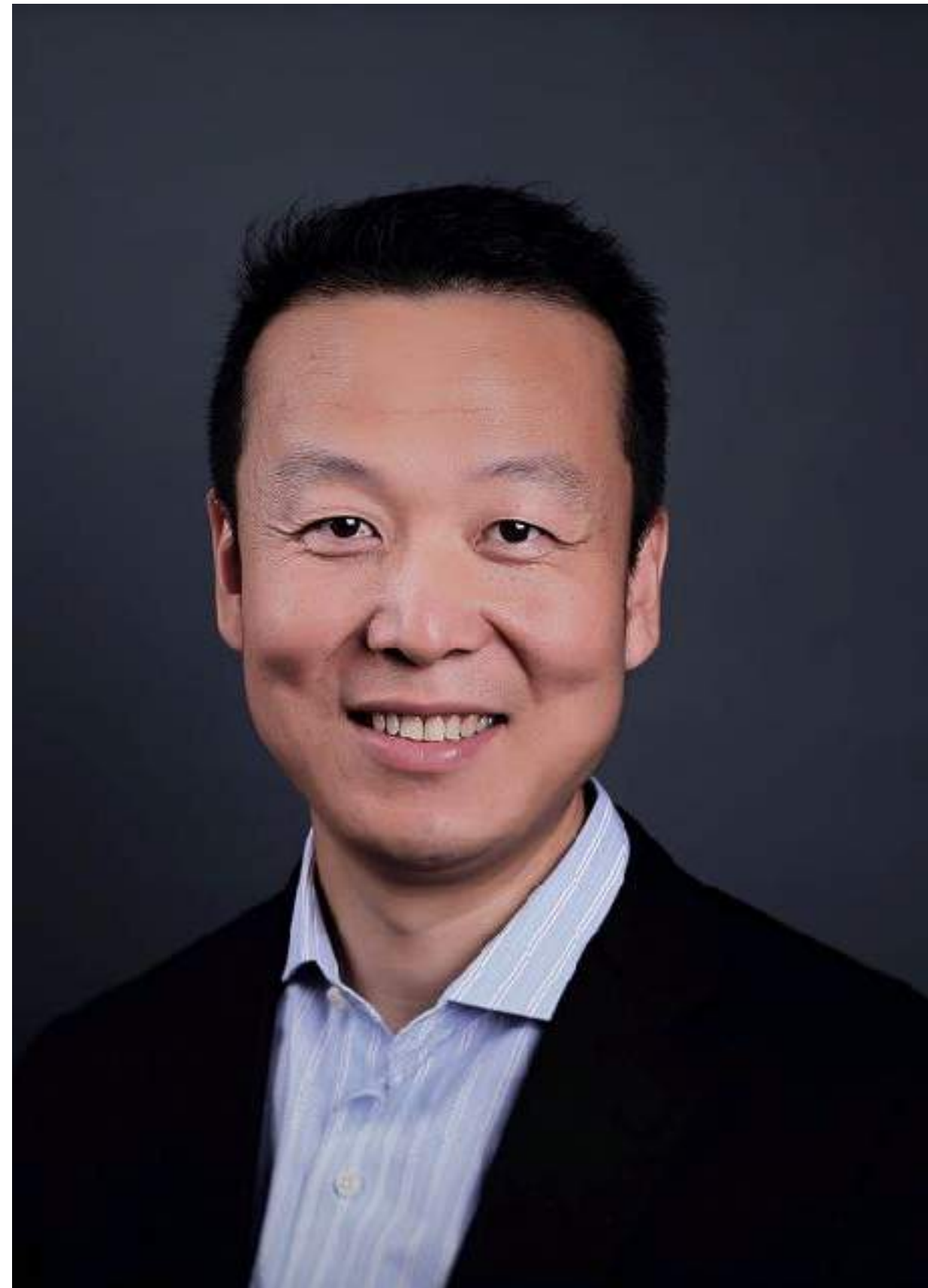




Accelerate 加速度

*DevOps*能力成长模型工作坊

DevOps Coach



刘征

Martin Liu

*Ops*背景

云计算

工具链

许峰

Franklin Xu

*Dev*背景

IT交付

管理





DevOps教练 <http://devopscoach.org>

加速度

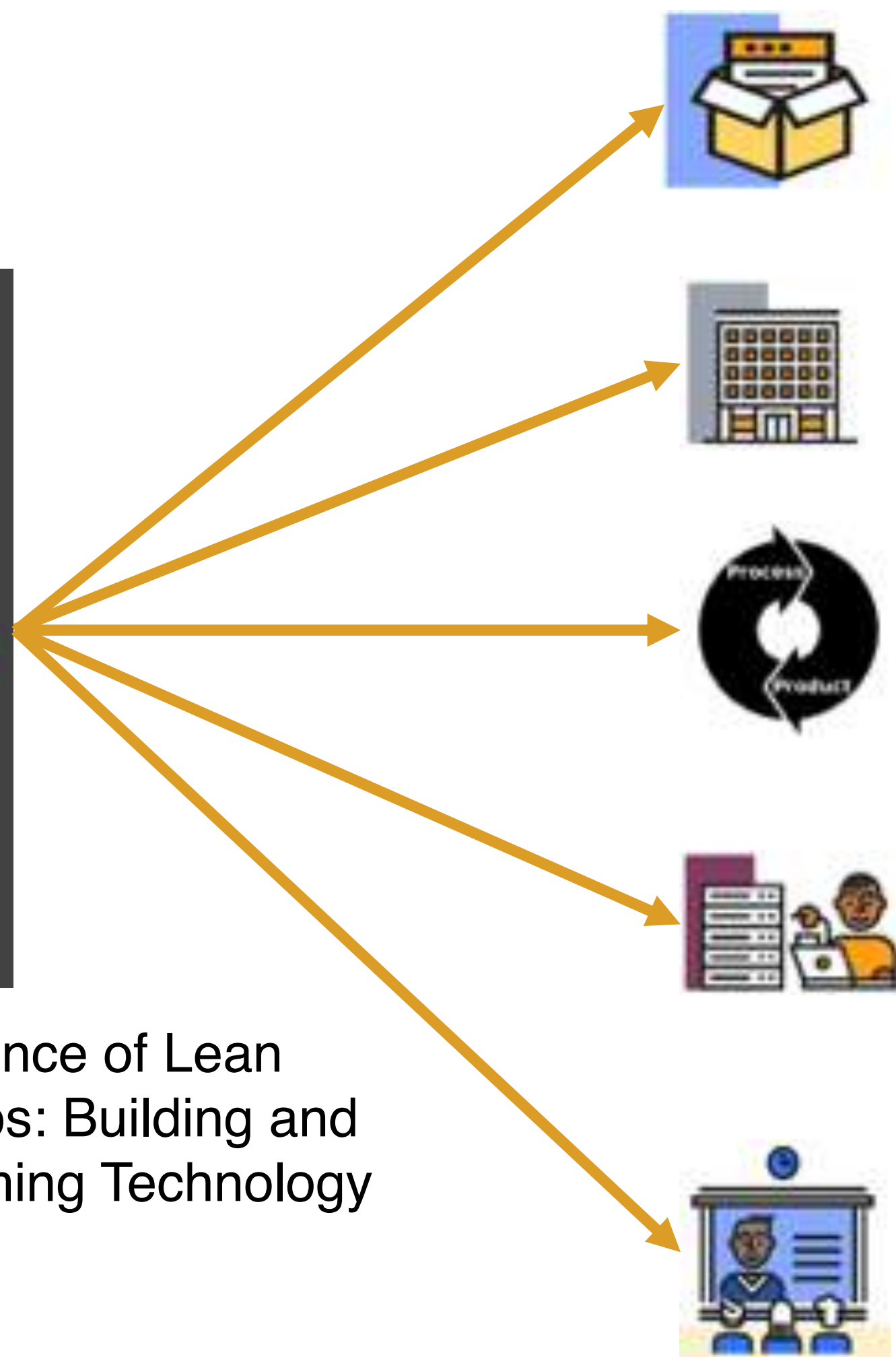
业务一如既往等于日落西山



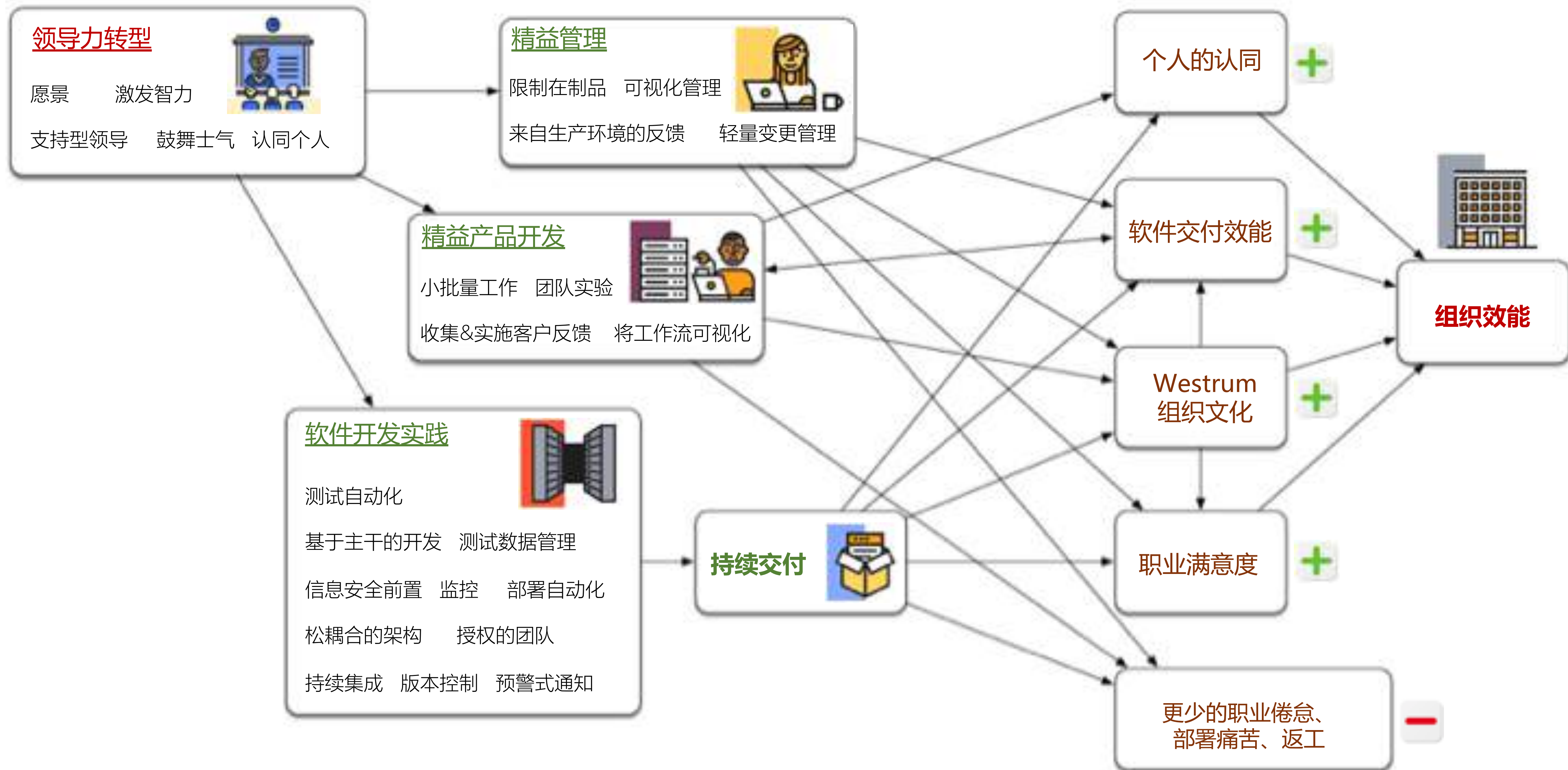


Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations

Link: <http://a.co/eo5FyQj>



| 分类 | DevOps能力描述 | |
|---------|------------|----------------------|
| 持续交付 | 1 | 对生产环境的所有工件进行版本控制 |
| | 2 | 运用自动化部署流程 |
| | 3 | 实施持续集成 |
| | 4 | 运用基于主干的开发方法 |
| | 5 | 实施测试自动化 |
| | 6 | 进行测试数据管理 |
| | 7 | 前置信息安全管理 |
| | 8 | 实施持续交付 |
| 系统架构 | 9 | 应用松耦合的架构设计 |
| | 10 | 授权团队进行架构重构的决策 |
| 产品和流程 | 11 | 收集和实施客户反馈 |
| | 12 | 运用价值流模式可视化工作流 |
| | 13 | 运用小批量的工作模式 |
| | 14 | 培养和赋能团队进行试验 |
| 精益管理和监控 | 15 | 应用轻量变更审批流程 |
| | 16 | 业务决策得到从应用到基础架构的全堆栈支持 |
| | 17 | 前瞻性地监控系统的状况 |
| | 18 | 应用WIP来进行价值流的工作管理 |
| | 19 | 通过可视化来监控团队工作质量和进行沟通 |
| 企业文化 | 20 | 发展并支持生机型企业文化 |
| | 21 | 鼓励和支持学习 |
| | 22 | 支持和辅助团队间的协作 |
| | 23 | 提供工作所需要的资源和工具 |
| | 24 | 支持和落实领导力转型 |



高效能者是怎样完胜高风险和不确定性的？

- 通过快速的交付能取悦客户的服务和产品，组织能快速的交付价值
 - ★ 小型团队（Two pizza team）
 - ★ 短迭代周期
 - ★ 度量来自客户的反馈

不得不加速度

- ▶ 交付取悦客户的产品和服务
- ▶ 通过市场交互理解客户的需求
- ▶ 系统顺应合规和监管的变化
- ▶ 响应潜在风险：
 - ◆ 安全威胁
 - ◆ 经济形势变化



软件技术是核心竞争力

- James Bessen：战略性的使用技术，大于并购公司和业务创新（2017）
- 技术和利润的关联性（2008）
- 软件技术转型和加速度适用于所有组织
- 在DevOps运动中，涌现出来
 - ★ 各种实践
 - ★ 各种能力

The background of the slide is a blurred photograph of two people, a man and a woman, in what appears to be a meeting or office setting. The man is on the left, seen in profile, and the woman is on the right, facing forward. They are both looking towards the center. The image is out of focus, emphasizing the text overlay.

10 Deploys per day Dev & ops cooperation at Flickr

John Allspaw & Paul Hammond
Velocity 2009

Velocity



DevOps 是
技术实践
学习的流程
和文化

这些是效能产出的驱动力

The background of the slide features a dark blue field with out-of-focus binary digits (0s and 1s) in a lighter blue color. Overlaid on this is a semi-transparent grey rectangle containing the text.

DevOps

适用于各种类型的组织

高效能组织能够达到两倍的目标或者更高

$2x$

三年的市场总值
增长高于 50%

商业目标

- ▶ 生产率
- ▶ 利润率
- ▶ 市场份额
- ▶ 用户数

非商业目标

- ▶ 产品或服务的质量
- ▶ 运营效率
- ▶ 客户满意度
- ▶ 产品或服务的数量
- ▶ 达成组织的使命和目标

高效能是怎么炼成的？

成熟度是已经过时的模型





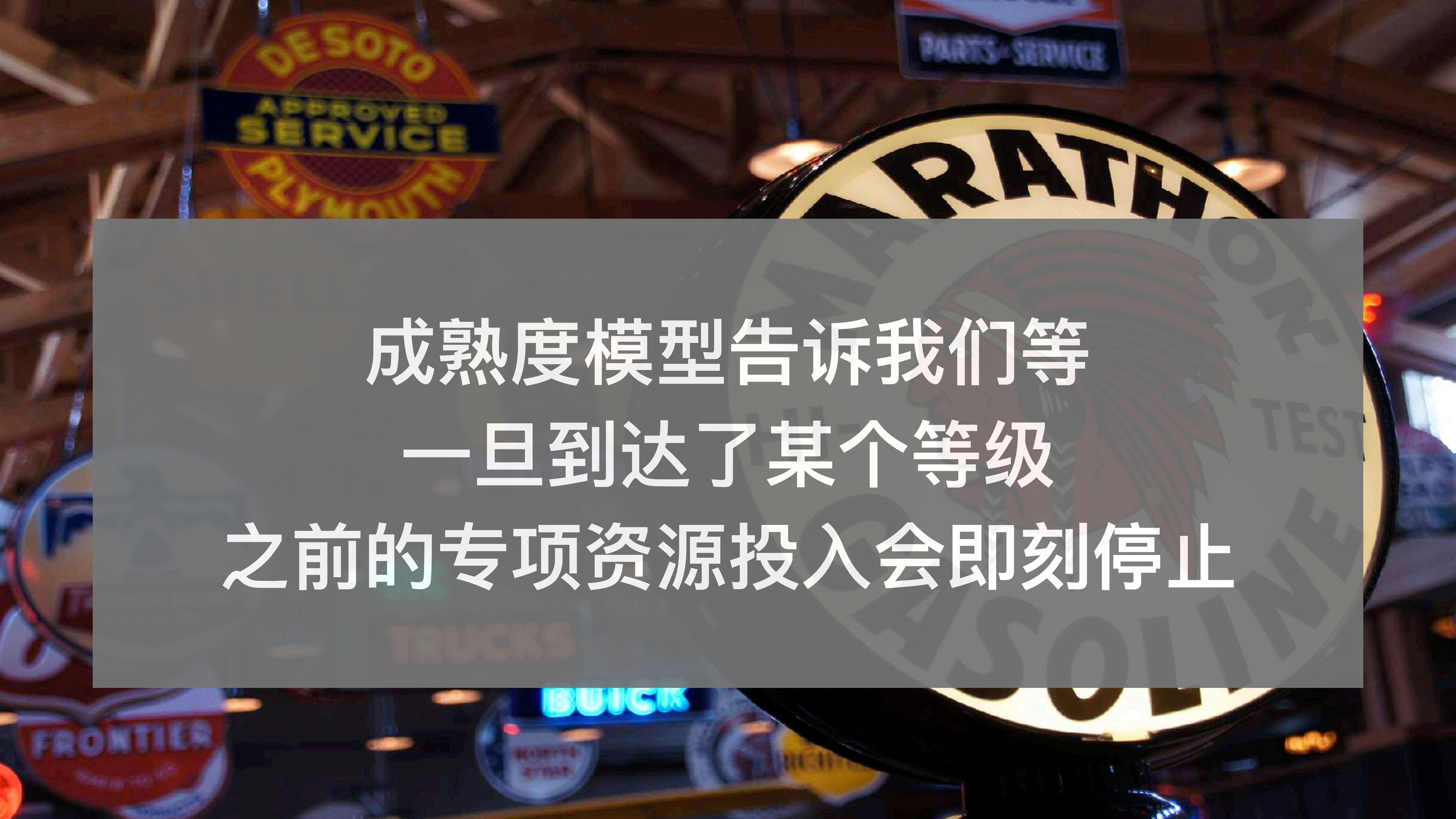
书呆子专属的成熟度模型

- ▶ 2005年，角色级别上限升至70，重新设置角色职业，增加新专业技能、副本、战场和种族，可以变更角色
- ▶ 2008年，角色级别上限升至80，增加死亡骑士职业，开放更多副本和新的装备
- ▶ 2010年，角色级别上限升至85，新增两个种族，游戏的内容在改头换面的艾泽拉斯大陆进行，增加新专业技能


书呆子专属的成熟度模型

- ▶ 2011年，角色级别上限升至90
- ▶ 2013年，角色级别上限升至100，新增新的大陆德拉诺
- ▶ 2017年，角色级别上限升至120，部落和联盟接在海上找到了新的盟友

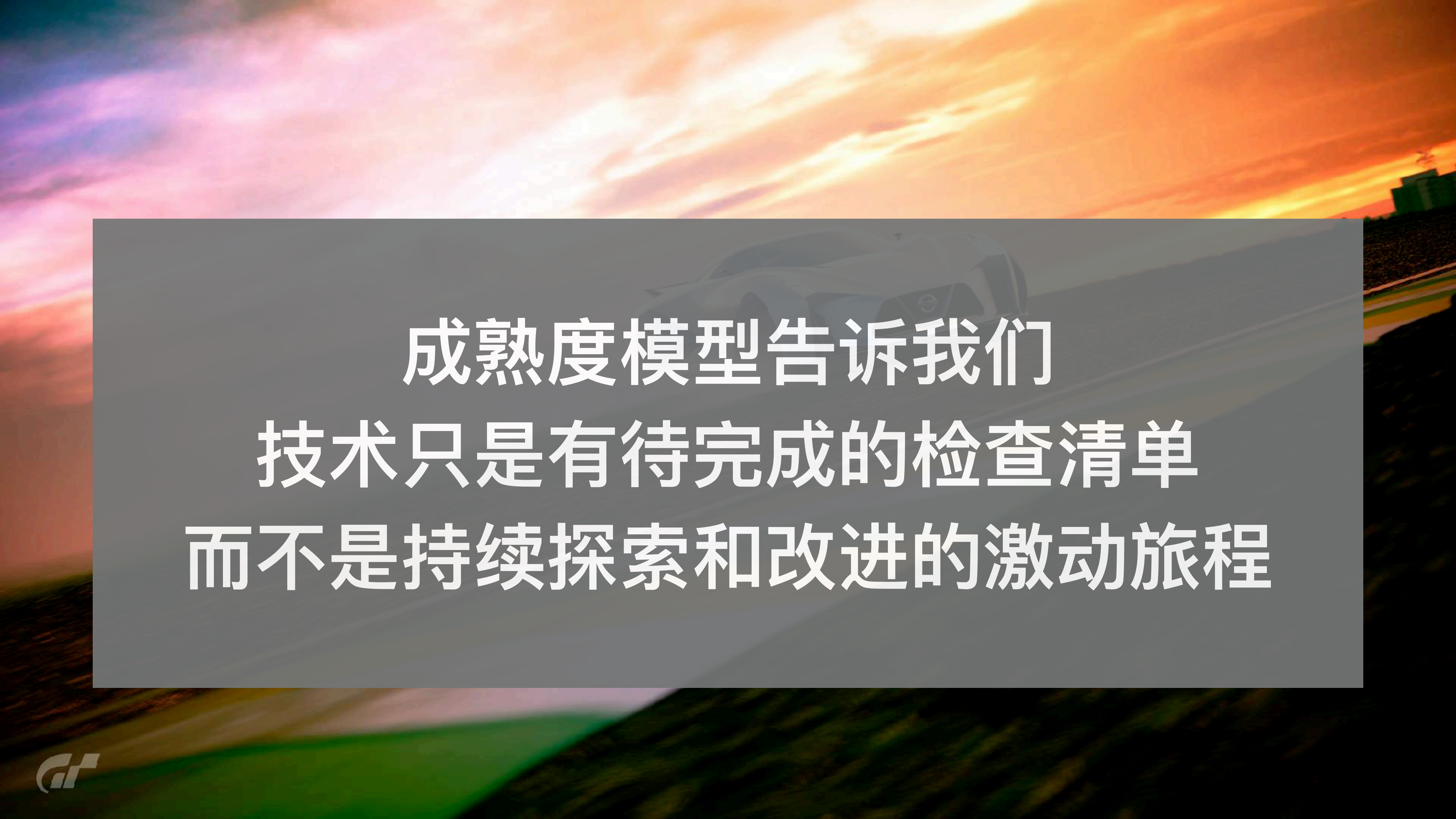
成熟度模型告诉我们
已经到达某个目的地
可是整个世界从你身旁正呼啸而过



成熟度模型告诉我们等
一旦到达了某个等级
之前的专项资源投入会即刻停止



成熟度模型告诉我们
所有的人都是遵循着
一模一样的成功之旅

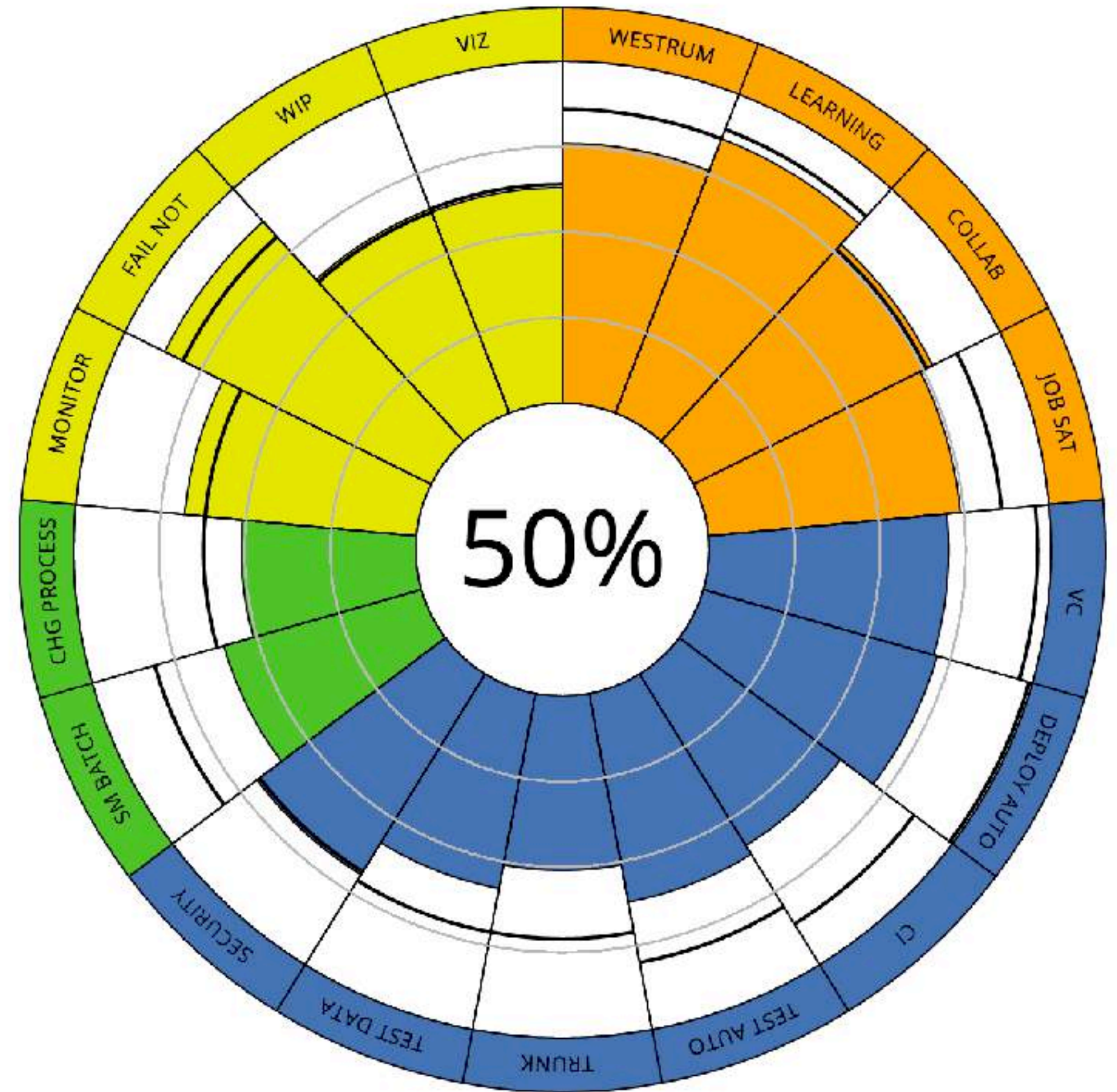
The background of the slide is a photograph of a Formula 1 car on a racetrack during a sunset. The sky is filled with vibrant orange, yellow, and pink clouds. The car is in the center of the frame, slightly blurred, suggesting motion. The track surface is visible in the foreground and background.

成熟度模型告诉我们
技术只是有待完成的检查清单
而不是持续探索和改进的激动旅程



建议参考和应用DevOps能力成长模型

- 适用于本来能力和起点就参差不齐的不同团队
- 符合每个团队的优劣势和业务上下文的不同
- 每个团队聚焦在不同的、各自的约束点/弱点上
- 根据行业基线设置下一步的目标





DevOps教练 <http://devopscoach.org>

软件交付效能

重要的度量方向



度量软件交付效能

- ▶ 同时聚焦产出和全局度量：
 - ▶ 部署频率（在业务需求被满足的前提下）
 - ▶ 变更的前置时间
 - ▶ 故障恢复时间（MTTR）
 - ▶ 变更失败率

吞吐量
稳定性

我们看到 更高的吞吐量 伴随着 稳定性

他们如影随形。并不需要做折中，退而求其次的建议
也是多余的。

高效能DevOps团队更敏捷 Agile

46x

代码部署更频繁

巨大的差距在于：一天多次部署
和每周一次，甚至更少

440x

代码提交到部署的前置时间更短

巨大的差距在于：小于一天
和一周甚至更长

高效能DevOps团队更可靠 Reliable

96x

更快速的恢复时间

巨大的差距在于：小于一天
和几天

1/5x

更低的变更失败率

巨大的差距在于：0~15%失败率
和 31~45%



2017 DevOps状态调查报告： 2017 IT Performance by cluster

| 问卷调查 | 高等IT效能 | 中等IT效能 | 低等IT效能 |
|---|--------------|-------------|-------------|
| 部署频率 针对主要应用和服务，你的组织多久部署一次代码？ | 按需 (一天多次) | 1周 < X < 1月 | 1周 < X < 1月 |
| 变更前置时间 针对主要应用和服务，变更的前置时间是多长（从代码提交到代码在生产环境中运行成功的时间）？ | 小于1小时 | 1周 < X < 1月 | 1周 < X < 1月 |
| 故障恢复时间（MTTR） 针对主要应用和服务，如果发生服务故障，一般多久能恢复服务（比如：计划外宕机，服务损害）？ | 小于1小时 | 小于1天 | 1天 < X < 1周 |
| 变更失败率 针对主要应用和服务，变更结果是回滚或随后修复的比例是多少（比如：导致服务损害，服务中断，需要紧急修复，回滚，向前修复，补丁）？ | 0~15% | 0~15% | 31~45% |

常见错误

- 产出 > 输出
 - Outcomes > outputs
- 全局/团队 > 局部/个体
 - Global / team > locale / individual



常见错误：代码行数 Line of code

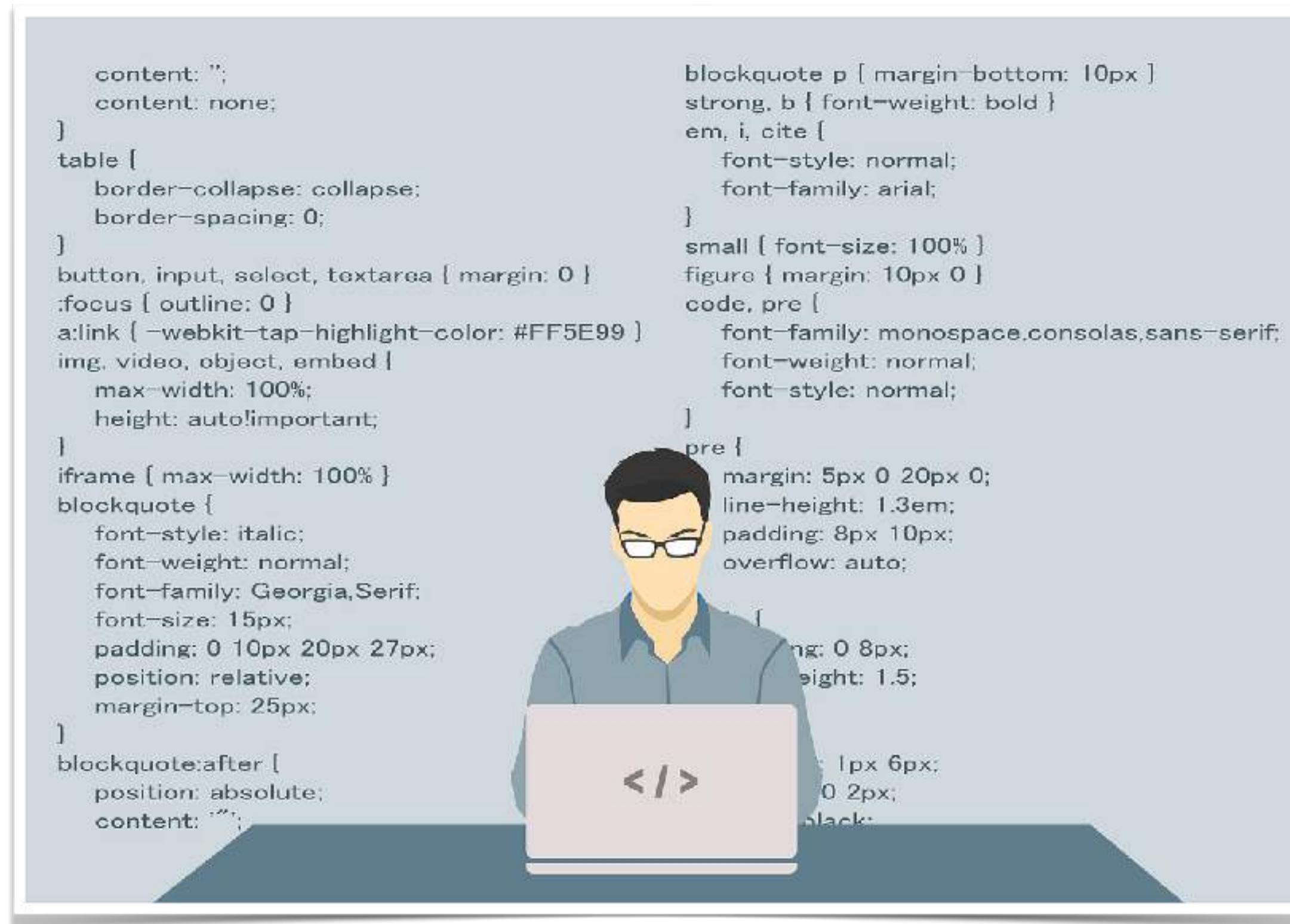
➤ 越多越好？

- ★ 更臃肿的软件
- ★ 更高的维护成本
- ★ 更高的变更成本

➤ 越少越好？

- ★ 玄妙莫测且无法理解的代码

➤ 理想的：用最有效的代码解决业务问题



常见错误： 速率 Velocity

- Agile: 问题被拆分成故事，评估完成这些工作的点数
- 在冲刺结束的时候，由客户签收的点数被记录下来 = 速率
- Velocity 速率是 产能计划 工具。而不是 生产效率工具
- 为什么速率不能作为生产效率使用？
 - ★ 速率只是一个相对的度量，而非绝对的。因此，很不适合用于团队间的比较
 - ★ 点数评估被夸大，被博弈
 - ★ 聚焦在团队的完成度，通常牺牲了团队间的协作（这是一个全局目标）



常见错误：利用率 Utilization

- 利用率仅仅在某个点上/程度上是好的
- 利用率越高越好么？
 - ★ 在高利用率下，我们失去了用于非计划工作的可支配的时段
 - ★ 排队理论：在利用率达到100%的时候，前置时间接近于无穷大
 - ★ 当你们的利用率越来越高的时候，所有团队将花费越来越长的时间完成任何的工作





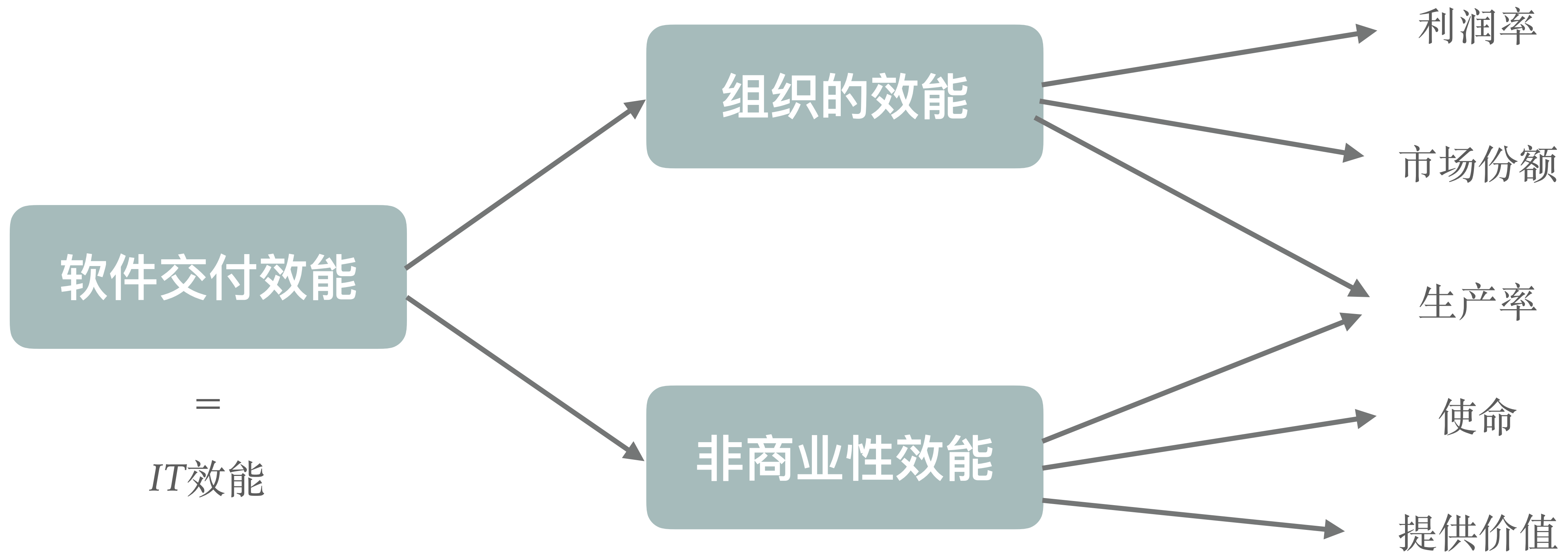
DevOps教练 <http://devopscoach.org>

技术实践很关键

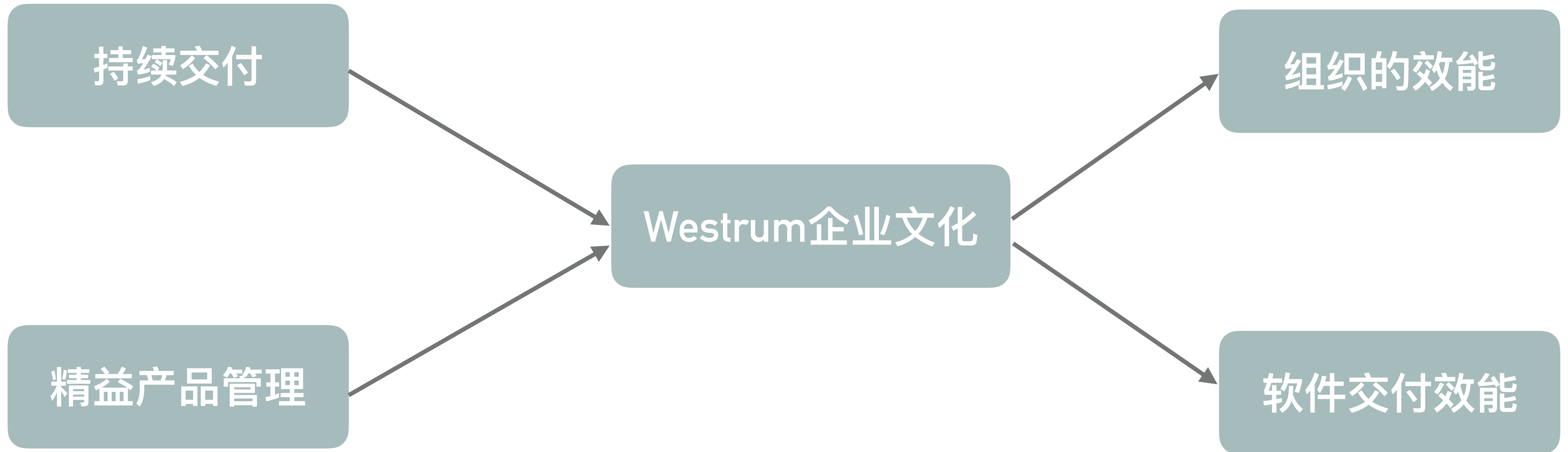
*DevOps*是各种技术实践的大融合



度量效能： 软件交付效能的影响地图



度量和改变文化：企业文化的影响地图



持续交付的驱动力 — 持续交付的影响地图

1. 版本控制
2. 自动化部署
3. 持续集成
4. 主干开发
5. 自动化测试
6. 测试数据管理
7. 模式化安全管理
8. 松耦合架构
9. 赋能团队
10. 监控
11. 前瞻性预警通知

持续交付

Westrum企业文化

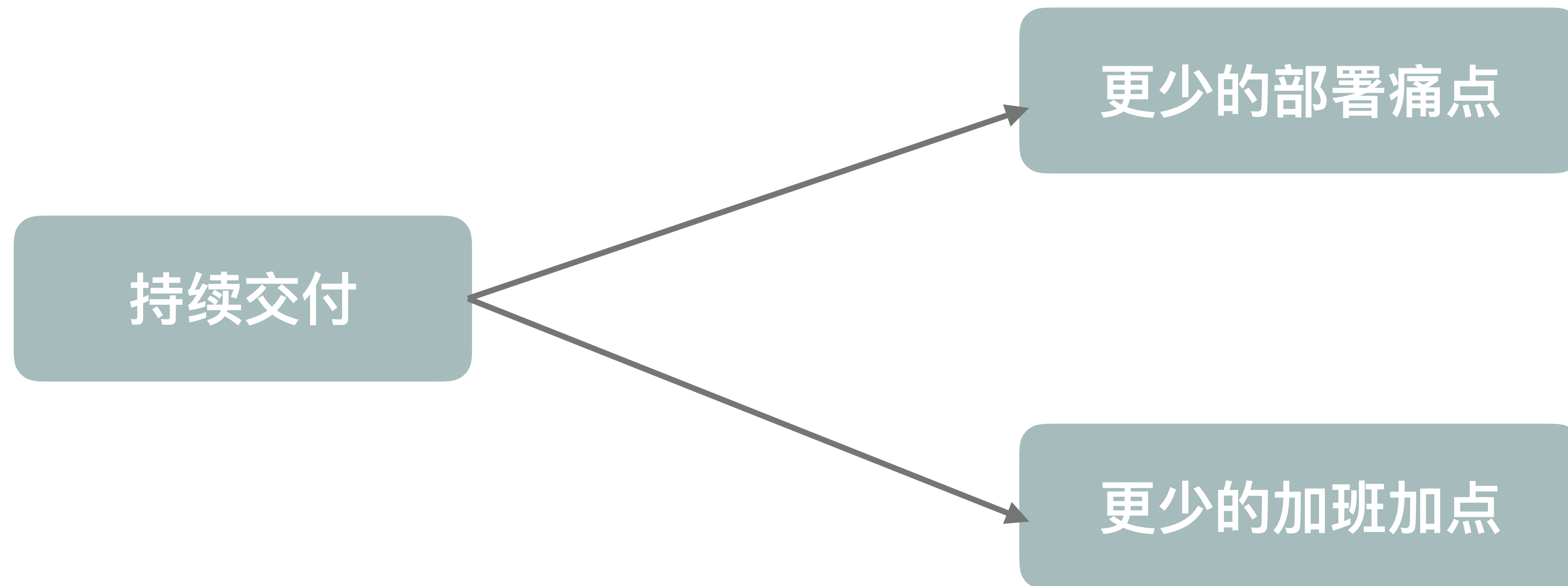
软件交付效能

组织的效能

更少的返工

个人的认同

持续交付意味着使工作更可持续发展





DevOps教练 <http://devopscoach.org>

架构很关键

而不是技术



技术堆栈并不关键

- 低效能者们的相似之处：
 - 使用外包的方式开发软件
 - 使用Mainframe系统
- 但是：
 - 使用Mainfram系统并不和效能相关联
 - 工作在绿地应用或棕地应用也不和效能相关联

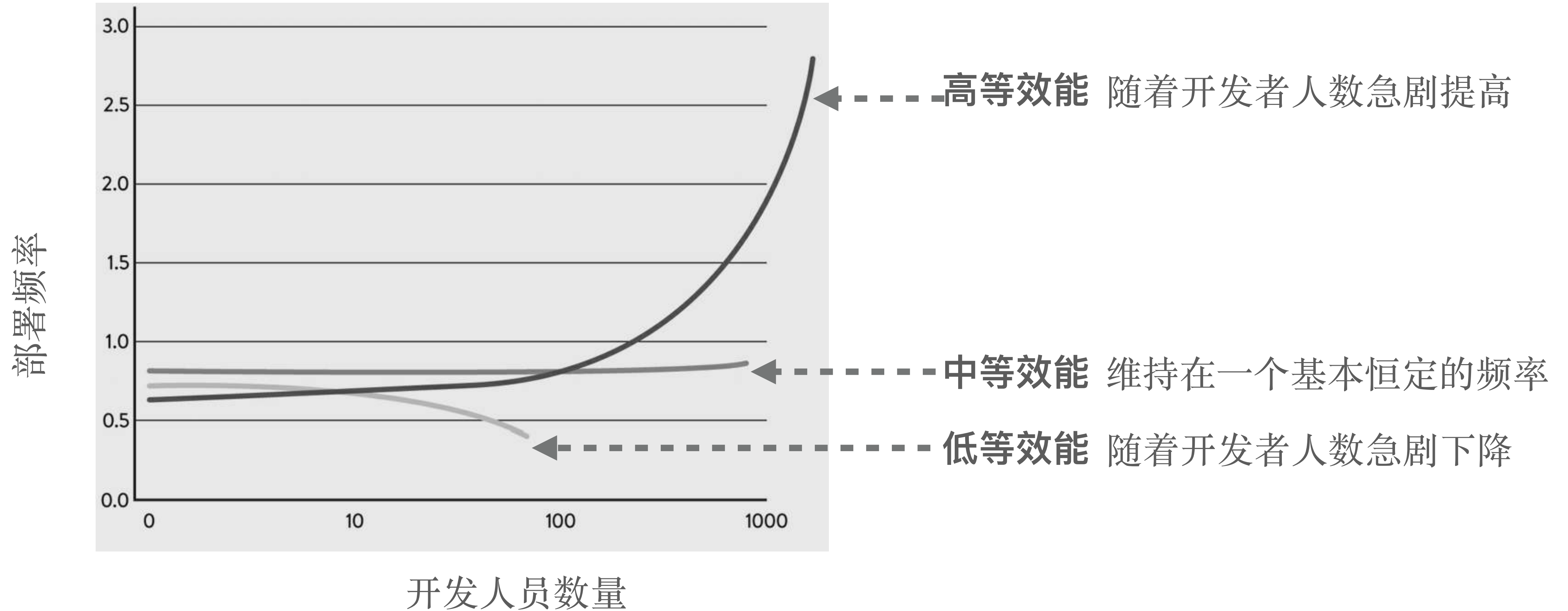


关键的是架构的产出

► 我们的团队是否能：

- ★ 变更系统的设计
- ★ 测试系统
- ★ 部署系统
- ★ 而不依赖与外部人进行沟通和协作

每个开发者，每天的部署次数



“

任何组织在设计一套系统时，所交付的设计方案在结构上都与该组织的沟通结构保持一致。

-梅尔.康威



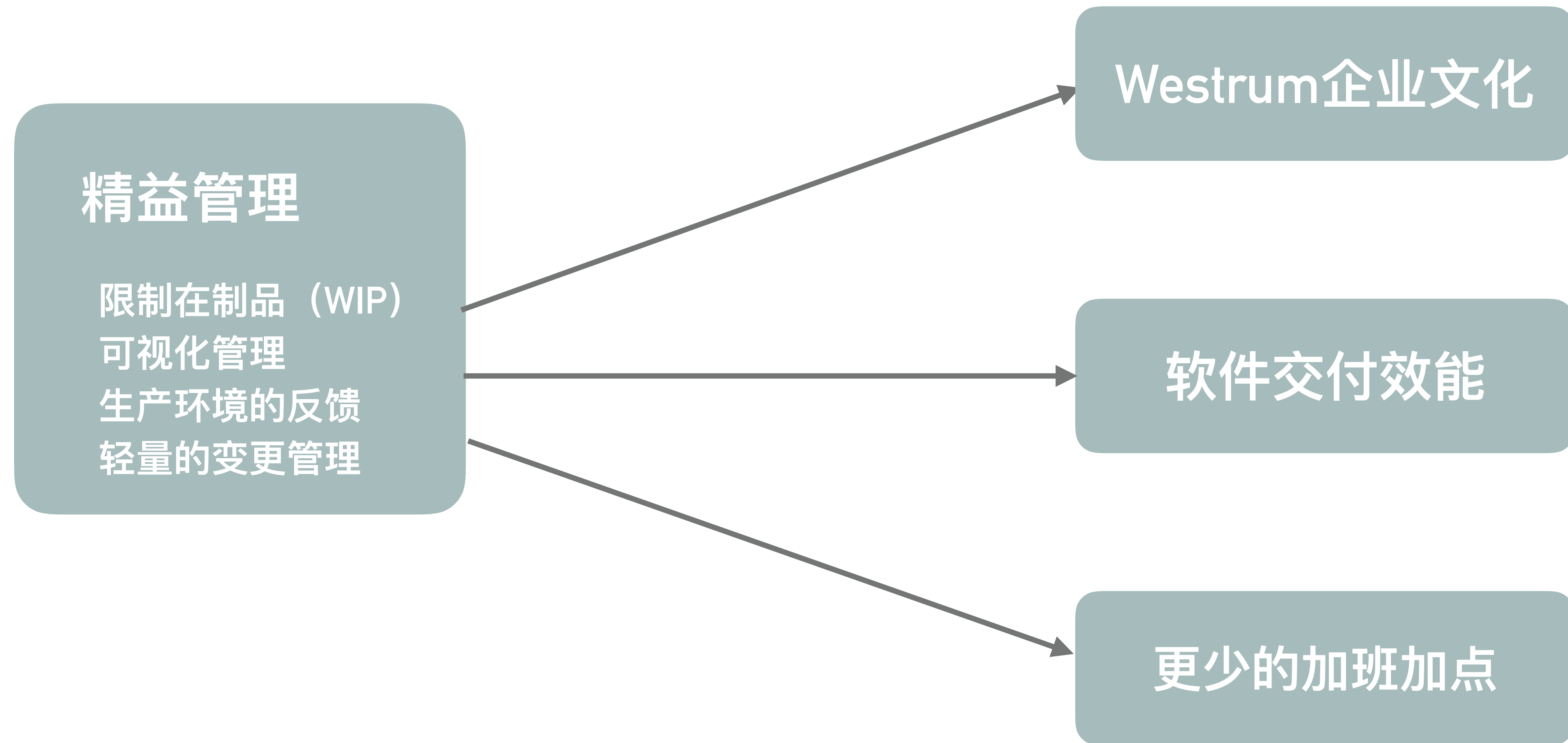
DevOps教练 <http://devopscoach.org>

软件管理实践和 产品开发很关键

两手抓，两手都要硬

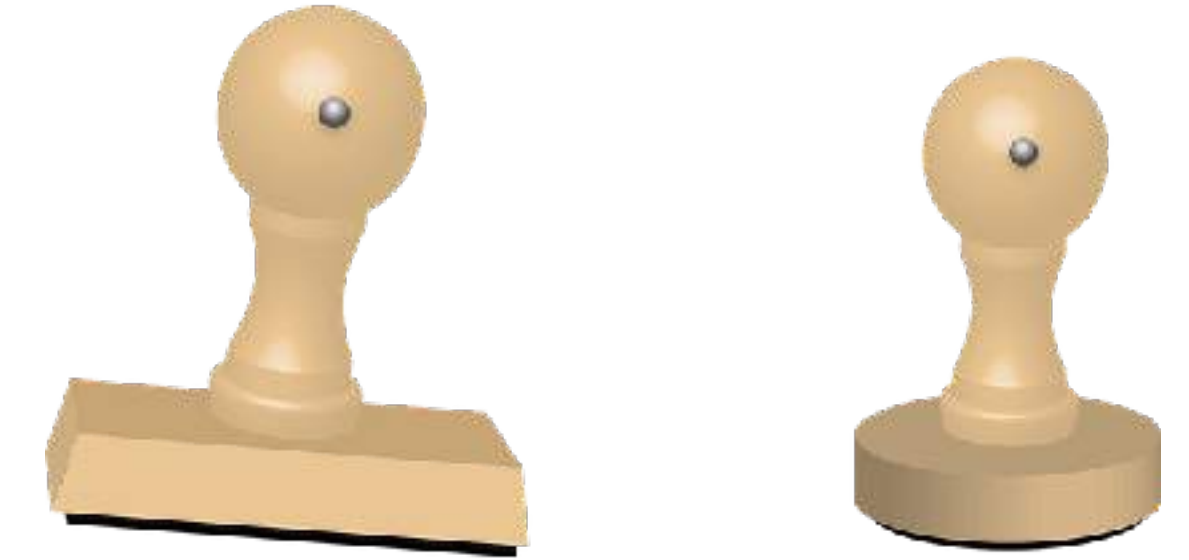


精益管理实践的影响地图



实施轻量的变更管理流程

► 评估变更审批流程对软件交付效能的影响，结果：

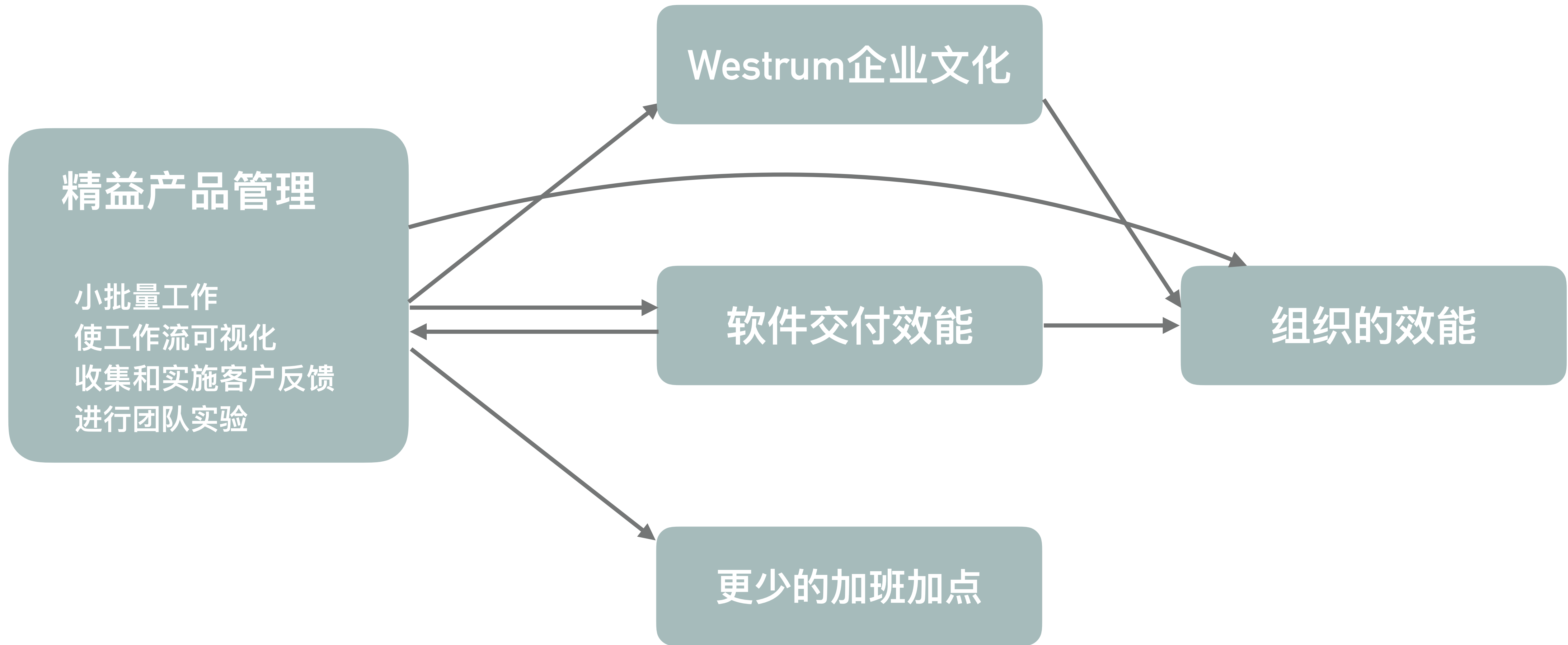


1. 只针对高风险变更进行审批与软件交付效能是无关的
2. 没有变更审批流程，或者仅仅依靠同行评审的团队展现出更高的软件交付效能
3. 需要外部人员（CAB）进行审批的变更流程导致更低的软件交付效能

► 外部CAB审批对：前置时间、部署频率、恢复时间有负面影响；相反CAB对提高生产系统的稳定性没有相关性

► 建议：应用基于同行评审（结对编程、团队内代码评审、结合流水线）的轻量变更审批流程

精益产品管理的影响地图





DevOps教练 <http://devopscoach.org>

领导力很关键

组织转型领导力



领导力转型的5个维度

愿景

- 明晰组织方向。
- 明确团队方向。
- 预见5年后团队方向。

智力激发

- 挑战团队现状
- 挑战团队不断提出新问题。
- 挑战团队对工作的基本设想。

个体认同

- 赞扬高出平均水平的工作。
- 认可工作质量的提高。
- 亲自赞美个人的出色表现。

鼓舞型沟通

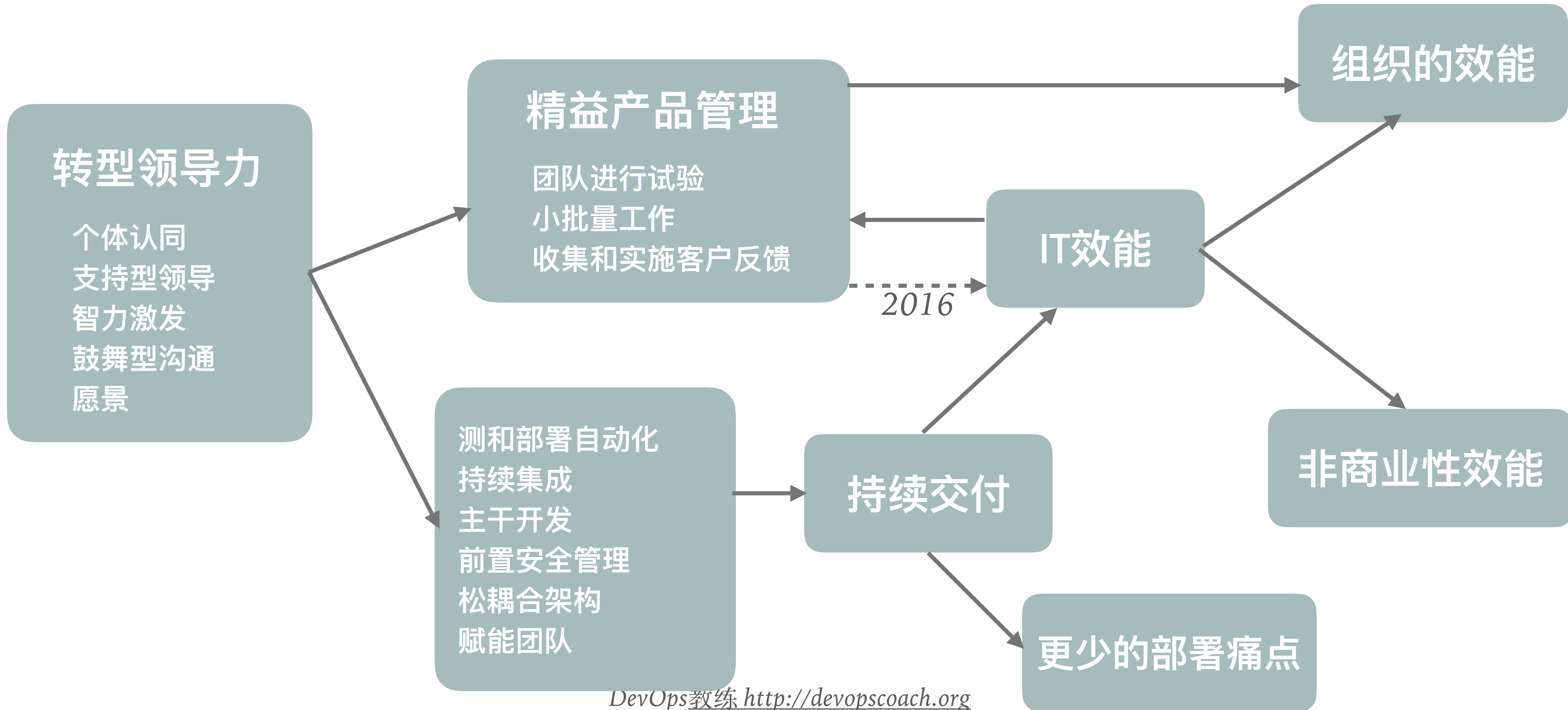
- 激发团队成员的自豪感。
- 宣传团队内部的正能量。
- 激发热情和积极性; 鼓励人们看到这种变化带来的机会。



支持型领导

- 在行动之前考虑他人的感受。
- 思考他人的个体需求。
- 关心个人的兴趣。

转型领导力和效能之间的关系



我们一起能将工作做的更好



总是有更好的方法

➤ 聪明的投资在技术和实践上，使我们的工作变的更好：

- 工作在：减少部署痛点方面
- 工作在：降低精疲力尽/加班加点上
- 工作在：提高NPS员工净推荐值上



在高效能组织中工作的雇员更愿意
向周围的朋友推荐自己的组织

2.2x



DevOps教练 <http://devopscoach.org>

怎样开始?

应用能力成长模型的套路



建议的套路

- 开始于度量几个少量的指标
 - 聚焦在产出型的、全局的指标上
 - 考量有哪些在你控制之内的事情可以推动这些指标的提升的—不仅仅在技术方面，而且还包括非技术方面
- 分享你的成功案例！利用本地社群



DevOps教练 <http://devopscoach.org>

Question?

问答时间

谢谢！



刘征



*DevOps*教练

许峰



*DevOps*咖啡馆