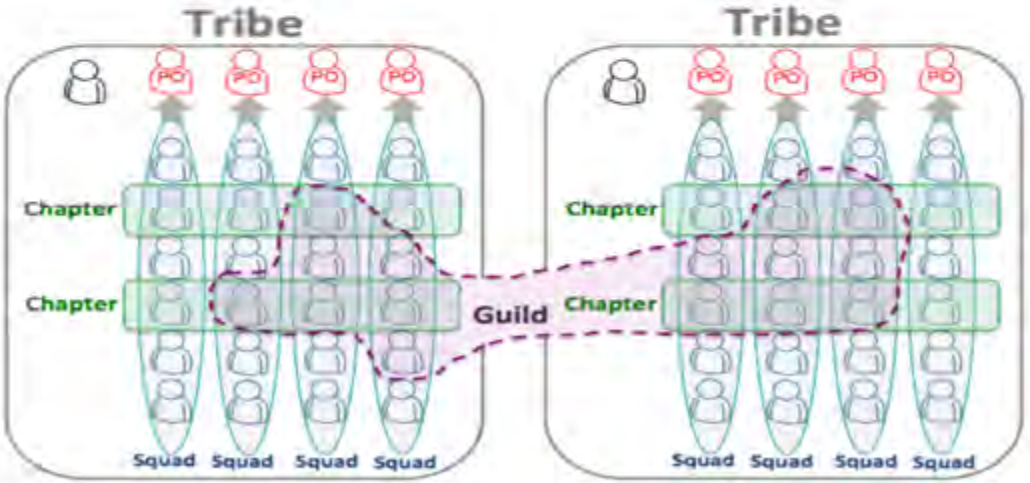
– Peter Deng
https://www.facebook.com/note.php?note_id=409881258919

Release Engineering
Responsible for:
- Controlling releases
- Daily pushes
- Weekly pushes

Infrastructure Engineering
- Data Centers
- Server build out
- Capacity
- Performance
- Incident Management

"Squads, Chapters & Guilds"

Spotify Organizational Structure – A real-world example of a service oriented, cross-functional team can be found at Spotify. Their organizational structure is not functionally aligned but rather is organized into small Agile teams, which they refer to as Squads. These are self-contained teams organized around a deliverable or service which the business provides. The following image and descriptions are taken from Henrik Kniberg and Anders Ivarsson's October 2012 paper "Scaling Agile @ Spotify with Tribes, Squads, Chapters & Guilds.

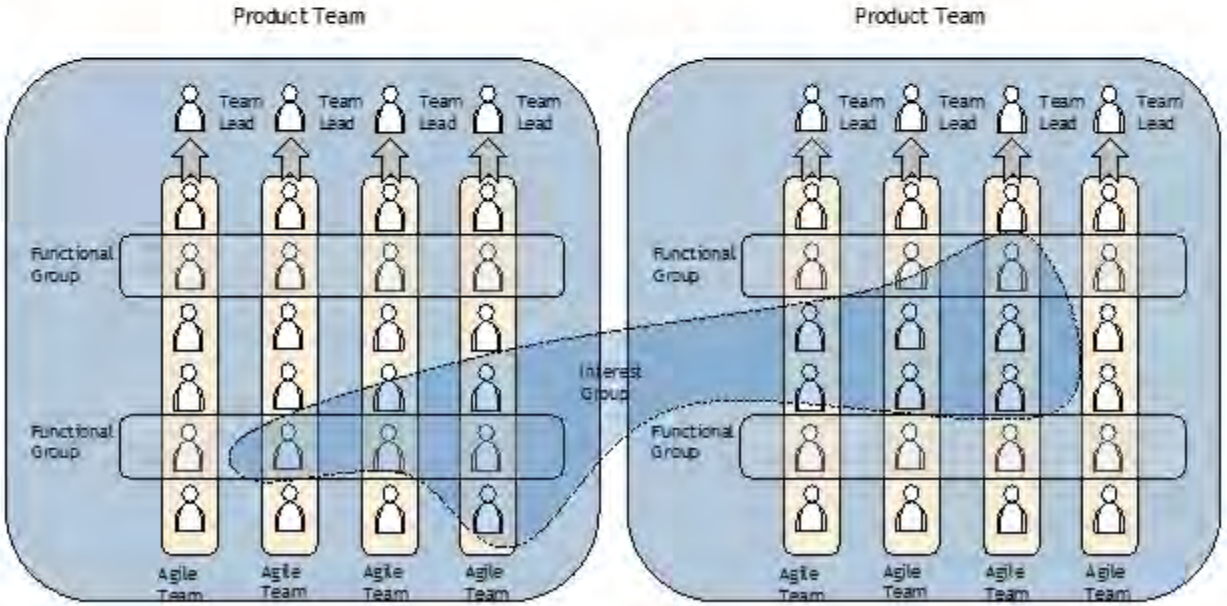


Product & Agile Teams

akf PARTNERS

Product Teams ("Tribes") and Agile Teams ("Squads")

- A Product Team is a collection of Agile Teams that work in related areas. The Product Team can be seen as the "incubator" for the mini-startups focused on services.
- Each Product Team has a lead who is responsible for providing the best possible habitat for the Agile Teams within that Product Team.
- Agile Teams within a Product Team are all physically in the same office, normally right next to each other.
- Product Teams are less than 100 people in total.

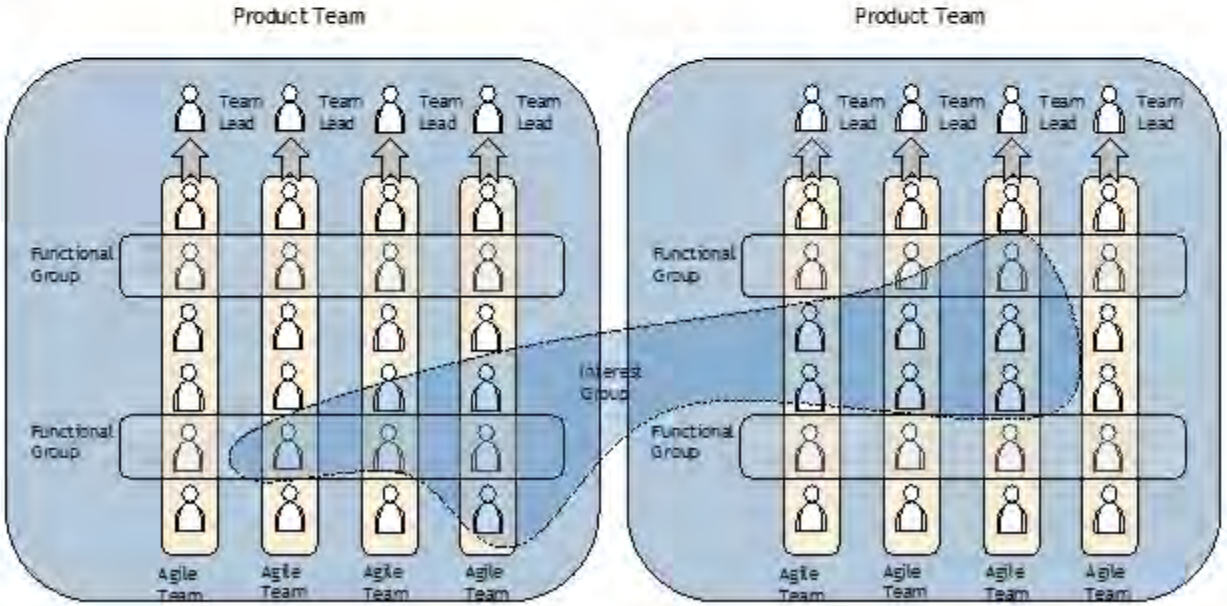


Functional Groups

akf PARTNERS

Functional Groups ("Chapters")

- Small group of people having similar skills and working within the same general competency area, within the same Product Team.
- Each Funtional Group meets regularly to discuss their area of expertise and their specific challenges.
- The Functional Group lead is a line manager for her Funtional Group members, with all the traditional responsibilities such as developing people, setting salaries, etc.
- However, the Functional Group lead is also part of a Agile Team and is involved in the day-to-day work, which helps her stay in touch with reality.

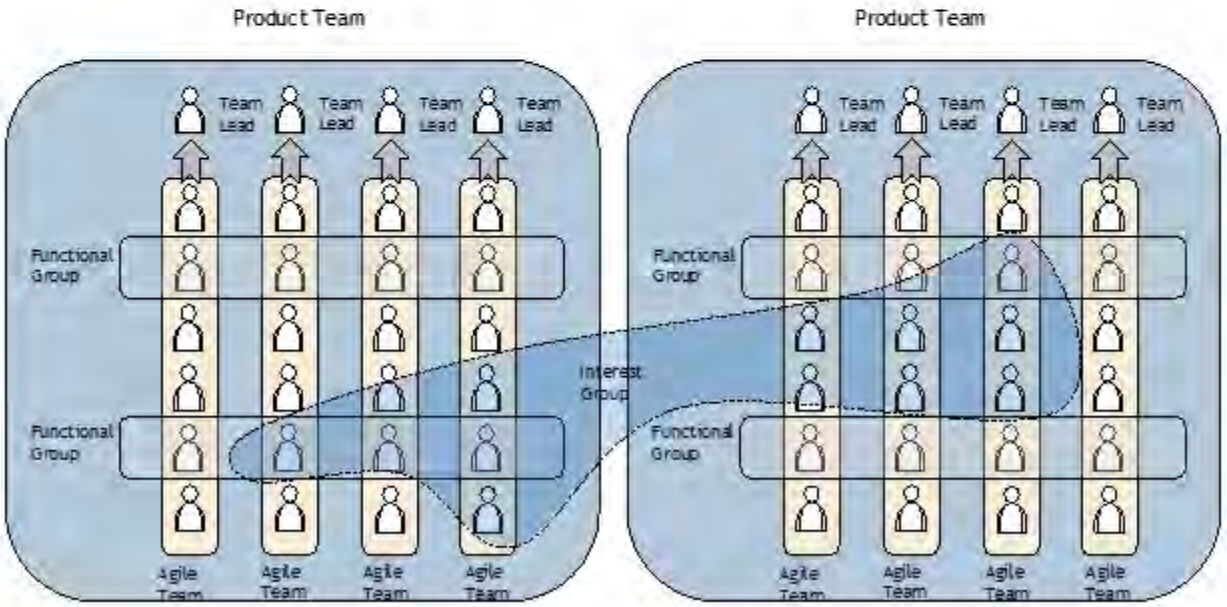


Interest Groups

akf PARTNERS

Interest Groups ("Guilds")

- An Interest Group is a more organic and wide reaching "community of interest", a group of people that want to share knowledge, tools, code, and practices.
- Functional Groups are always local to a Product Team, while an Interest Group usually cuts across the whole organization. Some examples are: the web technology guild, the tester guild, the agile coach guild, etc.



Key Takeaways

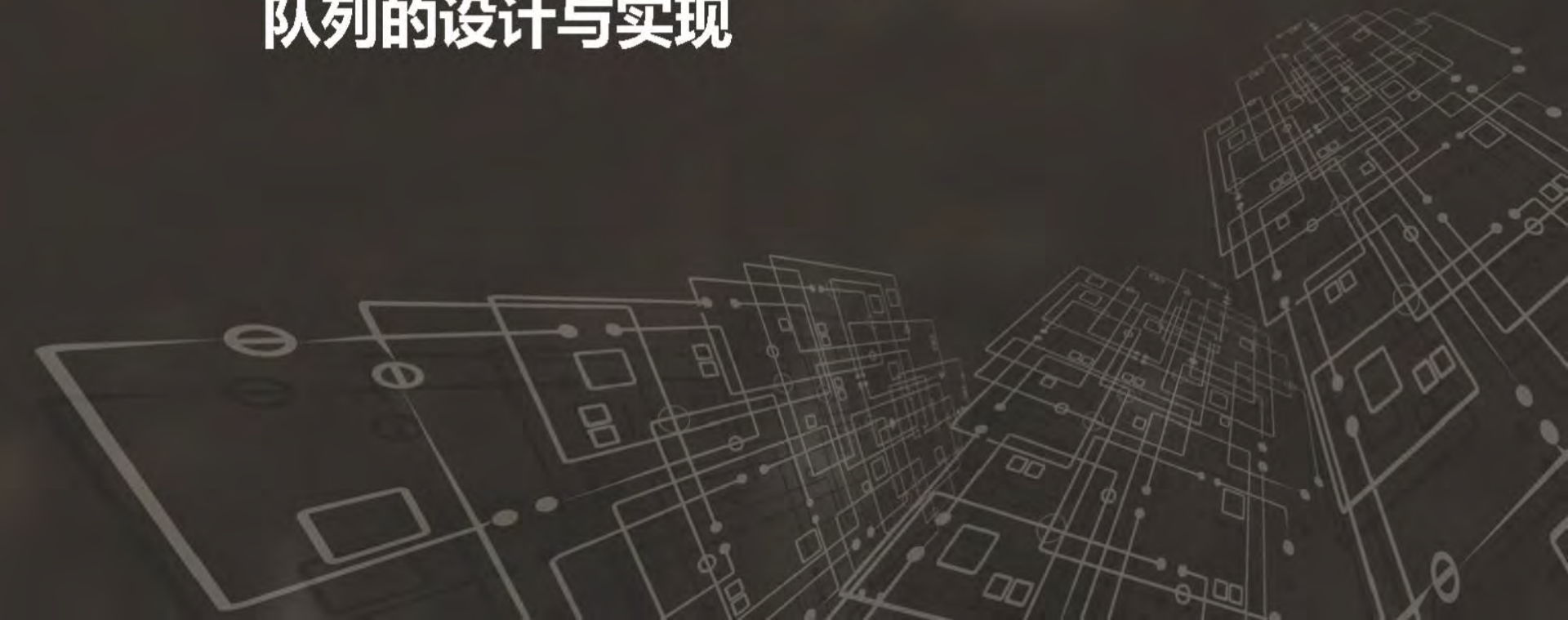
akf PARTNERS

For fastest Time to Market, Best Quality, Best Organizational Scale and Highest Levels of Innovation, teams must:

- Cross functional – have all the skills embedded within the team to do the job they are asked to do.
- Need very little communication with other teams to do their job.
- Work against outcomes – not requirements or tasks
- Own the outcomes AND the solutions to achieve them – no shared ownership of solutions (software and hardware)

2017 Software Architecture Summit

PhxQueue
微信开源高可用强一致分布式队列的设计与实现



自我介绍

- 梁俊杰 Unixliang
- 微信 基础消息组
- 2011年11月加入腾讯
  - 曾负责 微博
    - 私信
    - 反垃圾系统
  - 现负责 微信
    - 消息系统
    - 跨域同步组件
    - 队列中间件

听众受益

- 了解微信后台分布式消息队列的架构演进
- 了解 PhxQueue 的功能特性
- 了解 PhxQueue 设计与实现细节
- 了解微信后台消息队列最佳实践