Geekbang 极客邦科技 InfoQ

QCon 全球软件开发大会【北京站】2016

微服务，你玩的起吗？

- 勿在浮沙筑高台

禚娴静

ThoughtWorks

Intemational Software Development Conference

微服务架构

微服务架构是一种架构模式，它提倡将单一应用程序划分成一组小的服务，服务之间互相协调、互相配合，为用户提供最终价值。每个服务运行在其独立的进程中，服务与服务间采用轻量级的通信机制互相沟通（通常是基于HTTP协议的RESTful API）。每个服务都围绕着具体业务进行构建，并且能够被独立的部署到生产环境、类生产环境等。另外，应当尽量避免统一的、集中式的服务管理机制，对具体的一个服务而言，应根据业务上下文，选择合适的语言、工具对其进行构建。

- Martin Fowler

为什么要采用微服务架构?

初衷

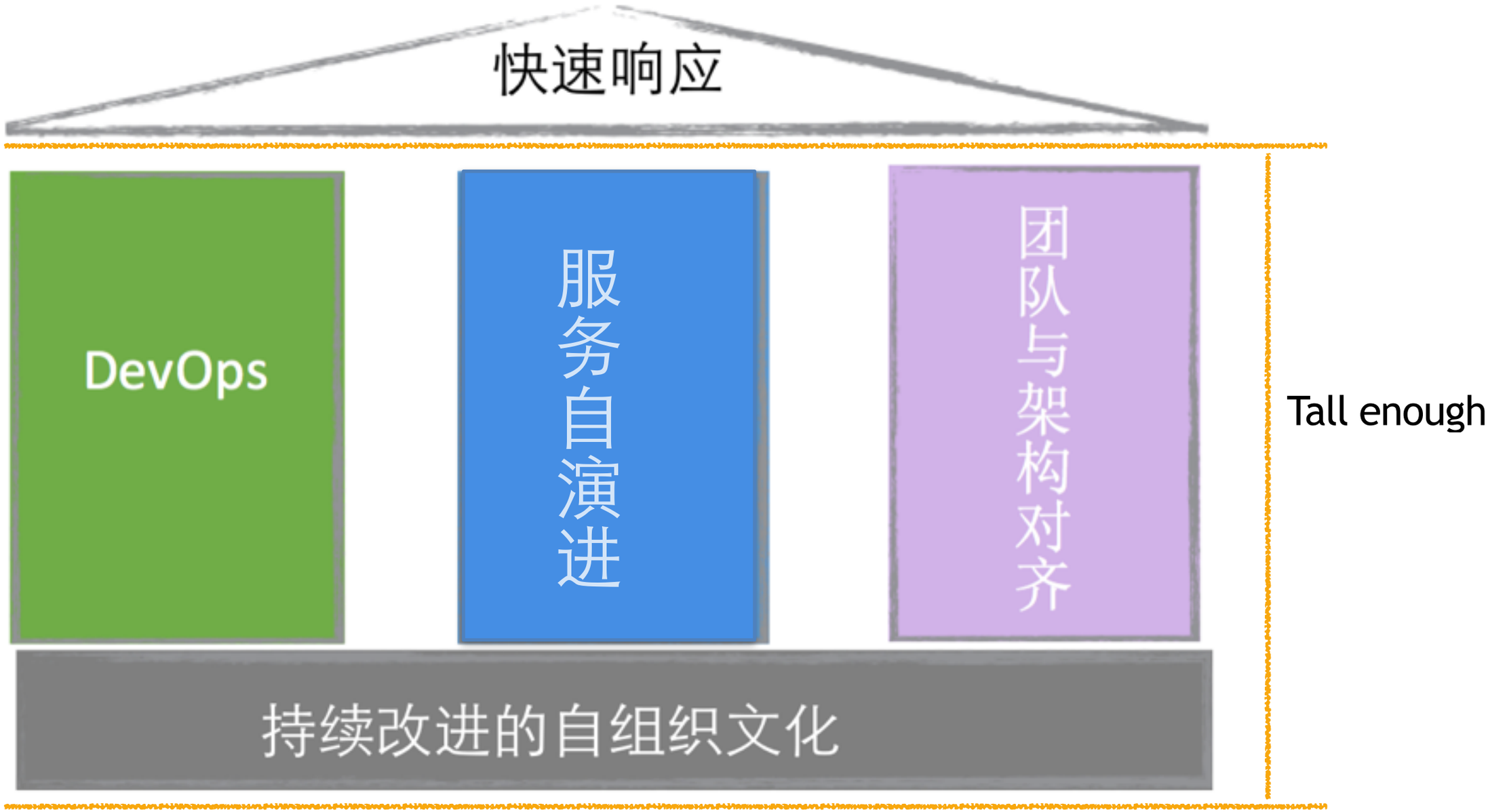
> 让系统尽可能快的响应变化！

- Rebecca Parsons

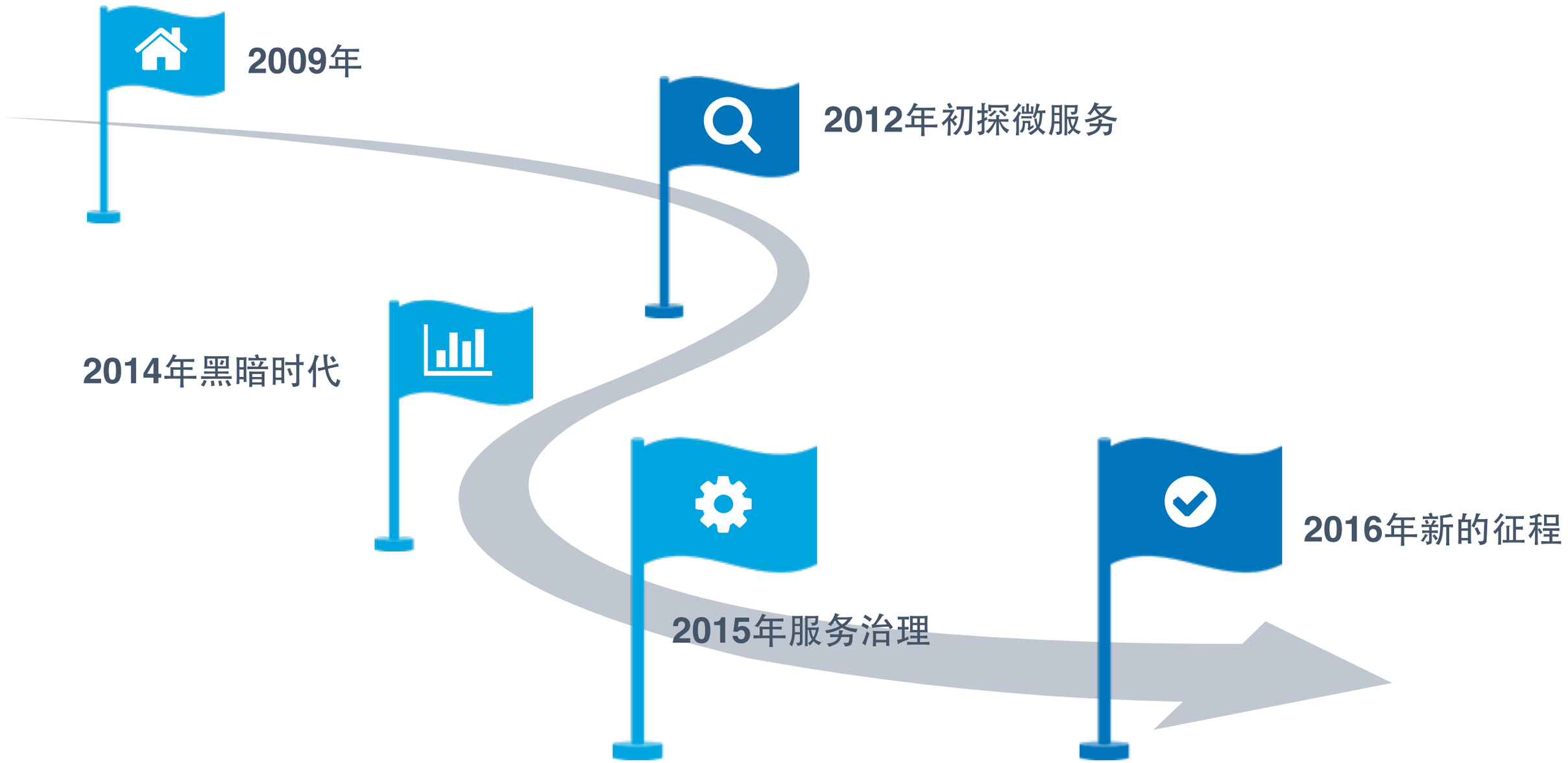
- 服务足够小
- 独立运行
- 轻量级通信机制
- 独立的部署
- 去中心化

- 要多小？
- 怎么部署？
- 出错怎么办？
- 如何保证一致性

玩得起微服务的条件

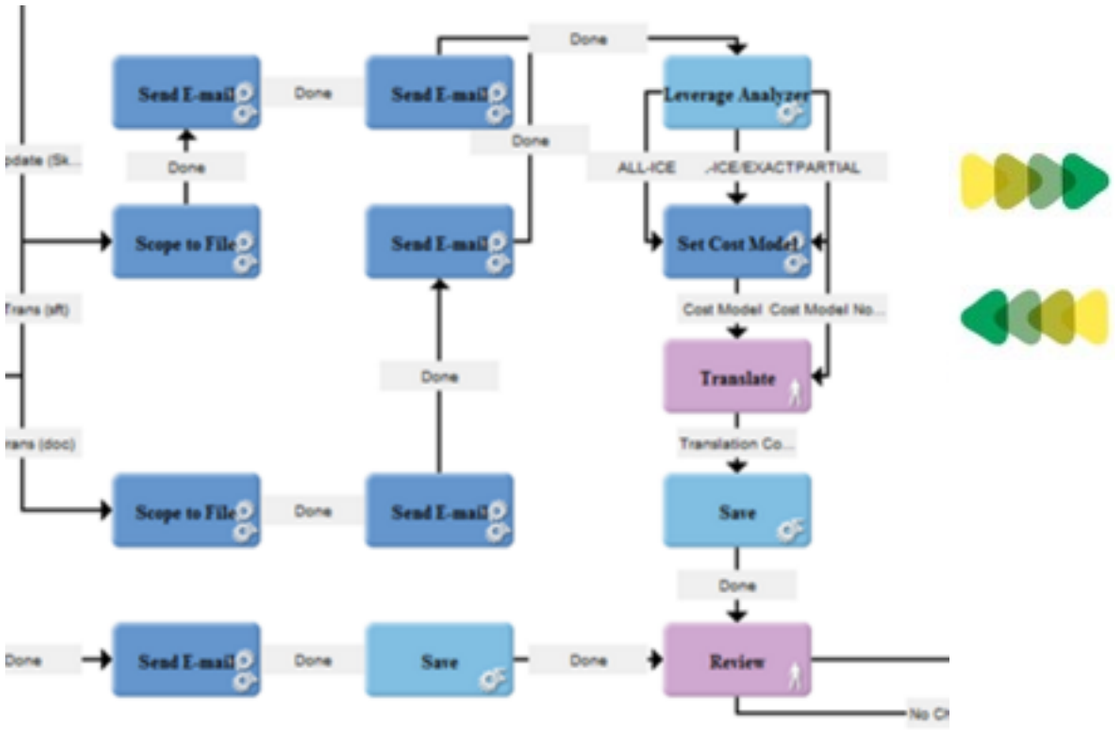


背景介绍



2012年新的需求

业务系统A



业务系统B

重复?

业务系统A

- 模块A
- 模块B
- 模块C

重用服务

新的业务系统A

业务系统B

- 模块A
- 模块B
- 模块C
- 模块D

领域服务

微服务架构初步形成

业务系统A

业务系统B

服务E

服务A

服务B

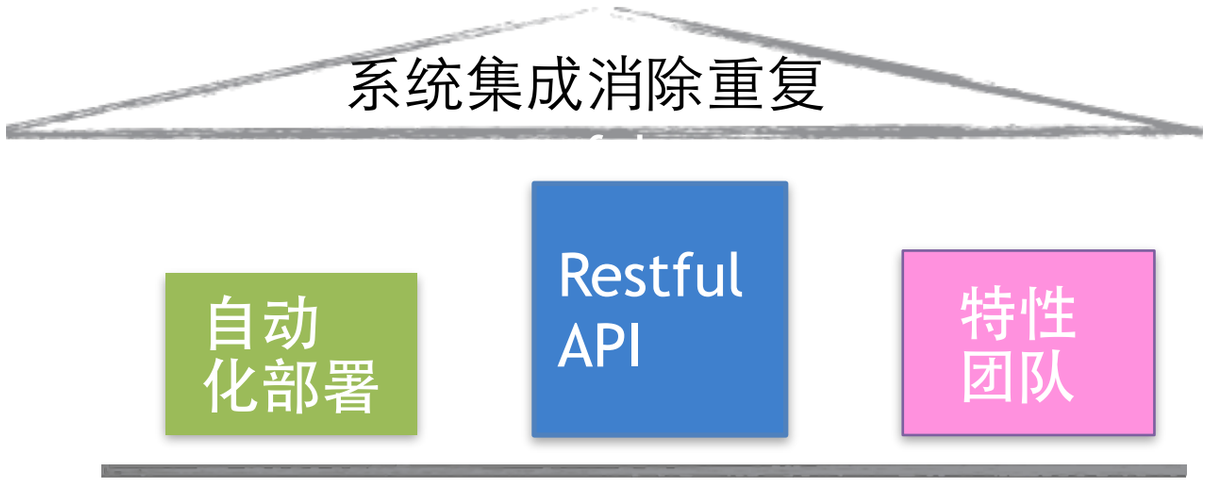
服务F

服务C

服务D

领域服务

那时的微服务



它给我们带来了哪些好处?

新业务C三个月上线

业务系统A

业务系统C

业务系统B

服务E　服务A　服务B　服务F

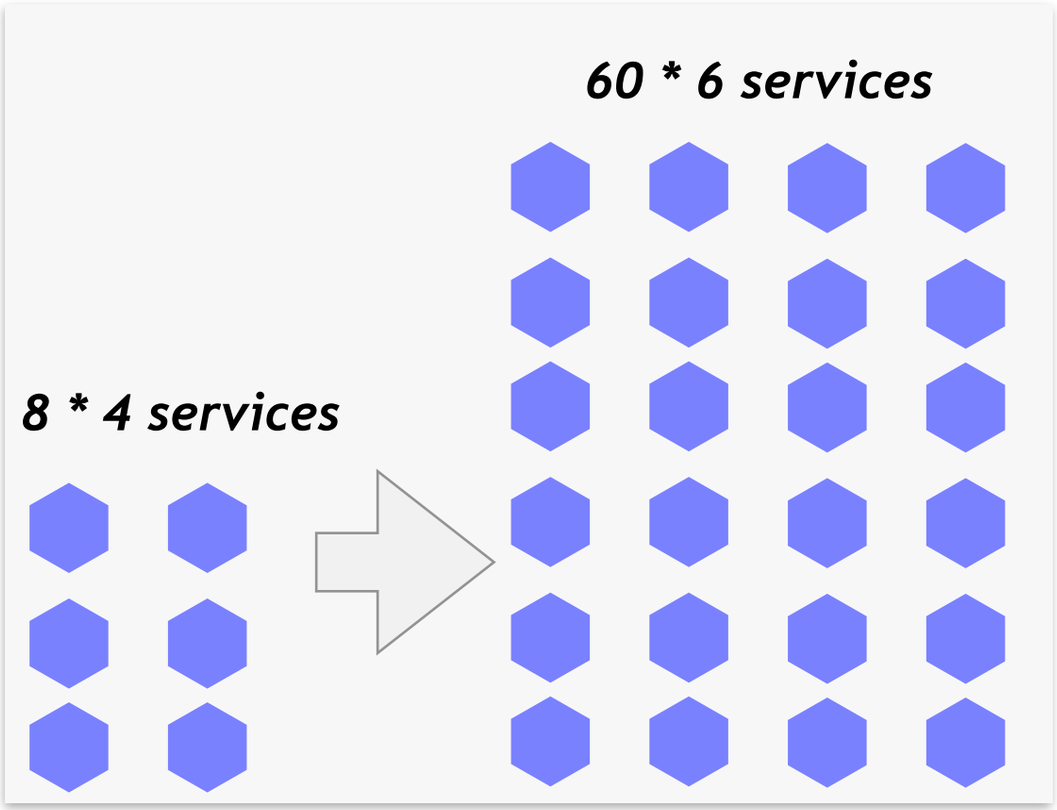
服务H　服务C　服务D　服务G

领域服务

你是我们见过最棒的团队！

更多服务更快一点?

服务越来越多，周期越来越短



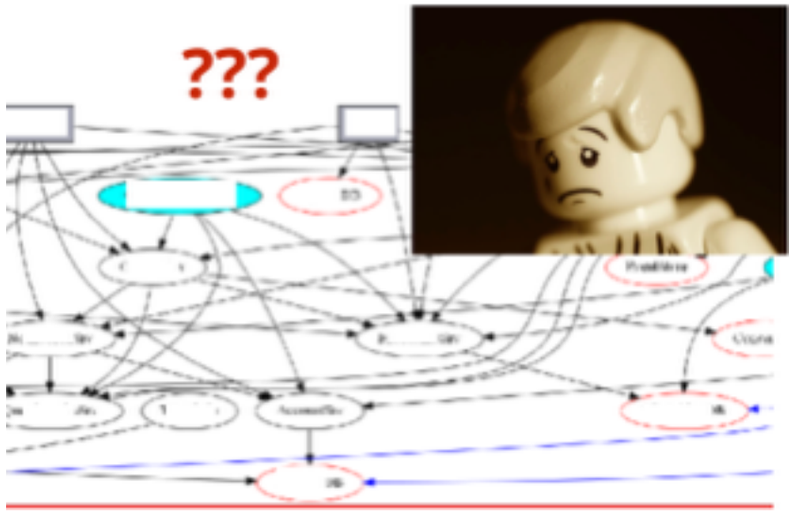
2014年问题倍出

1 环境手工维护，频频出错

部署成功率很低，部署时经常有一堆环境修改需求，运维人员出错机会增加，运维效率极低。

2 缺乏有效监控

无法快速有效定位问题，无法快速有效知晓服务运行状态，服务资源浪费。



2014年问题倍出（二）

3 服务过大，堵塞交付

快速增长的结果导致服务过大或者服务过小。而过大的服务导致整个提交流水线堵塞，测试人员无法拿到新的版本，交付延期

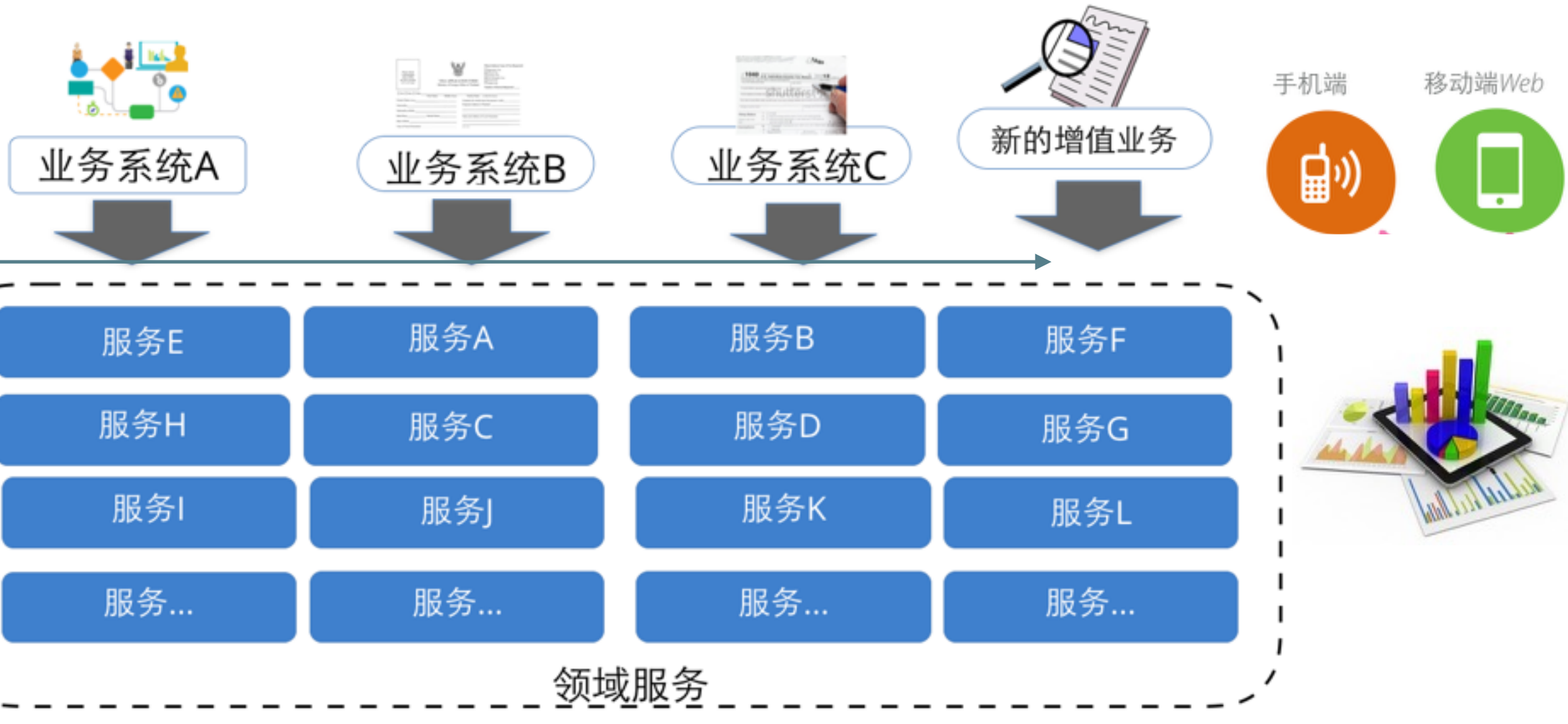
4 团队出现冲突，架构腐化严重

交付不能完成导致各角色间的冲突越来越严重，为了快速上线质量遭到牺牲。架构无人守护，各种不一致性，服务内部接口一片混乱，核心人员离职。

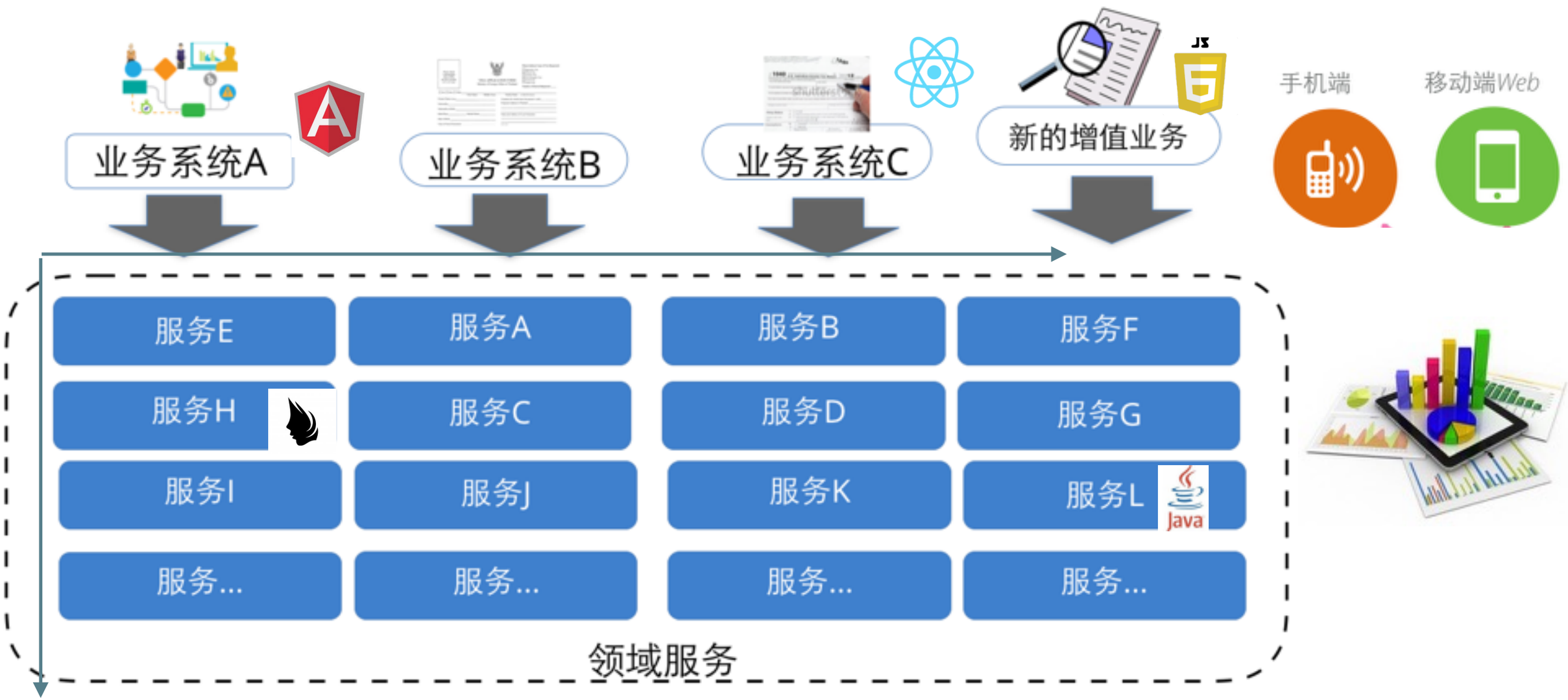
不要再添加任何服务！

当然不是！

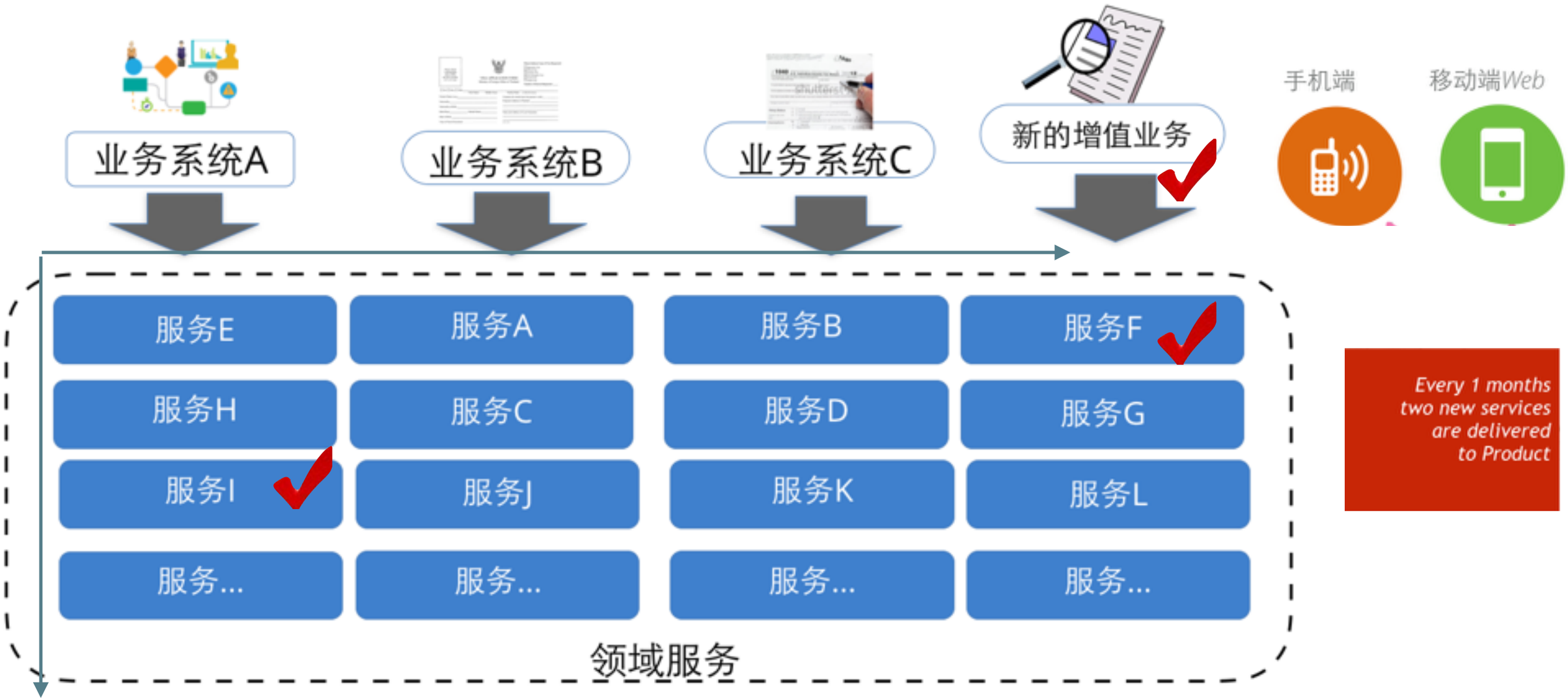
业务迅速扩张


业务系统A
业务系统B
业务系统C
新的增值业务
手机端
移动端Web
服务E
服务A
服务B
服务F
服务H
服务C
服务D
服务G
服务I
服务J
服务K
服务L
服务...
服务...
服务...
服务...
领域服务

技术选型灵活

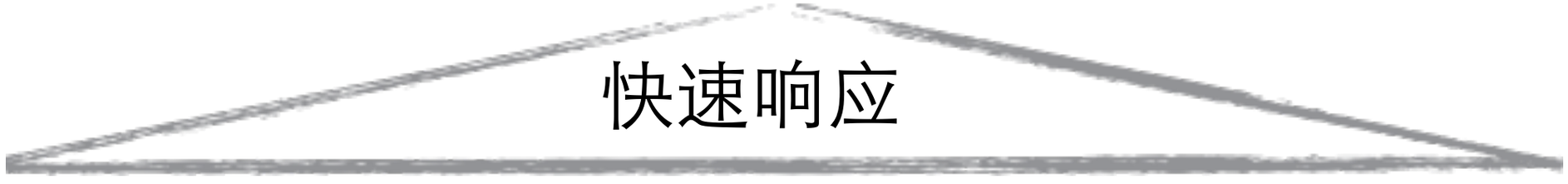

业务系统A
业务系统B
业务系统C
新的增值业务
手机端
移动端Web
服务E
服务A
服务B
服务F
服务H
服务C
服务D
服务G
服务I
服务J
服务K
服务L
服务...
服务...
服务...
服务...
领域服务

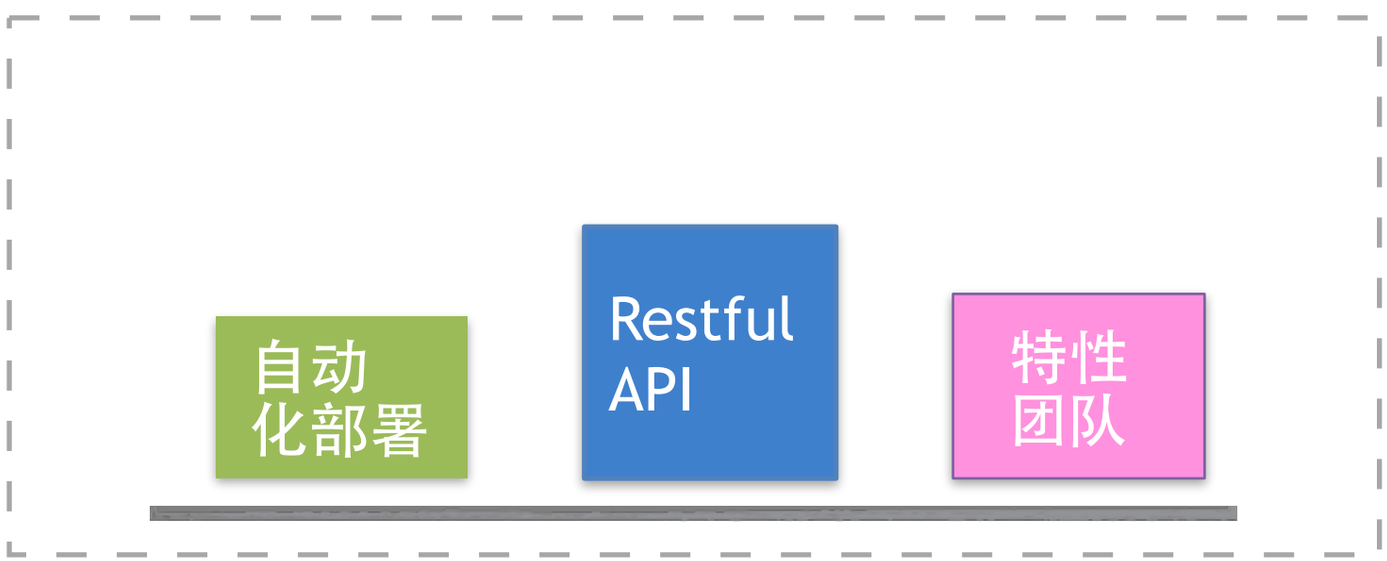
助力客户快速占领市场


业务系统A
业务系统B
业务系统C
新的增值业务
手机端
移动端Web
服务E
服务A
服务B
服务F
服务H
服务C
服务D
服务G
服务I
服务J
服务K
服务L
服务...
服务...
服务...
服务...
领域服务
Every 1 months two new services are delivered to Product

那怎么办?

2015年服务治理





只有这些是不够的

回到问题

1 环境手工维护，频频出错

部署成功率很低，部署时经常有一堆环境修改需求，运维人员出错机会增加，运维效率极低。

2 缺乏有效监控

无法快速有效定位问题，无法快速有效知晓服务运行状态，服务资源浪费。

3 服务过大，堵塞交付

快速增长的结果导致服务过大或者服务过小。而过大的服务导致整个提交流水线堵塞，测试人员无法拿到新的版本，交付延期

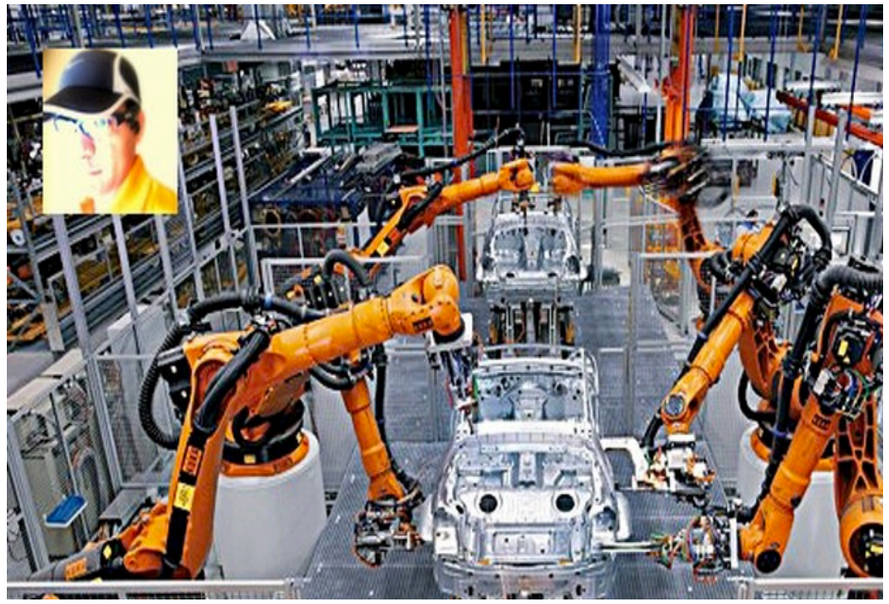
4 团队出现冲突，架构腐化严重

交付不能完成导致各角色间的冲突越来越严重，为了快速上线质量遭到牺牲。架构无人守护，各种不一致性，服务内部接口一片混乱，核心人员离职。

1.运维自动化

1 基础设施自动化

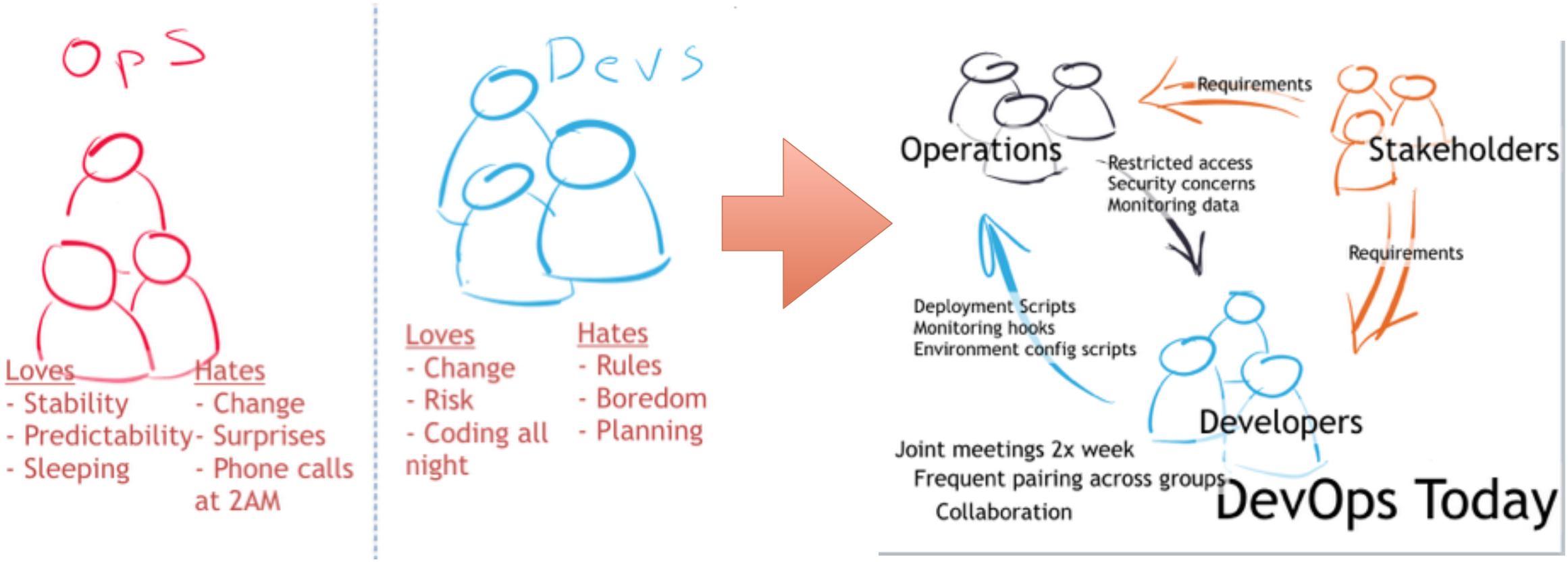
实施了新的部署流程，成功率大大提高，部署时间缩短到30分钟。



2 高效监控

服务提供状态汇报，利用Splunk聚合日志，对服务运行状态进行监控，大大提高运维效率。

2.DevOps一家亲



2012

2015

回到问题

环境手工维护，频频出错

部署成功率很低，部署时经常有一堆环境修改需求，运维人员出错机会增加，运维效率极低。

缺乏有效监控

无法快速有效定位问题，无法快速有效知晓服务运行状态，服务资源浪费。

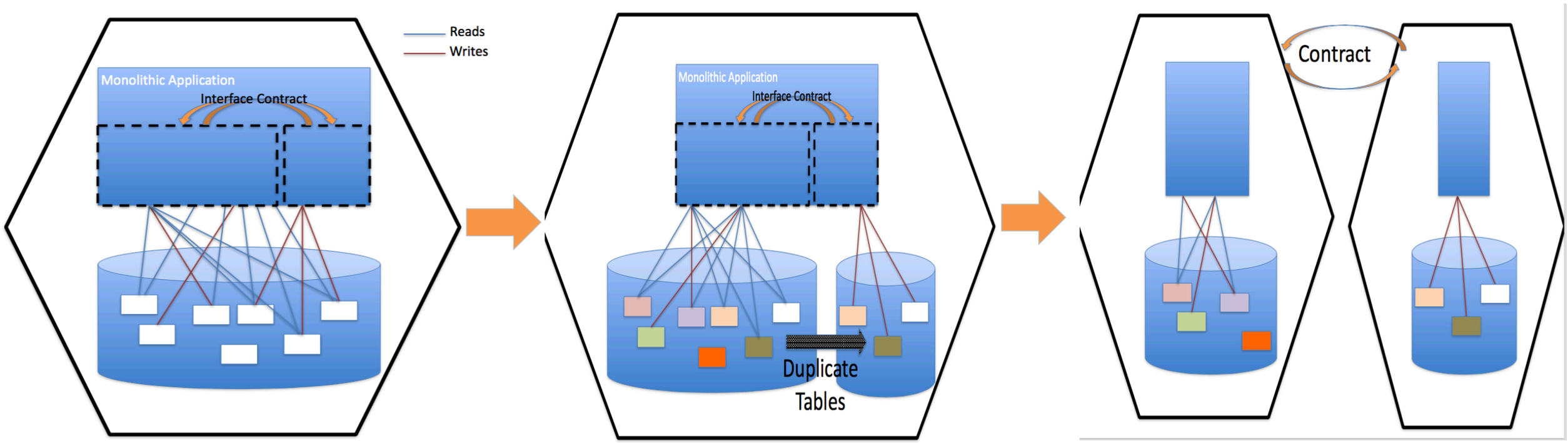
3 服务过大，堵塞交付

快速增长的结果导致服务过大或者服务过小。而过大的服务导致整个提交流水线堵塞，测试人员无法拿到新的版本，交付延期

4 团队出现冲突，架构腐化严重

交付不能完成导致各角色间的冲突越来越严重，为了快速上线质量遭到牺牲。架构无人守护，各种不一致性，服务内部接口一片混乱，核心人员离职。

3.服务拆分

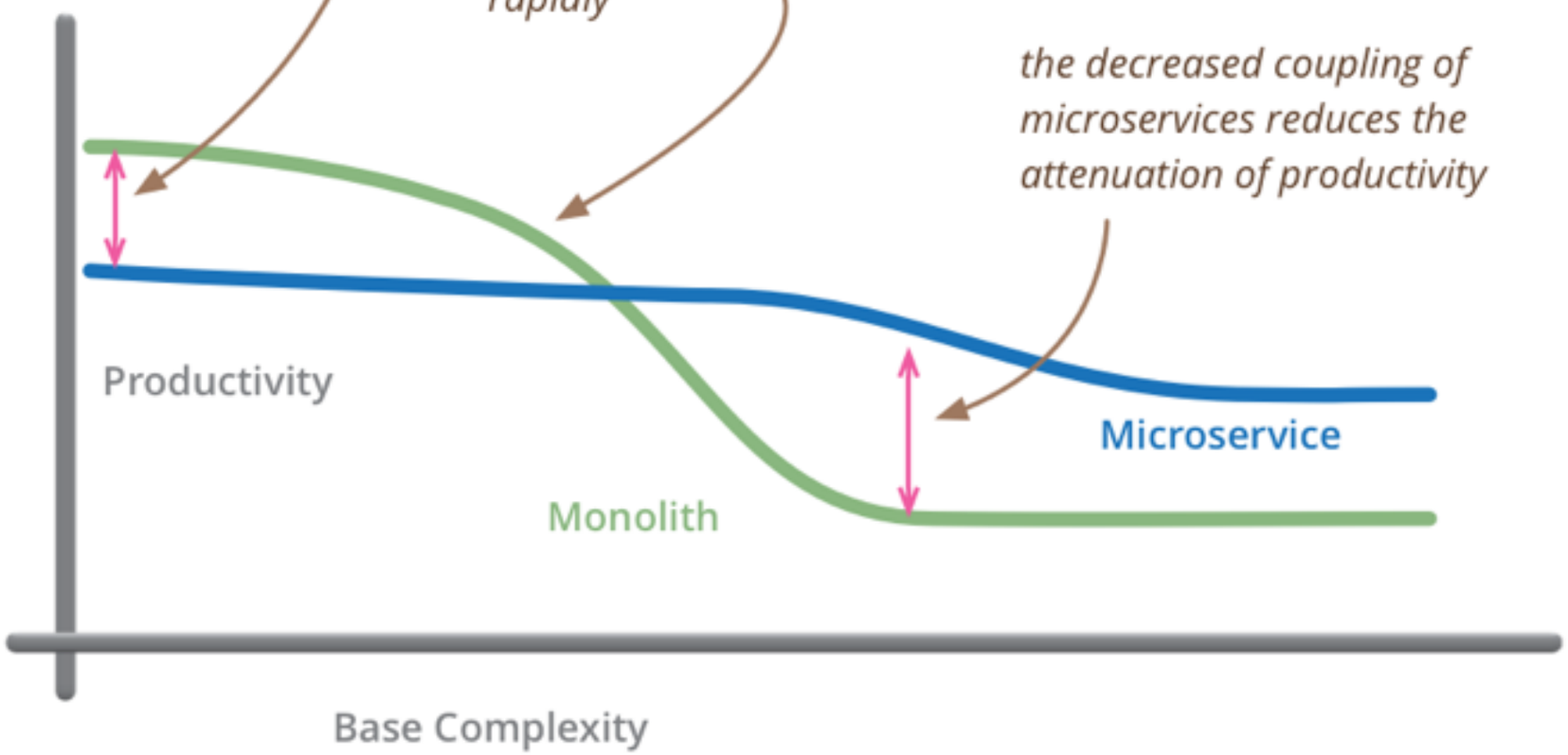

Reads
Writes
Monolithic Application
Interface Contract
Monolithic Application
Interface Contract
Duplicate
Tables
Contract

微服务的附加成本

for less-complex systems, the extra baggage required to manage microservices reduces productivity

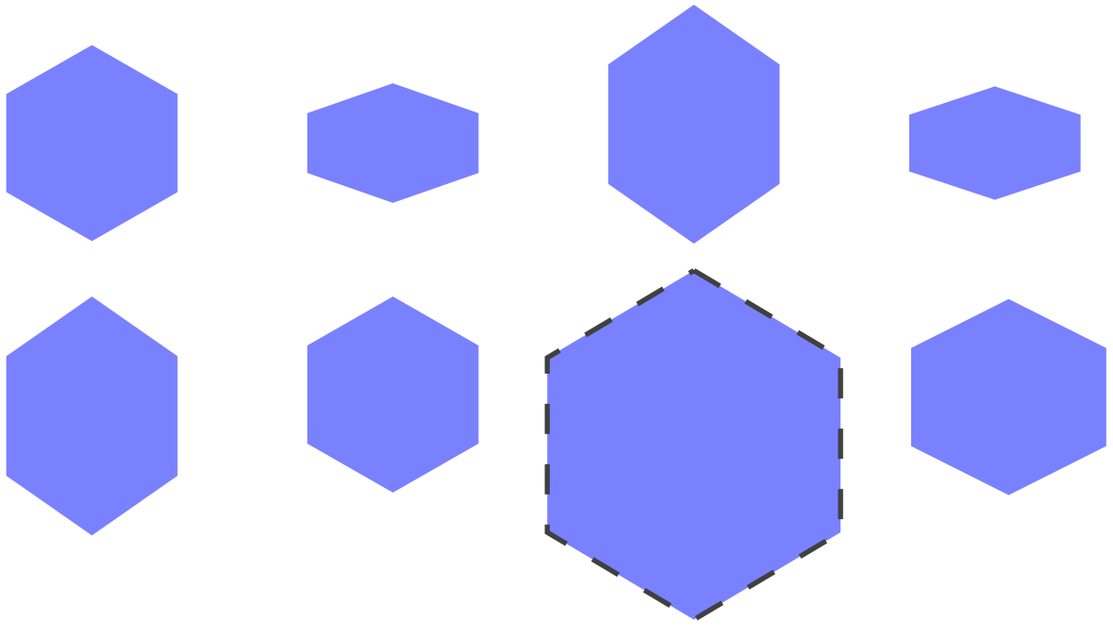
as complexity kicks in, productivity starts falling rapidly

the decreased coupling of microservices reduces the attenuation of productivity



but remember the skill of the team will outweigh any monolith/microservice choice

3.服务自演进



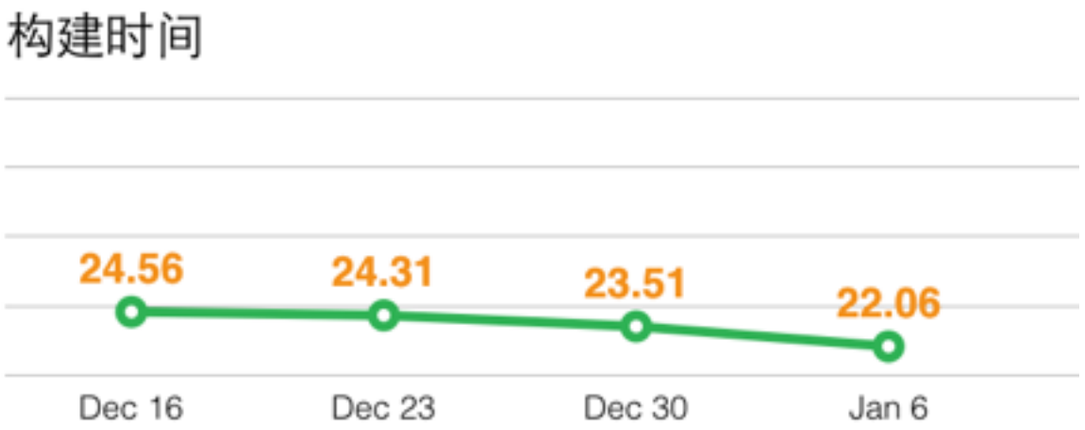
1 划分合适的业务边界

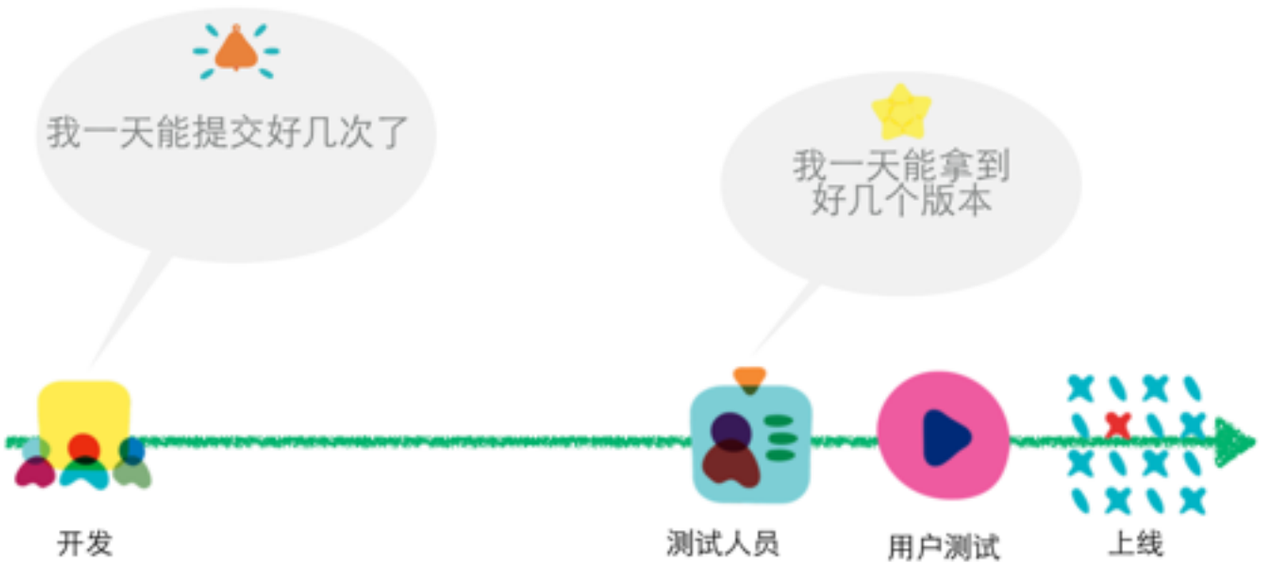
2 进行合适模块化

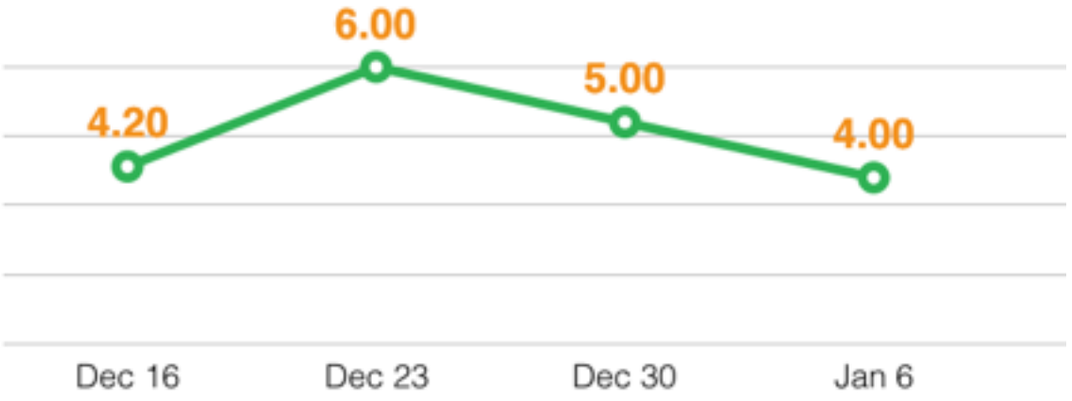
3 可测试的

4 拒绝跨上下文的数据表连接

交付畅通


构建时间
24.56
24.31
23.51
22.06
Dec 16
Dec 23
Dec 30
Jan 6


我一天能提交好几次了
我一天能拿到
好几个版本
开发
测试人员
用户测试
上线


平均每天新版本数量
4.20
6.00
5.00
4.00
Dec 16
Dec 23
Dec 30
Jan 6


这次竟然提前完成了!
项目经理

回到问题

环境手工维护，频频出错

部署成功率很低，部署时经常有一堆环境修改需求，运维人员出错机会增加，运维效率极低。

缺乏有效监控

无法快速有效定位问题，无法快速有效知晓服务运行状态，服务资源浪费。

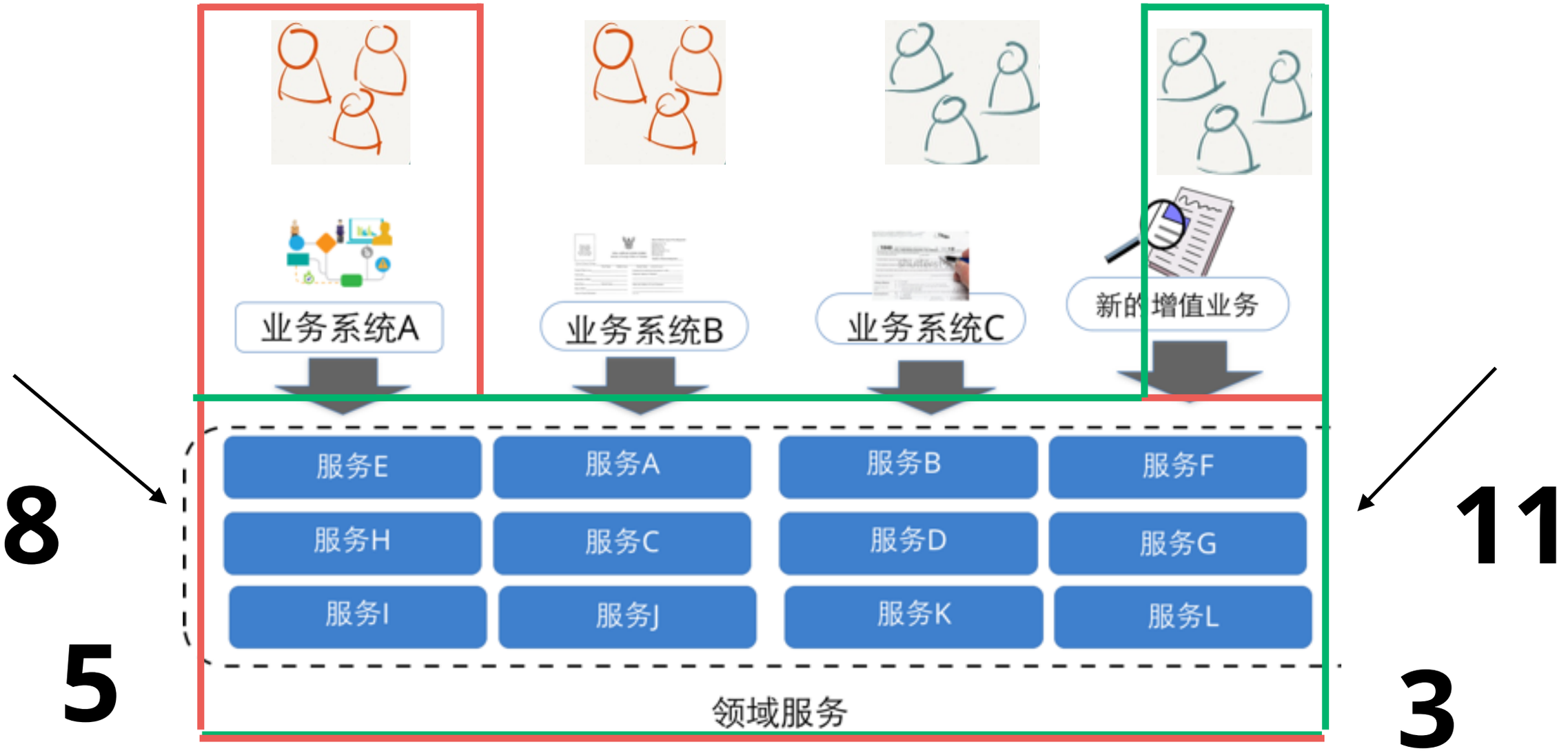
服务过大，堵塞交付

快速增长的结果导致服务过大或者服务过小。而过大的服务导致整个提交流水线堵塞，测试人员无法拿到新的版本，交付延期

4 团队出现冲突，架构腐化严重

交付不能完成导致各角色间的冲突越来越严重，为了快速上线质量遭到牺牲。架构无人守护，各种不一致性，服务内部接口一片混乱，核心人员离职。

所有人?


业务系统A
业务系统B
业务系统C
新的增值业务
服务E
服务A
服务B
服务F
服务H
服务C
服务D
服务G
服务I
服务J
服务K
服务L
领域服务
8
5
11
3

康威定律

> 设计一个系统的任何组织所产生的设计和架构都等价于其组织间的沟通结构。

—Melvyn Conway, 1967

康威逆定律

> 逐渐改进你的团队和组织结构来促进你所渴望的软件系统架构。
>
> —Sam Newman

微服务架构特性

- **服务足够小**
- **独立运行**
- **轻量级通信机制**
- **独立的部署**
- **去中心化**

新的团队结构

交付特性团队

交付特性团队

业务系统A

业务系统B

业务系统C

新的增值业务

API设计指导原则

Perf Guild

服务E

服务A

服务B

服务F

服务H

服务C

服务D

服务G

服务I

服务J

服务K

服务L

领域服务

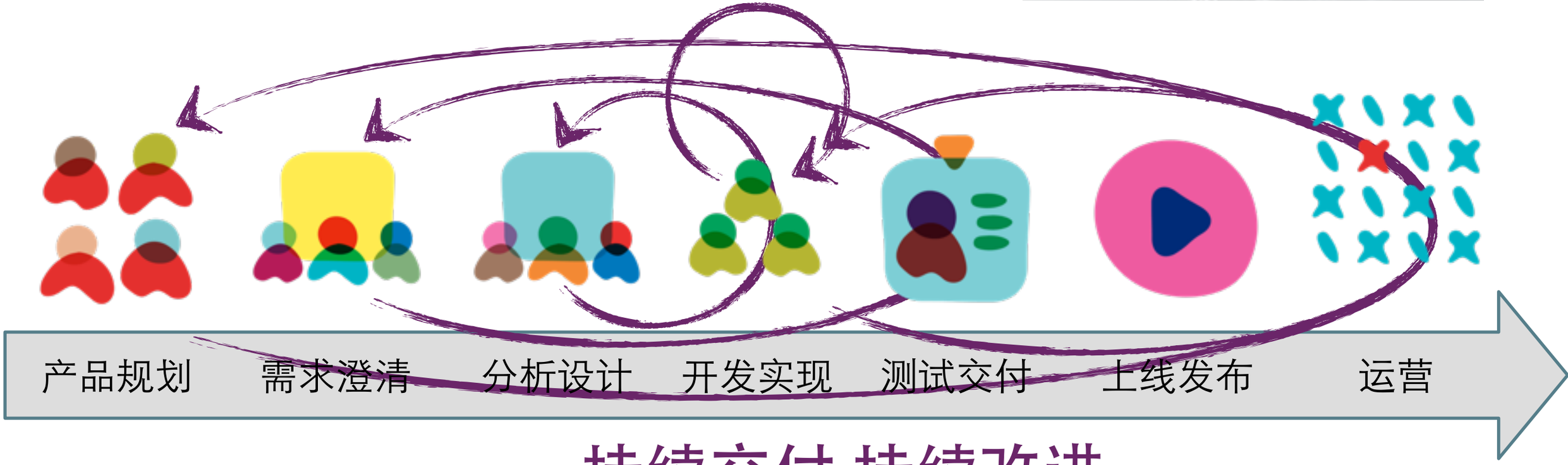
容错性

DevOps Guild

“没有衡量就没有改善！
你衡量什么就得到什么！”

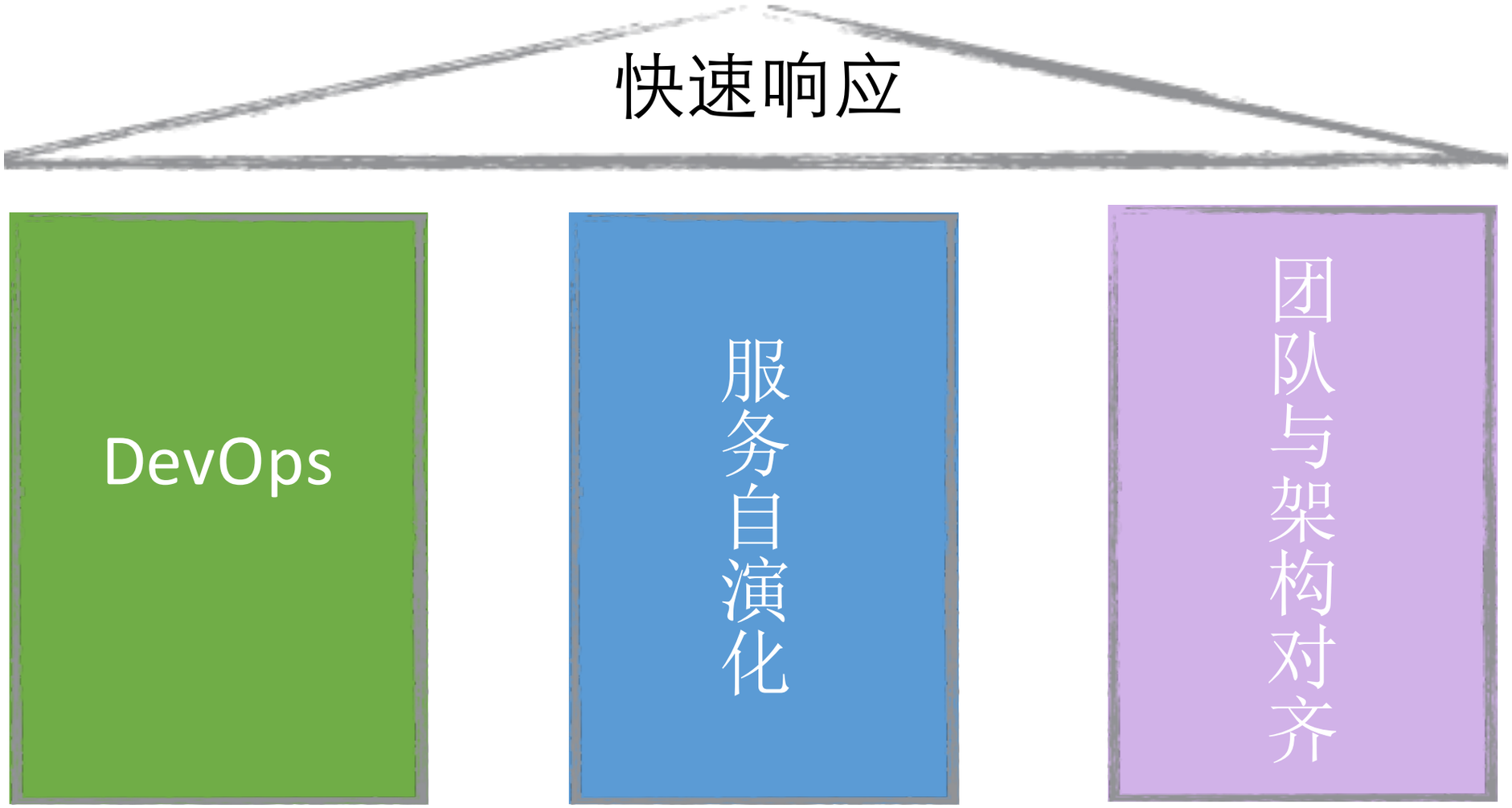
—Peter F. Drucker

建立闭环反馈系统

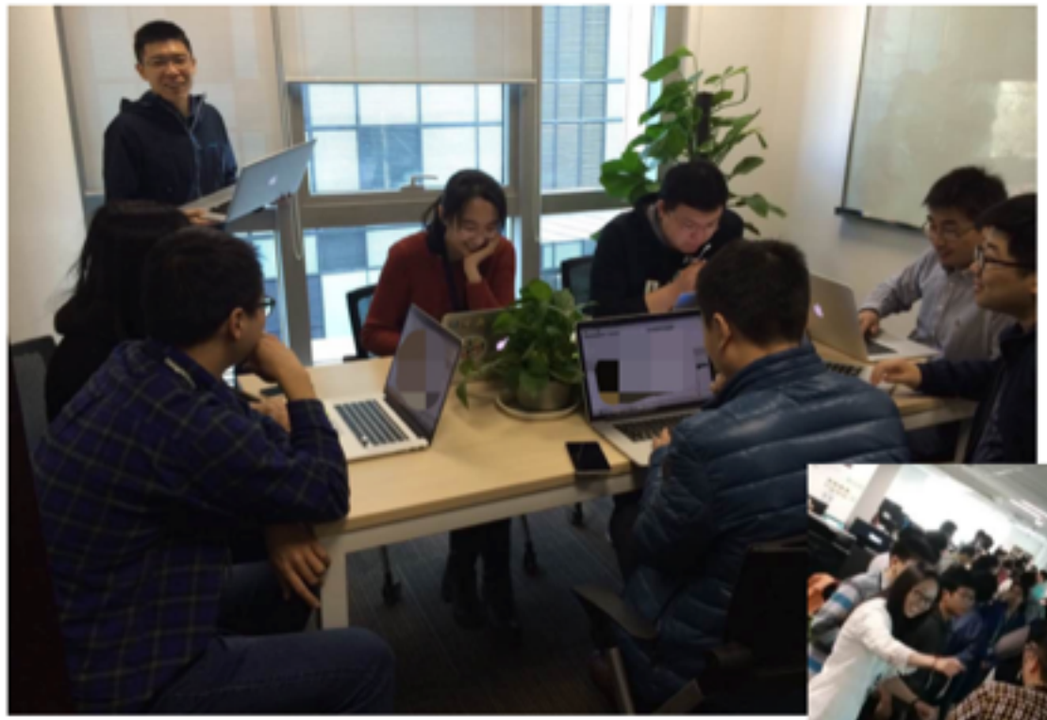


持续交付 持续改进

这样就足够了吗?



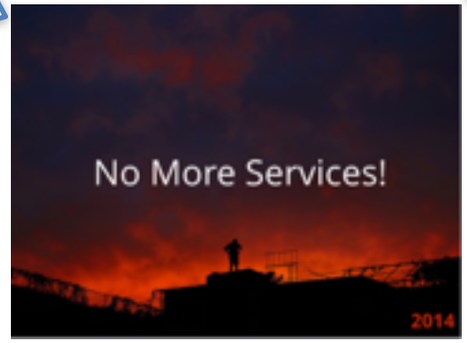
一个具有快速复原能力，一个具有持续改进自组织文化的团队！



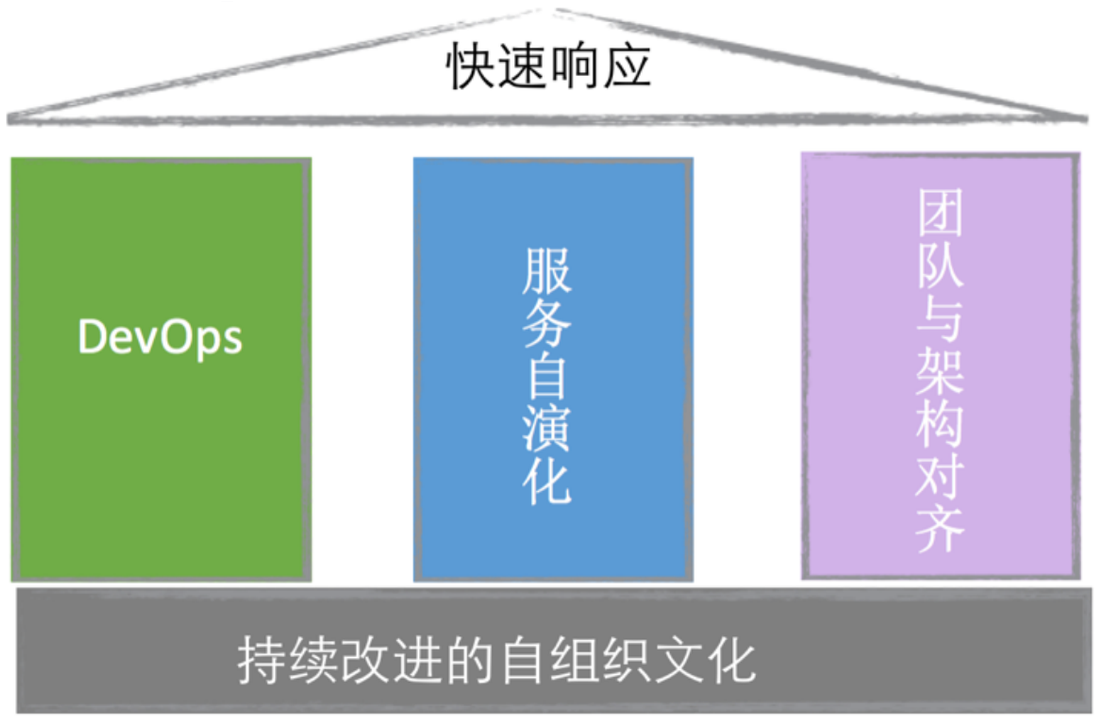

饭后
珑麒麟
"饭后"是一群程序员在中...
极客说
互联网
企业应用
订阅专辑
批量下载
相关推荐
上次播到:#4 解散DevOps
共4集
选集
排序
#4 解散DevOps
6天前
#3 微服务阴谋论
28天前
#2 读书雷达
1个月前
2015-09
Podcast-饭后
每周一博
技术大会
GUI Client
UI Server
Stock Service
Workgroup
request Context
Login Controller
DSP
ARM
Repository
xfer repo
Style Repository
Data Store
Possibly NoSQL
Store Sales Person
Search by attribute or description or other item
Select Color & size
View styles with item available
Execute transfer request
Enter style ID code

持续改进的自组织文化

2012

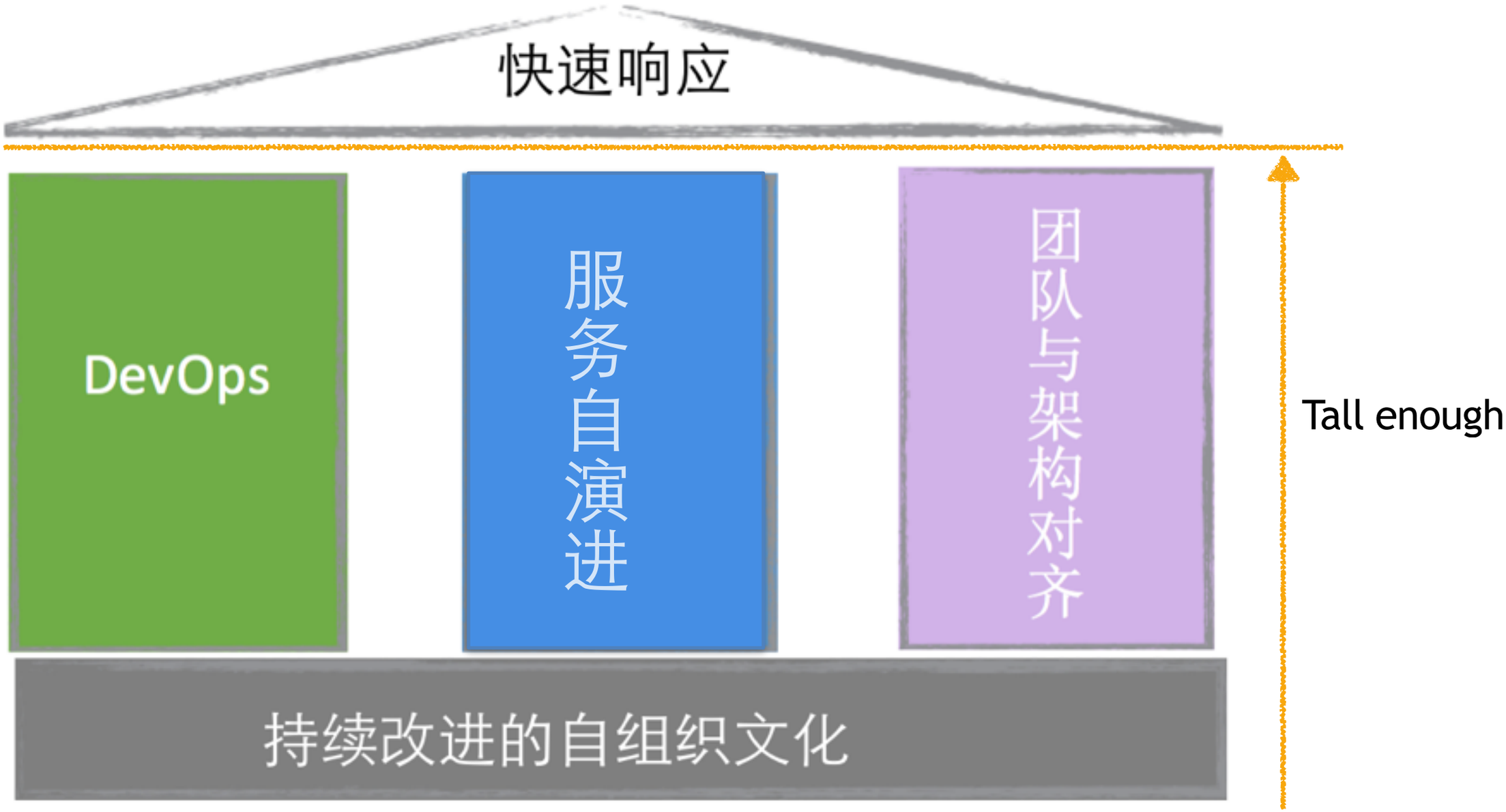


2014



2015

总结



2016新的挑战已然开始...

Geekbang 极客邦科技　InfoQ

THANKS!

禚娴静

xjzhuo@thoughtworks.com

MORE THAN TECHNOLOGY

2016技术雷达峰会

和真正的技术大牛讨论实际的技术问题

May. 07(周六)

北京朝阳

皇冠假日酒店

50% 优惠 COUPON

优惠码：QCON娴静

此优惠码可用于全天VIP门票

扫我报名

ThoughtWorks®