敏捷软件开发

- 敏捷概述和核心理念

敏捷诞生的历史背景

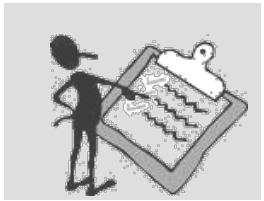
20世纪60年代 **软件作坊**

软件规模小，以作坊式开发为主；

70年代 **软件危机**

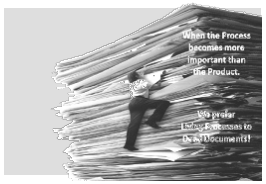
硬件飞速发展，软件规模和复杂度激增，引发软件危机；

80年代 **软件过程控制**



引入成熟生产制造管理方法，以“过程为中心”分阶段来控制软件开发（瀑布模型），一定程度上缓解了软件危机；

90年代 **重型过程**



软件失败的经验促使过程被不断增加约束和限制，软件开发过程日益“重型化”，开发效率降低、响应速度变慢；

2001~今 **敏捷正在流行**

随着信息时代到来，需求变化更快，交付周期成为企业核心竞争力，轻量级的，更能适应变化的敏捷软件开发方法被普遍认可并迅速流行。

- 软件开发顺应时代变化，从重型过程转向轻量型敏捷

敏捷宣言揭示更好的软件开发方法

敏捷宣言

我们正在通过亲身实践以及帮助他人实践，揭示更好的软件开发方法。通过这项工作，我们认为：

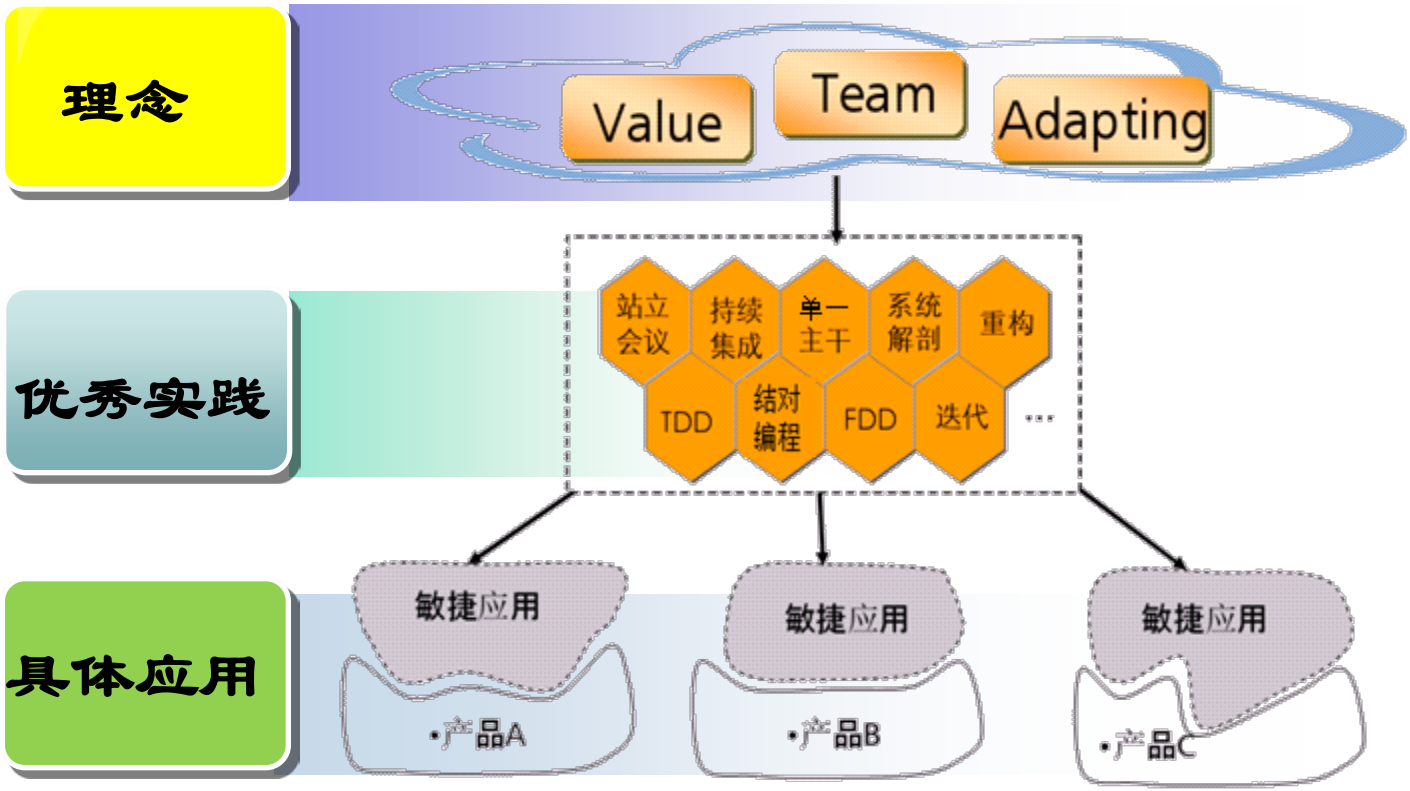
个体和交互	胜过	过程和工具
可以工作的软件	胜过	面面俱到的文档
客户合作	胜过	合同谈判
响应变化	胜过	遵循计划

虽然右项也具有价值，
但我们认为左项具有更大的价值。

Agile Alliance®

- 敏捷宣言（2001年）是敏捷起源的基础，由上述4个简单的价值观组成，敏捷宣言的签署推动了敏捷运动
- 核心思想：客户价值，以人为本，适应变化

敏捷=理念+优秀实践+具体应用

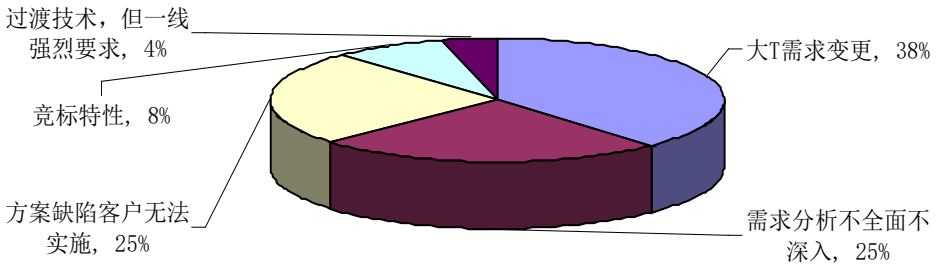


敏捷包括3个层次
- 理念（敏捷核心思想）
- 优秀实践（敏捷的经验积累）
- 具体应用（能够结合自身灵活应用才是真正敏捷）

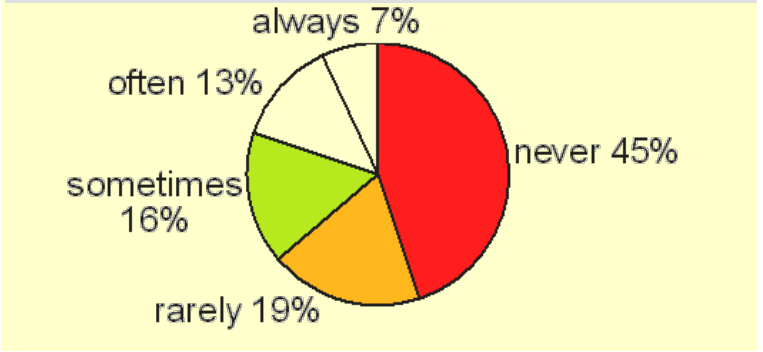
理念：聚焦客户*价值(Value)*，消除浪费

企业（华为）：研发版本废弃特性

- 07.1-08.6年某产品线所有产品中重要特性无应用的比例达22%（需求变更和分析不足占63%）



软件业：45%的软件特性客户没有使用



Source：Standish Group 来自5万个软件开发项目的调查

行业（电信业）：“电信级”带来的浪费

- 电路交换机的2000多个功能只用了1%
- 路由器网管的告警只有0.01%是有意义的
- 我们提供的上万种业务套餐80%以上使用者不到10个人，浪费了无数昂贵的资源

Source：中国电信总工韦乐平在《华为公司工程与技术大会》上的讲话

“价值”在“敏捷宣言”中的体现

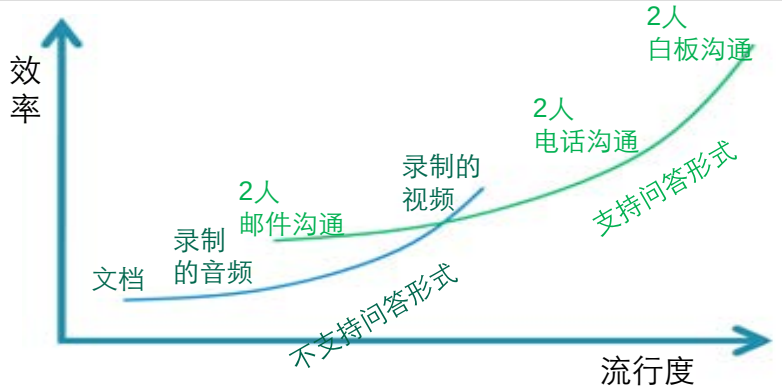
个体和交互	胜过	过程和工具
可以工作的软件	胜过	面面俱到的文档
客户合作	胜过	合同谈判
响应变化	胜过	遵循计划

- 产品商业成功为目标，聚焦客户价值、围绕价值流消除浪费

理念：激发团队(Team)潜能，加强协作

研究表明面对面的沟通最有效

业界调查：一个50人开发团队，每人平均30%时间用于编码，70%的时间用于与其他成员交流。



试点开发测试拉通，效率质量改善明显

	需求变更降低比例	补充场景数	TR4前发现缺陷比例	版本周期缩短（周数）
无线	49.36%	88	55.90%	2.82
核心网	45%	190	45.18%	3.5
网络	31%	330	42.5%	2.6
业软	30%	300	48.15%	2.1
公司平均	38.84%	908	47.93%	2.76

Source: 08年测试行业超过30个项目试点

人是软件开发的决定因素

研究表明1981年来自不同公司的优秀程序员生产率之比是7:1，而2007年最新的研究数据，则是40:1。

Source： 《经济学家2003》& DeMarco 研究报告

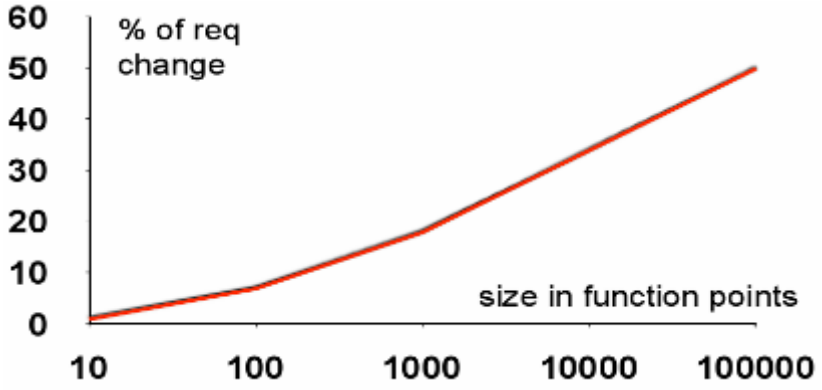
“团队”在“敏捷宣言”中的体现

个体和交互	胜过	过程和工具
可以工作的软件	胜过	面面俱到的文档
客户合作	胜过	合同谈判
响应变化	胜过	遵循计划

- 团队是价值的真正创造者，应加强团队协作、激发团队潜能
- 软件开发是一种团队活动，首先应做到提升沟通效率降低交流成本

理念：不断调整以适应(Adapting)变化

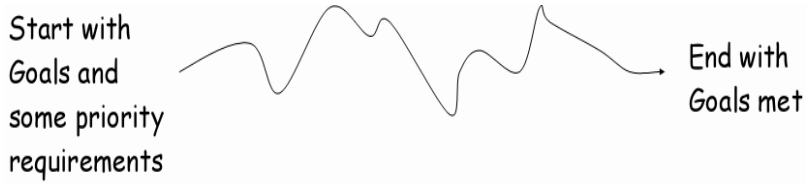
随软件规模增长，需求变化呈非线性增长



软件开发规律再审视

- 《人月神话》：软件开发是人类最复杂工作之一，软件具有四个属性：复杂性、一致性、可变性和不可见性。
- 软件开发是不可重复、探索性的、演进的，适应性过程。

软件开发是复杂不可预测的经验控制过程



麦当劳是简单可预测生产过程

“适应变化”在“敏捷宣言”中的体现

个体和交互	胜过	过程和工具
可以工作的软件	胜过	面面俱到的文档
客户合作	胜过	合同谈判
响应变化	**胜过**	**遵循计划**

- 不断的根据经验调整，最终交付达到业务目标的产品

深入理解敏捷理念

- **深入理解“聚焦客户价值”**
  - 标识和消除软件开发中的浪费
  - 交付刚刚好的系统
  - 随时构建质量，不容忍缺陷
  - 及时消除技术债务，持续保持快速响应
- **深入理解“激发团队”**
  - 认清团队的基本事实
  - 敏捷方式下管理者的转变
  - 敏捷方式下团队成员的转变
- **深入理解“适应变化”**
  - 认请“客户是逐步发现真正需求”
  - 小批量是快速交付的关键
  - 通过迭代计划不断调整以适应需求变化
  - 应持续保持良好的软件架构
  - 利用多层次反馈不断调整以逼近目标

	浪费类别	浪费举例
1	部分完成的工作	部分完成但没有最终落地的工作（没有转化成代码的设计文档；未及时合入的代码导致引发后续更多同步工作量）。
2	未应用特性	开发完成但没有被客户应用的特性（交换机2000多个功能客户只用了1%）。
3	再次学习	人员频繁流动导致经验不能积累，反复重新学习；在多个环节移交时，接收信息者也需要重新学习；拥有某领域的专家，但在开发过程中需要此领域经验时，他却没参与，而是团队重新摸索。
4	移交	知识信息的传递总是伴随信息丢失，隐形知识尤其困难，分工过细往往导致过多不必要的移交（如详细设计和实现分离，造成大量设计信息丢失）。
5	任务切换	研究表明多任务工作会导致效率下降20%-40%（员工多头工作或杂事繁多）。
6	延迟	因任务或资源相互依赖而导致工作停滞（集成时被关键模块阻塞，等待测试环境就绪）。
7	缺陷	解决缺陷活动本身就是浪费，而且缺陷越遗留到后端浪费越大。

Source：《精益软件开发》

聚焦客户价值，交付刚刚好的系统

- **在项目明显超负荷时，管理者简单地期望靠团队work harder来解决，最终导致**：
  - 质量下降
  - 项目延期
  - 客户不满意
  - 团队疲劳
  - 埋下长期隐患

- **当质量、进度、资源冲突时，能改变的只有项目范围，即选择“交付刚刚好的系统”**
  - 产品交付前，客户往往期望多而全的功能，产品交付后，客户把稳定的质量放在首位，尤其在电信领域，客户对产品质量要求是**Always work**，不是**Sometimes**。
  - 与其为了满足多而全的功能导致交付延迟，质量不稳定，不如按时交付刚刚好的系统，保证其高质量运行。
  - 交付刚刚好的系统，基于对客户需求的深入理解，并花时间了解细节，简化（simplify）需求（降低复杂性）而不是简单地拒绝需求（delete）。
  - 做到“交付刚刚好的系统”，同时需要管理者有足够的勇气和果断决策

聚焦客户价值，随时构建质量，不容忍缺陷

- **缺陷遗留带来高额成本：**
  - 对单独质量保证活动（如后端测试）的依赖，容易形成缺陷可以遗留到下个阶段的心理，导致缺陷发现成本升高（系统测试阶段缺陷定位和解决成本是开发阶段的10倍）
    - 例1：E公司开发阶段和测试阶段发现缺陷的比例为7：3，而我司大量缺陷集中在后端发现，带来高额成本。
    - 例2：我司顾问指出：华为测试和开发“相隔1000英里”。

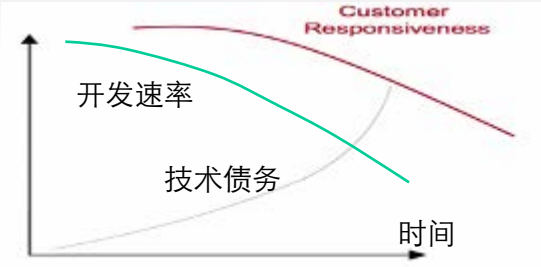
- **从项目一开始就随时构建质量：**
  - 形成零缺陷文化，不要容忍缺陷：发现缺陷应立即停下来解决，以保证在坚实的质量基础上前行。
  - 开发和测试紧密协作：测试人员参与到设计和开发过程中，共同预防缺陷的产生。

例如：持续集成暴露的问题需立即解决

聚焦客户价值，及时消除技术债务，持续保持快速响应

- **常见技术债务:**
  - 日益腐烂的架构、圈复杂度高的代码、低的测试自动化率、不及时清除的静态检查告警等。
- **为什么会有技术债务:**
  - 为满足短期商业目标，不影响其外部表现的情况下，会在技术方面进行一定的让步，这种让步虽对当前版本的质量影响甚微，但会严重影响后续版本响应客户需求的能力，从而形成技术债务。

- **对待技术债务的态度:**
  - 技术债务是有成本的，如不及时偿还，会随时间积累利息变高，导致开发效率大幅下降，从而降低客户响应能力。因此对待技术债务的态度是加以管理并及时偿还（如及时重构）。



激发团队，认清团队的基本事实

- **关于团队激励:**
  - 当团队自管理时效率最高
  - 人们对自己做出的承诺比别人要求的承诺更认真
  - 人们会尽力做到最好
  - 在强大的压力下努力工作，人们会自然而然地降低对质量的要求
- **关于团队绩效:**
  - 当团队成员不被打扰时，工作效率最高
  - 当团队解决自我问题时，提升最快
  - 广泛的、面对面的交流是团队工作最高效的方式
- **关于团队构建:**
  - 团队生产率大于相同数目的个体生产率之和
  - 当不同技能领域的人员组成团队并聚焦于工作时，产品更健壮

信任是高绩效团队的基石

Source: Jeff CSM Training material

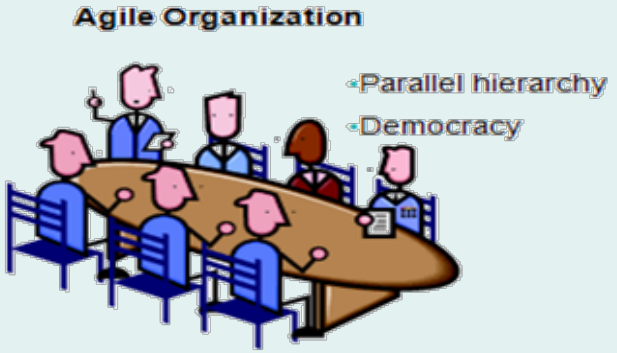
激发团队，敏捷方式下管理者的转变

传统方式

管理者努力“控制” 团队：

- 制定详细的工作计划，并做出详细的工作安排
- 指令性工作方式
- 监控过程
- 基于复杂规则的管理

敏捷方式



管理者努力“激发”团队：

- 通过目标来牵引团队自主工作
- 帮助团队提供资源，排除障碍
- 营造团队自我管理的工作氛围
- 作为教练辅导团队进步
- 基于简单原则的管理，原则简单但必须被遵守

敏捷方式下对管理者最大的挑战是学会放松”控制”(loose control)

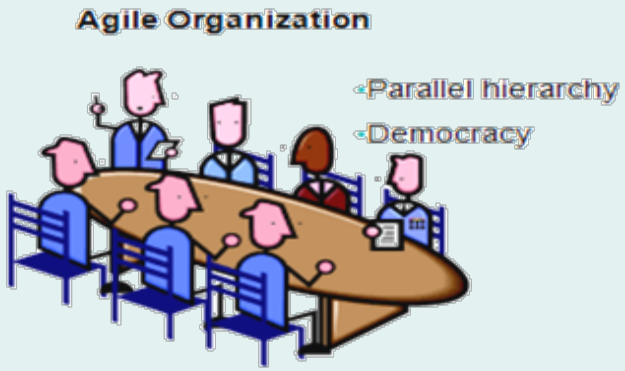
激发团队，敏捷方式下团队成员的转变

传统方式

团队成员是“**听从安排的独立贡献者**”：

- 被动等待主管下指令安排工作
- 独立工作为主，协作少
- 以文档和邮件为主要沟通方式
- 只关注个体任务“做完”，不关注团队目标
- 能力相对单一，学习动力不足

敏捷方式



团队成员是“**全方位的积极参与者**”:

- 共同参与计划制定和任务安排
- 团队协作贯穿工作始终
- 面对面交流是主要沟通方式
- 关注团队目标，共担责任
- 能力要求更广，主动学习适应岗位要求

从被动到主动的心态转变是团队成员适应敏捷开发的关键

适应变化，认清“客户是逐步发现真正需求”

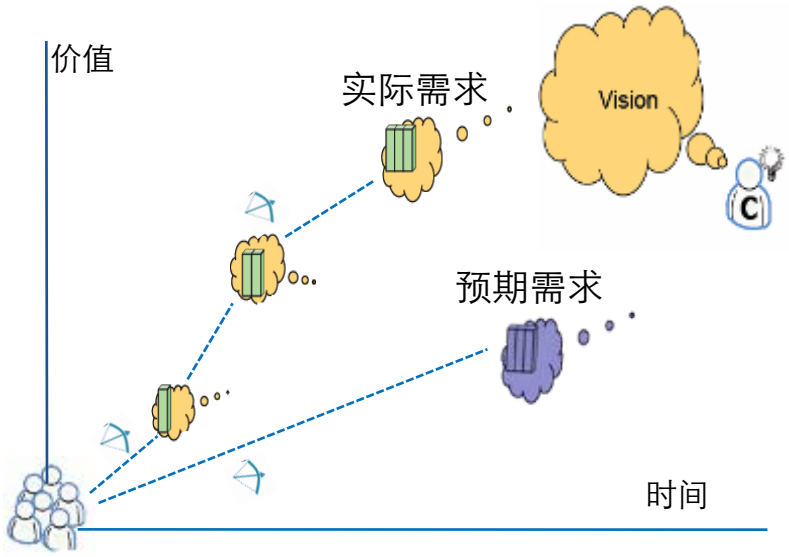
美好愿望

- 客户知道自己要的是什么
- 开发人员知道如何开发来满足客户需求
- 在开发过程中需求不会发生变化

我们认识到

残酷现实

- 客户是逐步发现他真正要的东西
- 开发人员逐步发现如何开发产品满足客户需求
- 在这个过程中随时可能发生变化

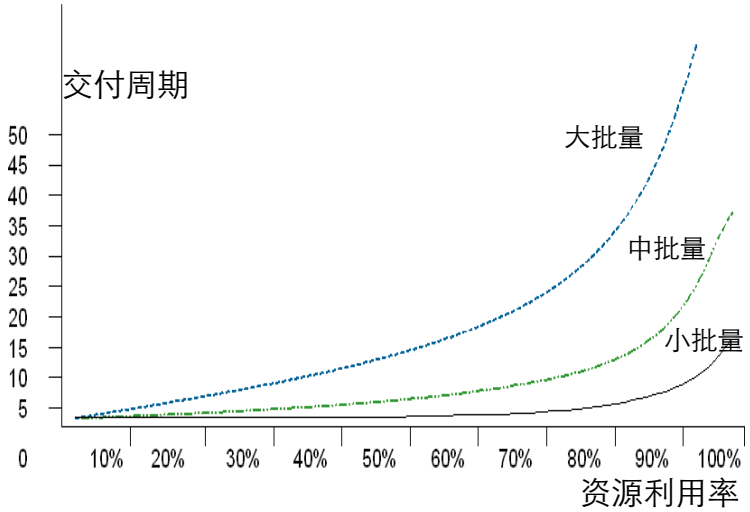


- 期望客户一开始就想清楚他们真正要的东西是不现实的。
- 我们应当通过不断的向客户交付可用的产品，启发客户逐步的发现真正的需求。

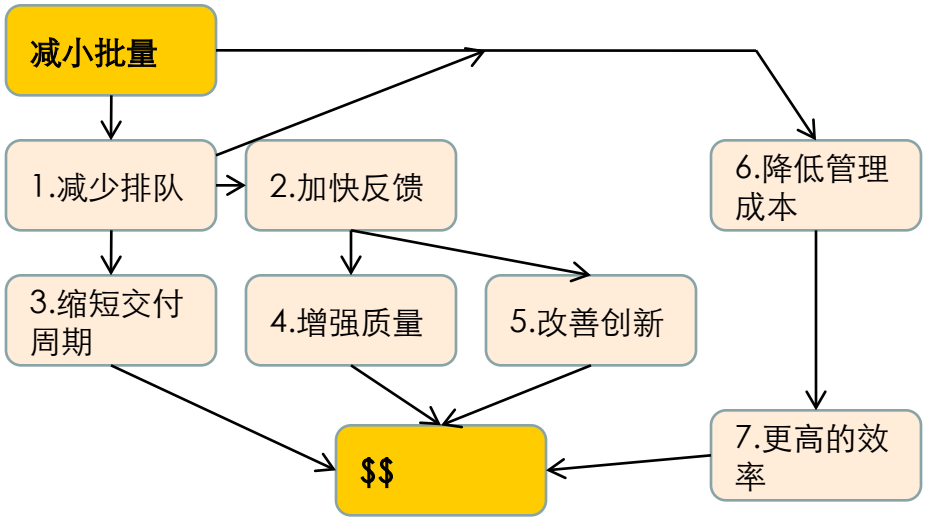
- 视频：乔布斯，石头的寓言

适应变化，小批量是快速交付的关键

排队理论：小批量与缩短交付周期、人员有效产出的关系



减少批量的好处



Source：Craig Larman

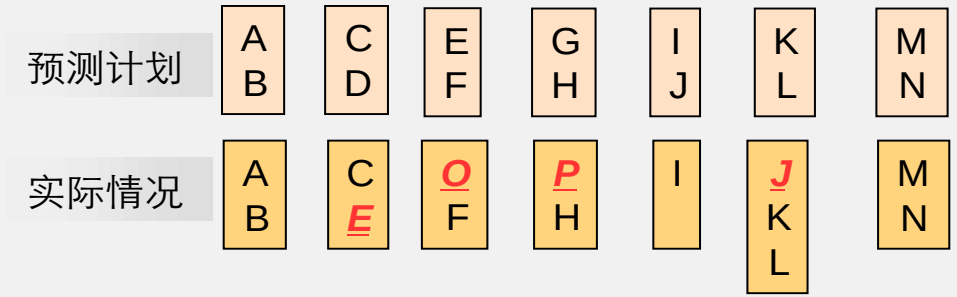
- 在需求响应周期相同的情况下，批量（一次开发的需求量）越小，资源利用率更高。
- 在资源利用率相同的情况下，批量越小，交付周期更短。

- 减小批量不仅带来缩短交付周期，而且还带来提高质量、促进创新、降低管理成本、更高的效率等其他好处，大幅提升商业价值。

- 我们首先要做的是通过尽早地、持续地交付有价值的软件来使客户满意。
- 经常性的交付可以工作的软件，交付的间隔可以从几个星期到几个月，交付的时间间隔越短越好。

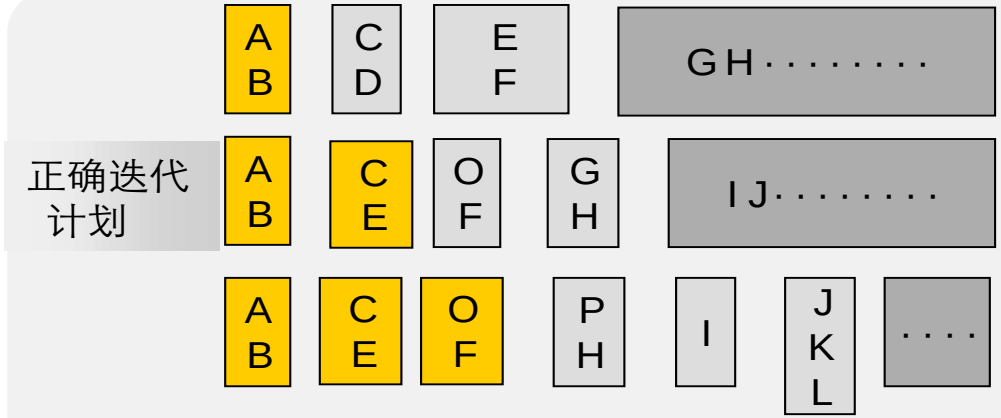
——摘自敏捷软件的十二个原则

适应变化，通过迭代计划不断调整以适应需求变化



项目范围常发生变化

- 需求出现了增加、删除、优先级调整（如图E、O、P、J）
- 工作量在需求细化后发现离原始工作量估计有偏差，引发计划调整；（如图中I）
- 客户使用了产品后，发现有些需求已不再需要（如图中D、G）



正确做计划方法

- 在每一轮迭代开始，只详细确定本次迭代的工作内容，并严格执行，对后续较远的迭代内容只做粗略的计划，避免浪费。

- 变化无法一次性预测，一开始制作大而全的计划易造成浪费
- 应根据迭代积累的经验和需求变化的情况对计划不断调整和细化

适应变化，应持续保持良好的软件架构

良好软件架构是适应变化的基石

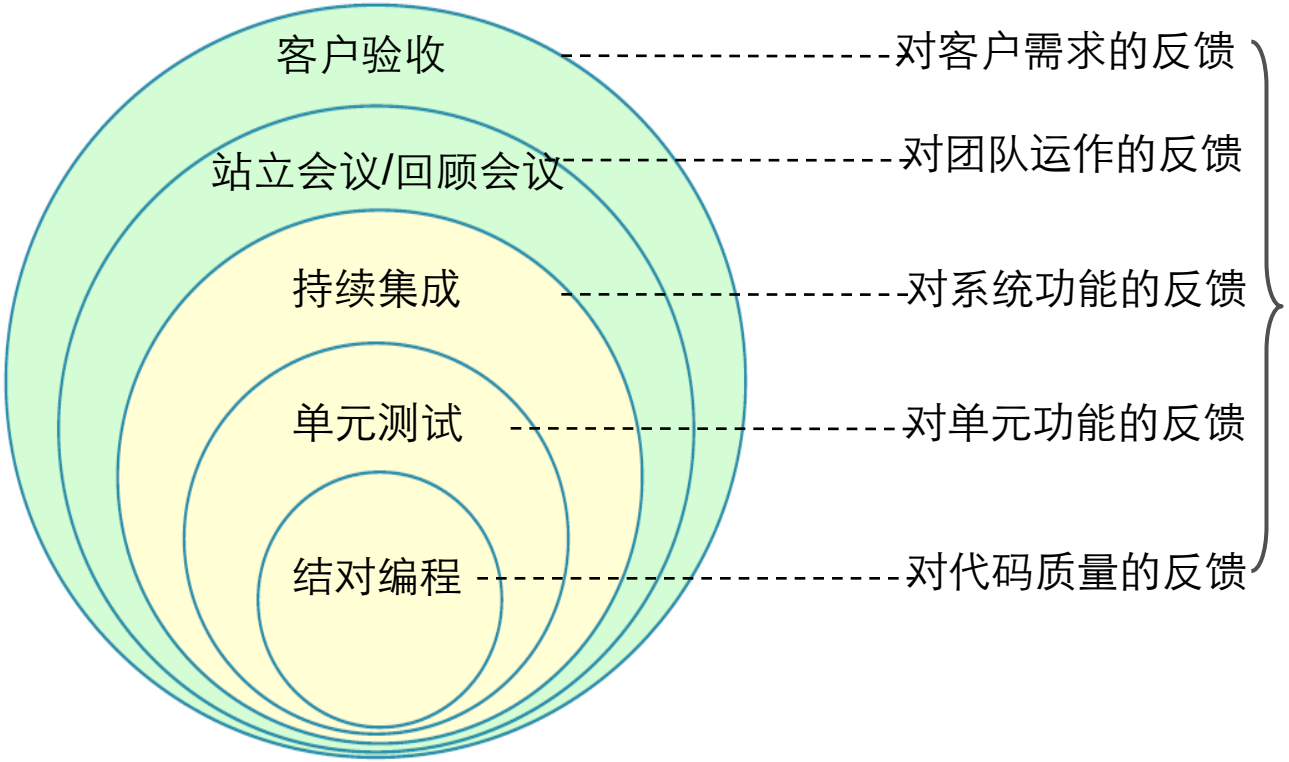
- 电信软件的特点是庞大、复杂、生命周期长，因此需要良好架构来保证长期的演进，避免大规模的返工；
- 优秀的架构通过可扩展性来很好地适应需求的变化，对敏捷起到支持作用，相反拙劣的架构会阻碍敏捷；
- 良好架构使系统部件处于松耦合状态，有助于制定出合适的增量开发/集成计划，使分层分级的持续集成更加容易实施。

软件架构需要尽早验证和持续维护

- 新产品开发通过早期迭代来实现和验证架构，有利于架构的尽早稳定；
- 增量开发需识别影响架构的需求，优先实现，规避架构风险；
- 通过重构及时维护和优化架构（偿还技术债务），使架构保持生命力。

适应变化，利用多层次反馈不断调整以逼近目标

多层次反馈手段



- 利用多层次反馈手段，在变化的环境中让团队准确地了解与目标的差距，不断调整自身行为，并逐步逼近靶心

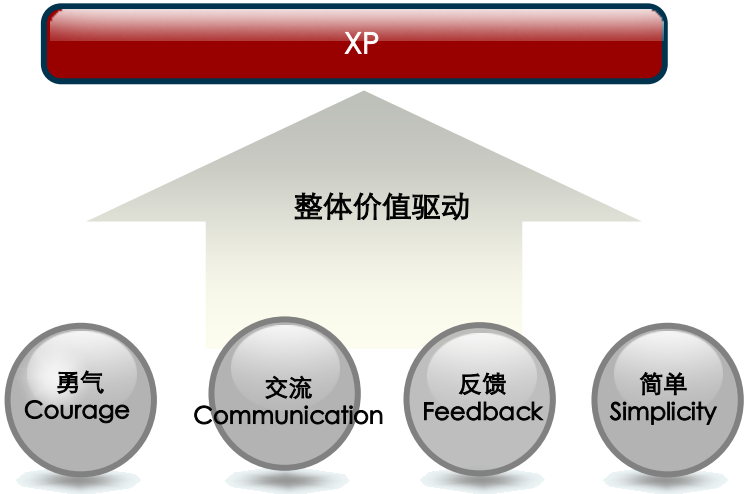
敏捷方法流派

- **Scrum**
- **极限编程，XP Extreme Programming**
- **特性驱动开发，FDD/Feature Driven Development**
- **敏捷建模，AM/Agile Modeling**
- **敏捷数据库技术，AD/Agile Database Techniques**
- **水晶方法，Crystal**
- **软件开发节奏，Software Development Rhythms**
- **自适应软件开发，ASD/Adaptive Software Development**
- **动态系统开发方法，DSDM/Dynamic Systems Development Method**
- **精益软件开发，Lean Software Development**
- **XBreed**
- **探索性测试**

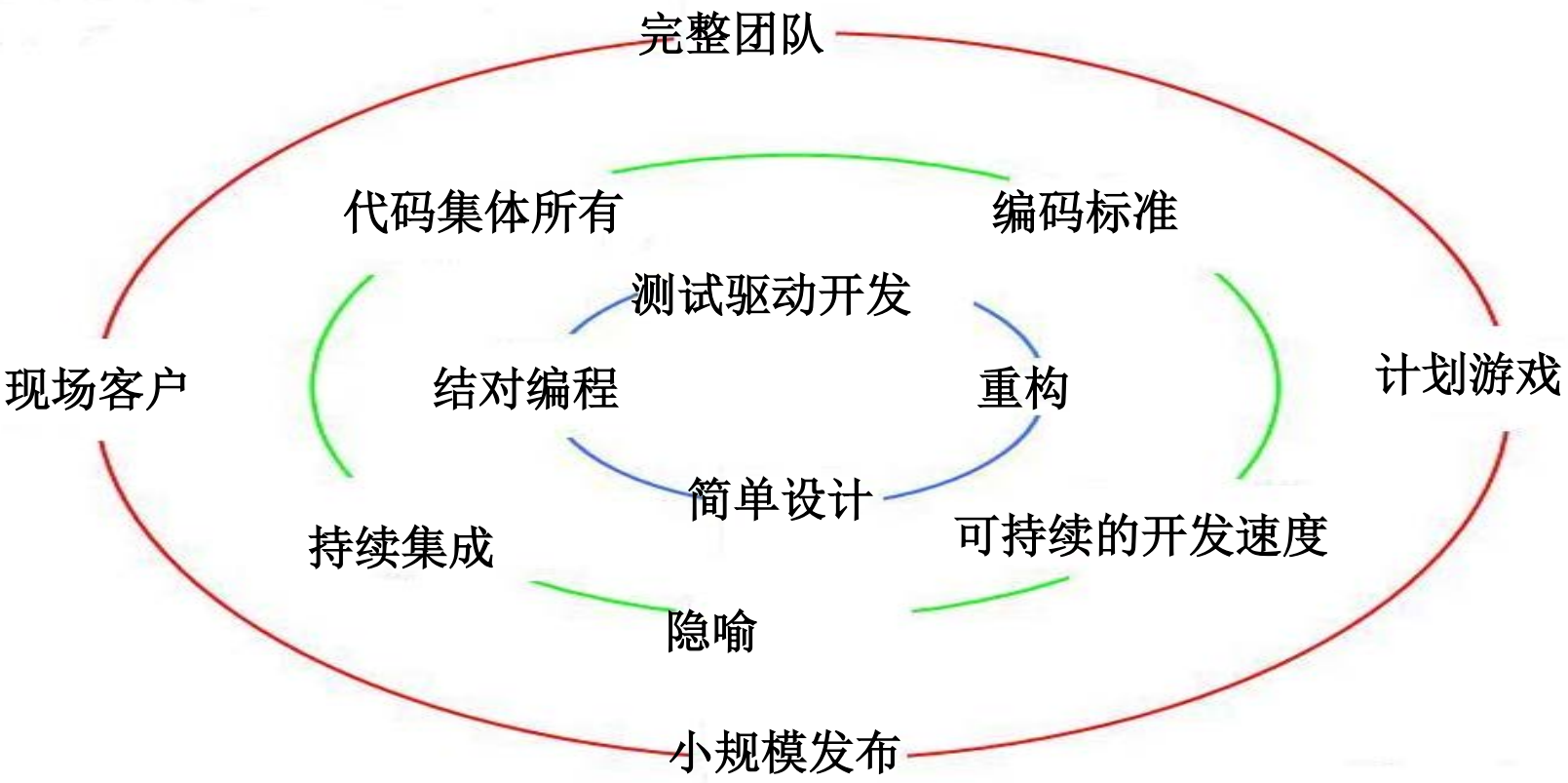
敏捷流派之XP

- **XP（eXtreme Programming 极限编程）**
  - **1996年，由Kent Beck创立**
  - **4项价值观、12项实践**
  - **技术型流派，注重开发实践**

- 最著名的敏捷流派之一，对敏捷思想的传播影响深远。



XP—12项实践



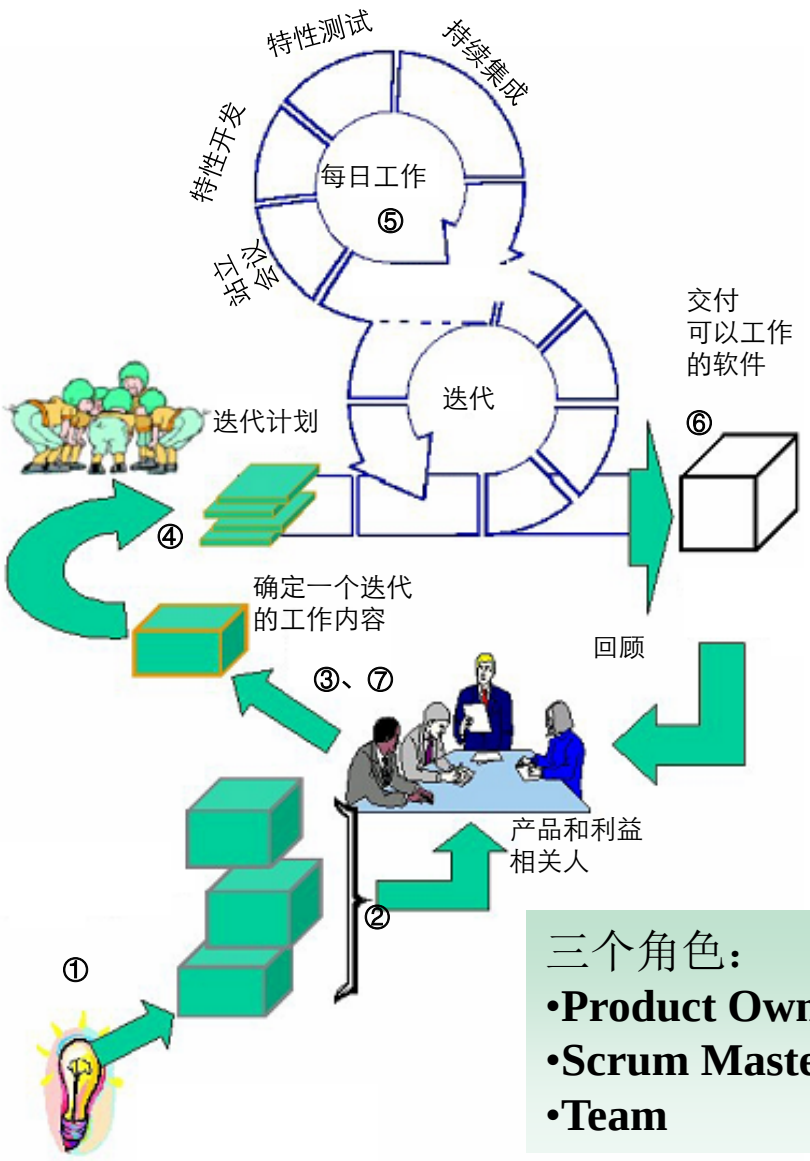
XP实践洋葱图
1层：面向编程方法
2层：小组团队活动
3层：面向项目和交付

敏捷流派之Scrum

- **Scrum**一词来自英式橄榄球：每个成员都明确自己角色，环绕同一目标，集体行动，奋力拼搏。
- **Scrum**定义了一个项目管理框架、方式，包括需求搜集、团队协作、项目运作、活力激发等。

- 核心：
  - **自管理自组织**的团队：
    公开、独立（雨伞型管理者）
  - **经验型**过程方法：
    相对于定义式过程，经验型过程不能被持续地“重复”，需要对过程持续改进。
  - **每日研讨**：更新任务、暴露并解决问题；通过回顾不断提高和学习。
  - **共同**的语言、价值观、实践

敏捷流派之Scrum——典型场景



① PO和开发团队对产品业务目标形成共识

② PO建立和维护产品需求列表（需求会不断新增和改变），并进行优先级排序

③ PO每轮迭代前，Review需求列表，并筛选高优先级需求进入本轮迭代开发

④ 开发团队细化本轮迭代需求，并按照需求的优先级，依次在本轮迭代完成

⑤ 开发团队每日站立会议、特性开发、持续集成，使开发进度真正透明

⑥ PO对每轮迭代（2—4周）交付的可工作软件进行现场验收和反馈

⑦ 回到第3步，开始下一轮迭代

三个角色：
- **Product Owner**
- **Scrum Master**
- **Team**

三个工件：
- 产品**Backlog**
- 迭代**Backlog**
- 燃尽图

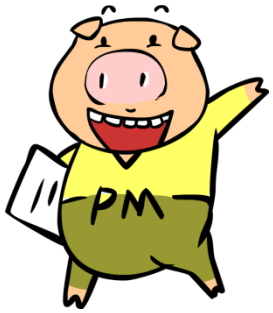
三个会议：
- 迭代计划会议
- 每日站立会议
- 迭代回顾会议

Scrum — 角色

Scrum — 角色

产品负责人
Product Owner

规划产品需求，投资回报ROI和发布计划；督促团队开发最具价值的功能



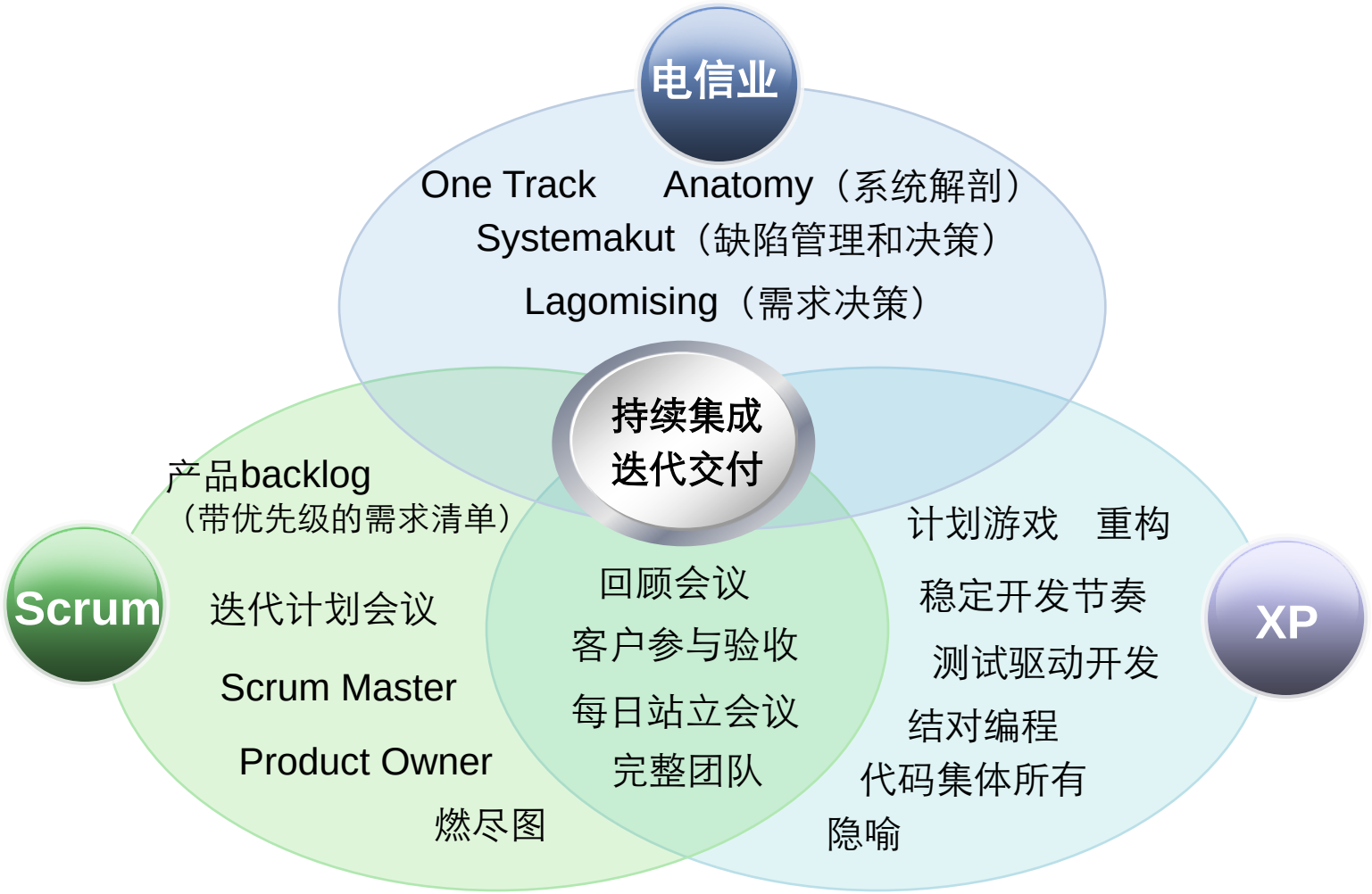
敏捷教练
Scrum Master

确保参与者都遵守Scrum的流程和规则

团队成员
Scrum Team

自组织，自管理寻找最优方案实现需求

优秀实践：业界敏捷优秀实践概览



- 电信业偏重大规模产品实践、Scrum偏重项目管理，XP偏重编程实践

具体应用：因地制宜选择适合的敏捷实践

敏捷理念

开发团队一

站立会议 + 排序的工作列表 + 持续集成 + ...

敏捷理念

开发团队二

迭代开发 + 持续集成 + 重构 + ...

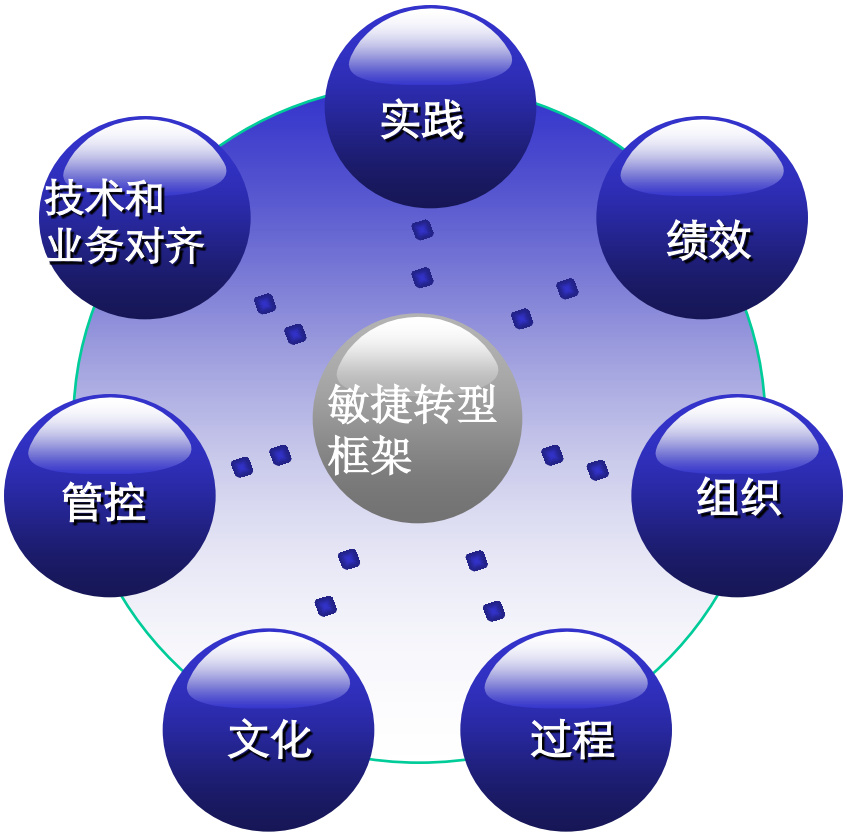
敏捷理念

开发团队三

迭代开发 + 持续集成 + 结对编程 + ...

- 团队在透彻理解敏捷理念的基础上，可以灵活选择最适合自己的实践，避免教条化

敏捷转型是系统性工程



敏捷转型7个方面优先级

	Wave1 (项目级)		Wave2 (产品级)		Wave3 (企业级)
	Stage1	Stage2	Stage3	Stage4	Stage5
Practices	1	1	2	3	4
Organization	2	2	1	1	3
Process		2	1	2	3
Performance			2	2	1
Governance			3	1	2
Culture	2	1	1	1	1
Alignment			2	2	2

Numbers represent typical relative importance at each stage.

Source：Cutter Agile Transformation（Jim Highsmith大师）

- 敏捷转型是系统工程，覆盖7个方面：实践、绩效考核、组织、过程、文化、管控、技术和业务对齐
- 敏捷在敏捷转型不同阶段，敏捷转型框架的7个方面引入的优先级不一样，初期以实践为主

对敏捷的常见误解

误解一：	敏捷开发意味着可以不需要文档、设计和计划
误解二：	敏捷只是一些优秀实践，或者是优秀实践的结合
误解三：	敏捷只适用于小项目开发
误解四：	敏捷只会对研发产生改变
误解五：	管理者不需要亲自了解敏捷，只需要管理上支持就可以了
误解六：	引入敏捷只需要按照既定的步骤去做就可以了
误解七：	敏捷是CMM的替代品，是另一种流程
误解八：	敏捷只注重特性的快速交付，在敏捷下架构不重要了