敏捷开发方法
实践分享

Walter Fan
2023-09

内容

1. 软件工程问题
2. 敏捷开发方法回顾
3. 敏捷开发最佳实践
4. 讨论

关于敏捷的箴言

图难于其易，为大于其细；
天下难事，必作于易；
天下大事，必作于细。
是以圣人终不为大，故能成其大
-

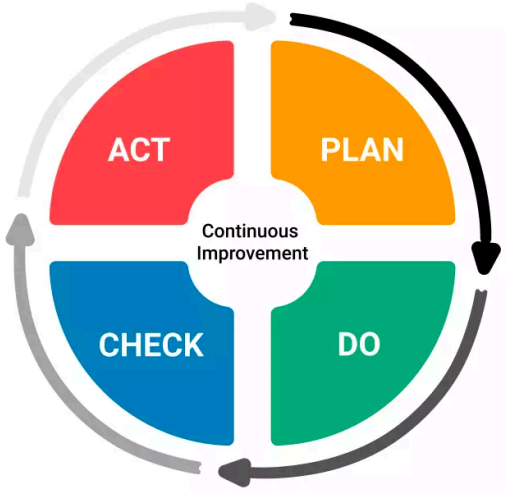
老子

不积跬步，无以至千里；
不积小流，无以成江海。
骐骥一跃，不能十步；
驽马十驾，功在不舍
-

荀子

Plans are useless but planning is indispensable

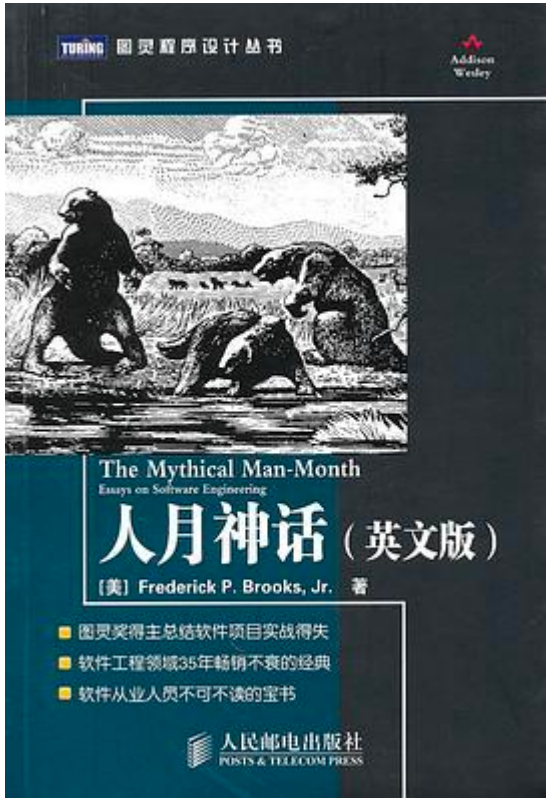
– 艾森豪威尔



软件工程问题

- 人月神话

Brooks's law: *adding manpower to a late software project makes it later*



- 梦断代码

软件开发方法

- 1.编码修复方法Code-fix: 边写边改
- 2.瀑布式方法Waterfall: 概念-需求-设计-编码-测试
- 3.进化式原型化方法Evolutionary Prototypeing:迭代开发
- 4.分步交付方法 Staged Delivery
- 5.RUP方法: www.rational.com
- 6.MSF方法(Microsoft Solutions Framework): www.microsoft.com/business/services/mcsmsf.asp
- 7.敏捷方法: XP, FD, Scrum, ASD, DSDM, etc.

瀑布方法

ERCM - Engineering Release Cycle Methodology

将软件开发过程分为需求分析, 设计,编码,测试等过程, 并以里程碑 milestone 来管理

共有如下里程碑

1. RC - Requirements Complete: 需求完成, 各方签署同意需求文档 (PRD/UI sign-off)
2. DC - Design Complete: 设计完成, 各方签署同意设计文档 (Design Spec sign-off)
3. FC - Feature Complete: 功能完成, 单元测试完成
4. CC - Code Complete: 代码完成, 集成测试完成
5. CF - Code Freeze: 代码冻结, 没有遗留严重的 bug
6. ER - Engineering Release: 工程项目正式发布

前松后紧, 不能及时的应对变化, 一旦在设计完成之后发生变化, 就会打乱原来井井有条的流程.

更要命的是, 在项目后期发现设计有重大缺陷, 或者与需求大相径庭, 需要推倒重来, 却发现时间已经远远不够了, 只能加班.

RUP - Rational Unify Process

四个阶段:

1.初始阶段 Inception

2.细化阶段 Elaboration

3.构建阶段 Construction

4.发布阶段 Transition

六个最佳实践:

•1 迭代式开发

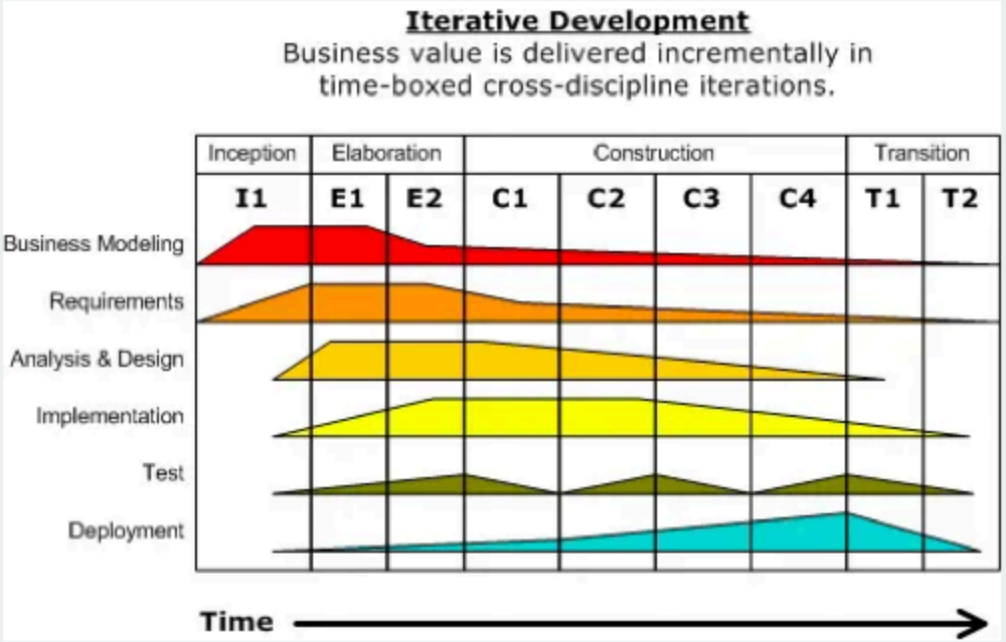
•2 管理需求

•3 使用组件

•4 可视化建模

•5 验证质量

•6 控制变化



敏捷方法

敏捷是一种方法, 代表一种精神, 让事情变得简单高效, 就象 XP 创始人 Kent Back 所说的快速地拥抱变化

所以敏捷的精神在于快速的应对变化, 告别冗长的流程, 专注于提供对于用户有价值的软件.

敏捷宣言

我们一直在实践中探寻更好的软件开发方法， 身体力行的同时也帮助他人。由此我们建立了如下价值观：

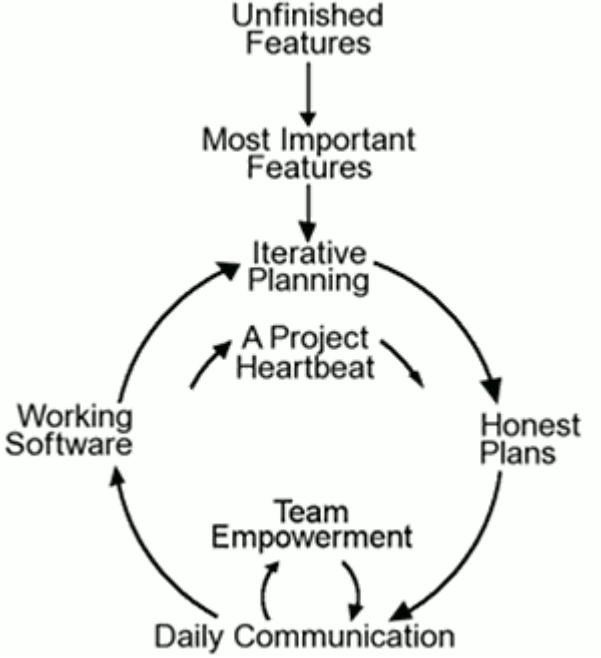
- 个体和互动 高于 流程和工具
- 工作的软件 高于 详尽的文档
- 客户合作 高于 合同谈判
- 响应变化 高于 遵循计划

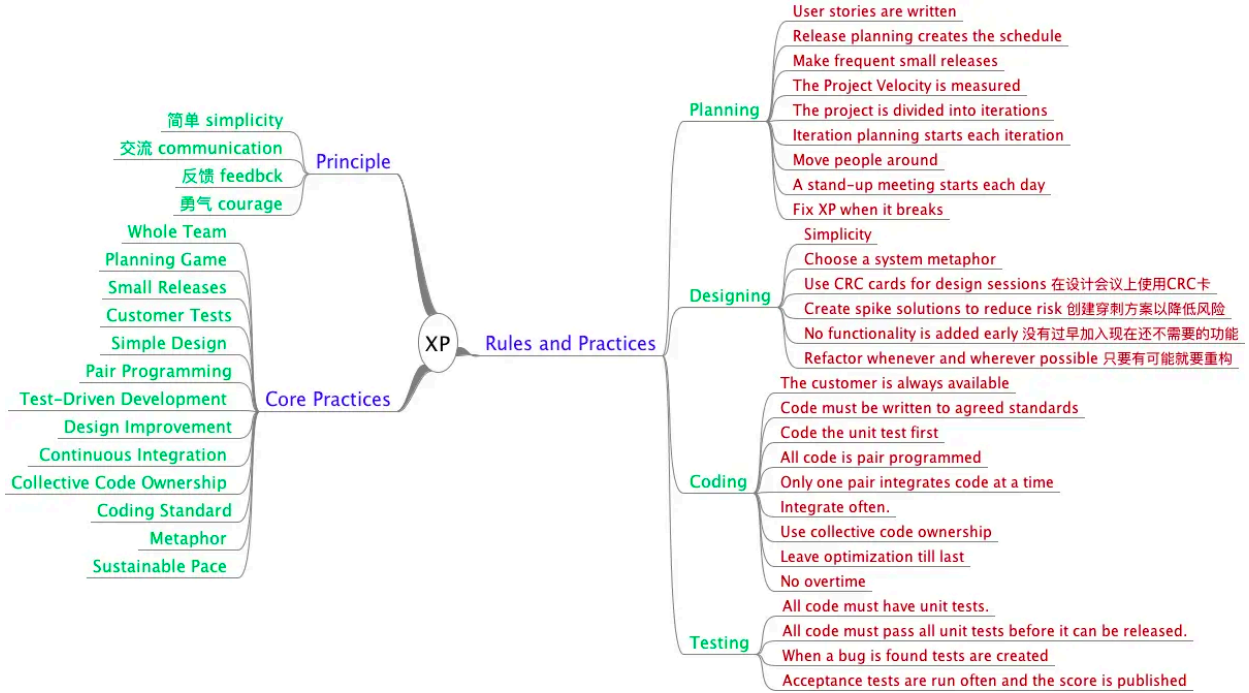
也就是说，尽管右项有其价值， 我们更重视左项的价值

敏捷原则

1. 通过快速交付有用的软件来满足客户需求
2. 欢迎不断变化的需求，即使是在开发后期
3. 工作软件频繁交付（几周而不是几个月）
4. 业务人员和开发人员之间密切的日常合作
5. 项目是围绕有积极性的个人建立的，这些人应该值得信任
6. 面对面交谈是最好的沟通方式（同地办公）
7. 可以工作的软件是衡量进度的主要标准
8. 可持续发展，能够保持恒定的步伐
9. 持续关注卓越技术和良好设计
10. 简单性 — 最大化未完成工作量的艺术—至关重要
11. 自组织团队
12. 定期适应不断变化的情况

XP 极限编程


Unfinished Features
Most Important Features
Iterative Planning
A Project Heartbeat
Working Software
Honest Plans
Team Empowerment
Daily Communication


XP
Rules and Practices
Principle
简单 simplicity
交流 communication
反馈 feedbck
勇气 courage
Core Practices
Whole Team
Planning Game
Small Releases
Customer Tests
Simple Design
Pair Programming
Test-Driven Development
Design Improvement
Continuous Integration
Collective Code Ownership
Coding Standard
Metaphor
Sustainable Pace
Planning
User stories are written
Release planning creates the schedule
Make frequent small releases
The Project Velocity is measured
The project is divided into iterations
Iteration planning starts each iteration
Move people around
A stand-up meeting starts each day
Fix XP when it breaks
Designing
Simplicity
Choose a system metaphor
Use CRC cards for design sessions 在设计会议上使用CRC卡
Create spike solutions to reduce risk 创建穿刺方案以降低风险
No functionality is added early 没有过早加入现在还不需要的功能
Refactor whenever and wherever possible 只要有可能就要重构
Coding
The customer is always available
Code must be written to agreed standards
Code the unit test first
All code is pair programmed
Only one pair integrates code at a time
Integrate often.
Use collective code ownership
Leave optimization till last
No overtime
Testing
All code must have unit tests.
All code must pass all unit tests before it can be released.
When a bug is found tests are created
Acceptance tests are run often and the score is published

精益思想

确保团队采用的流程要有助于构建对客户有价值的产品

MMF 最小市场特性

MVP 最小可行产品

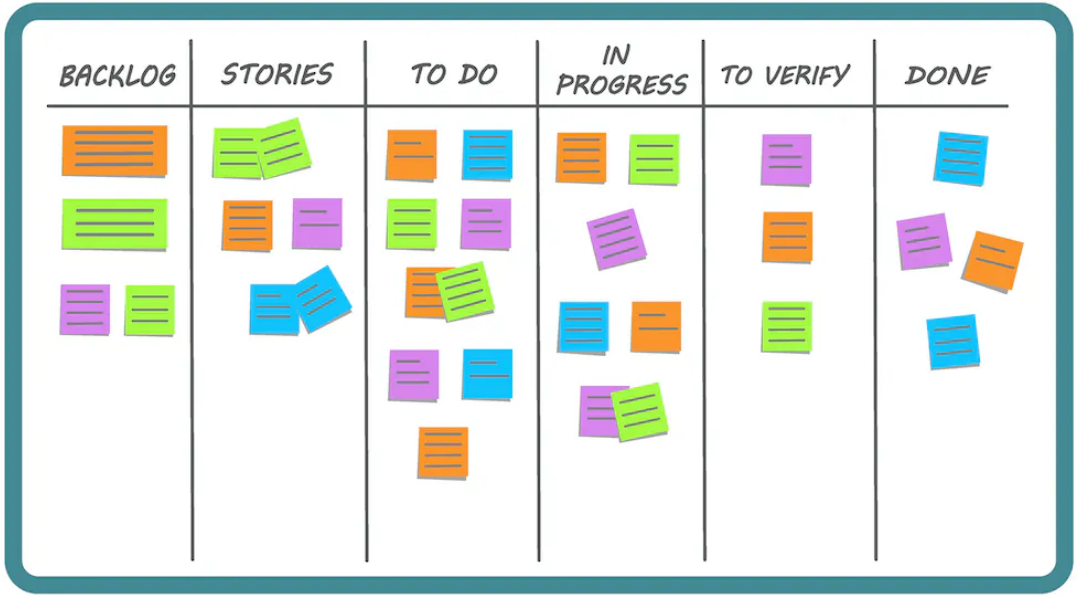
- 消除浪费
- 增强学习
- 推迟决定
- 尽快交付
- 授权团队

7大类浪费

1. 部分完成的工作
   partially done work
2. 多余的流程 extra processes
3. 多余的特性 extra features
4. 任务切换 task switching
5. 等待 waiting
6. 移动 motion
7. 缺陷 defects

看板

- 可视化工作流
- 限制 WIP
- 管理流动
- 显式化流程策略
- 实现反馈回路
- 协作式改进并且实验性进化



Scrum

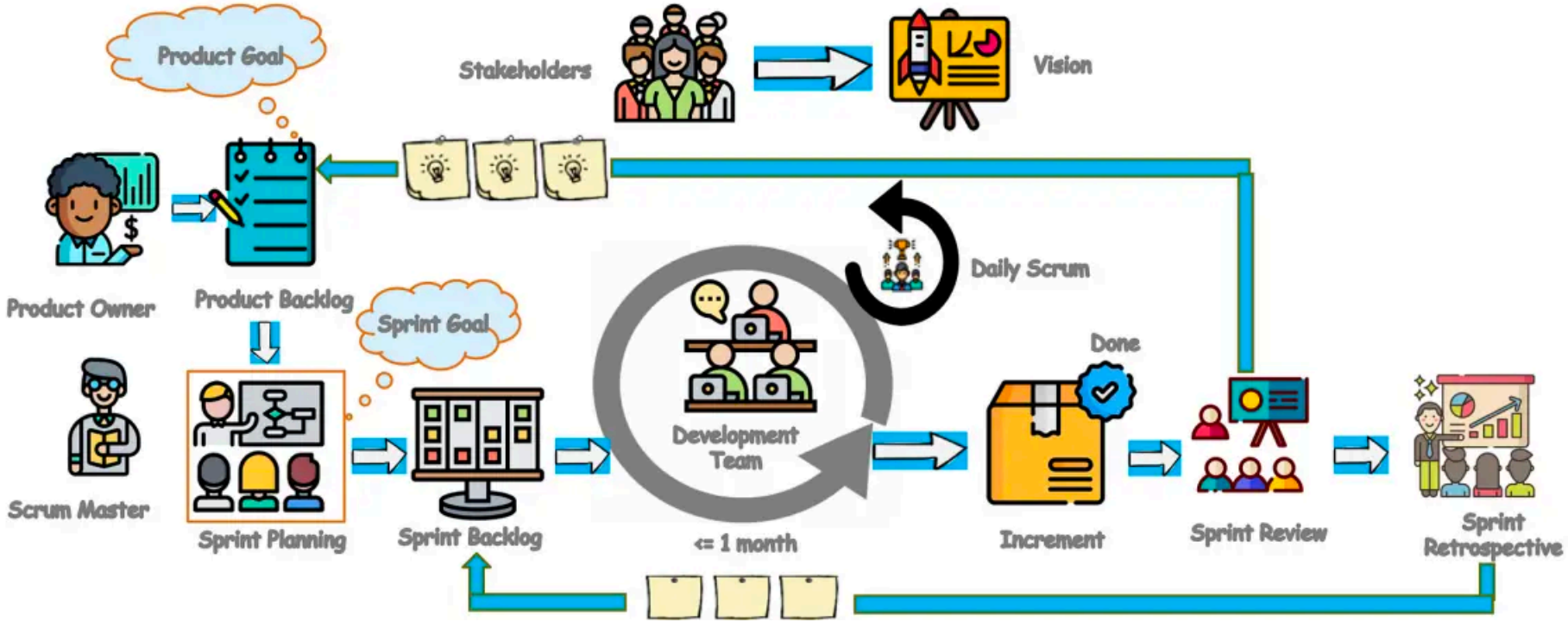
以相对固定的几周一个迭代周期(Sprint), 以跨职能,自组织的 Scrum team 持续交付对用户有价值的软件. 它比较强调**节奏感, 仪式感, 可操作性强**, 在多数互联网公司中广泛应用

Scrum 是一个轻量的框架，它通过提供针对复杂问题的**自适应**解决方案来帮助人们、团队和组织创造价值。

简而言之，Scrum 需要 Scrum Master 营造一个环境，从而：

1. 一名 Product Owner 将解决复杂问题所需的工作整理成一份 Product Backlog。
2. Scrum Team 在 一个 Sprint 期间将选择的工作 Sprint Backlog 转化为有价值的 Increment。
3. Scrum Team 和利益攸关者检查结果并为下一个 Sprint 进行调整。
4. 重复以上步骤

Scrum Framework



Scrum 的3-5-3 要点

3 种角色：

- 产品负责人（Product Owner）
- Scrum Master
- Scrum团队

5 种事件:

- Sprint 冲刺过程
- Sprint计划会议（Sprint Planning Meeting）
- 每日站会（Daily Scrum Meeting）
- Sprint评审会议（Sprint Review Meeting）
- Sprint回顾会议（Sprint Retrospective Meeting）

Scrum 的价值观

1. 承诺：致力于达成其目标并且相互支持。
2. 专注：主要关注点是 Sprint 的工作，以便尽可能地向着这些目标获取最好的进展。
3. 开放：对工作和挑战持开放态度。
4. 尊重：相互尊重，彼此是有能力和独立的人，并因此受到与他们一起工作的人的尊重。
5. 勇气：有勇气做正确的事并处理那些棘手的问题

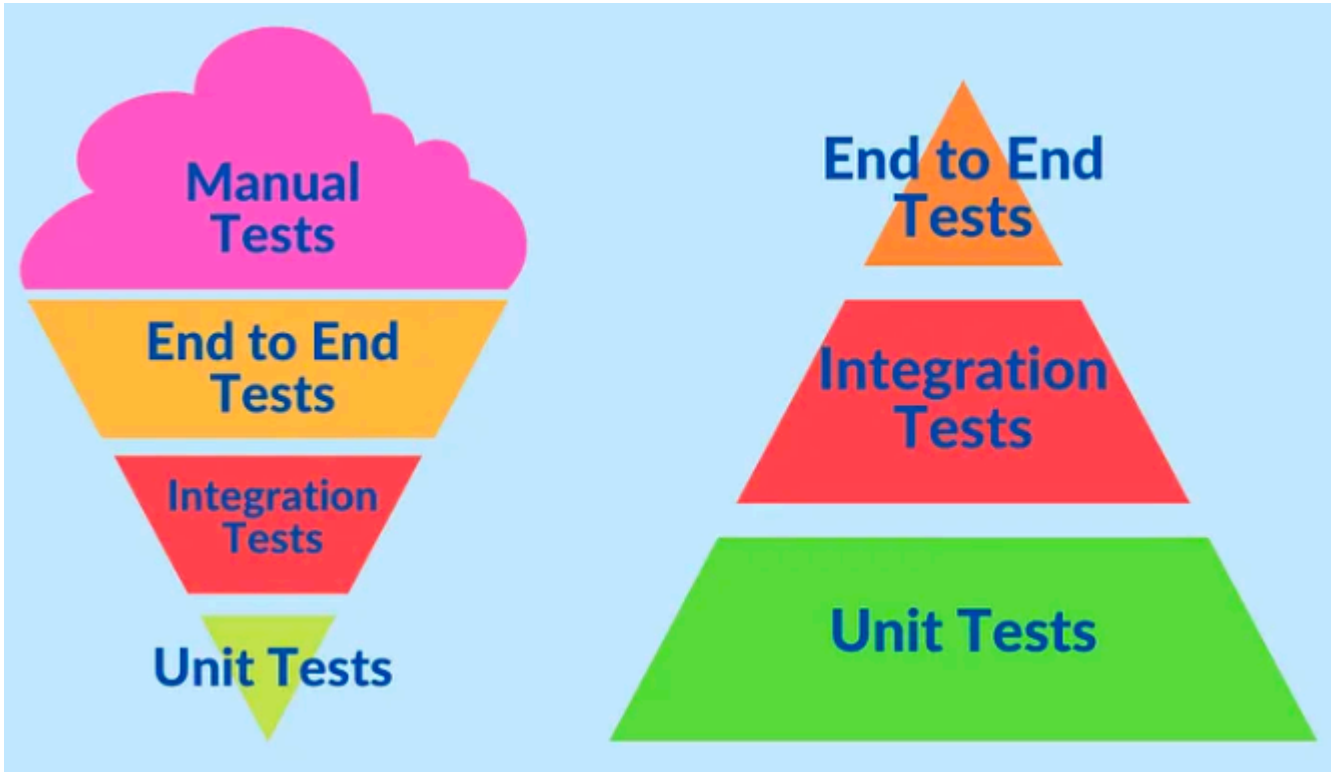
3 种工件:

- 产品Backlog（Product Backlog）
- SprintBacklog（Sprint backlog）
- Increment（产品增量）

Generally Accepted Scrum Practice (GASP)

- 可视化：看板 Kanban, 燃尽图与燃起图
- Feature 和 User story 的制订与拆分
  - 3C: Card, Conversation, Confirmation
  - Role-Action-Benefit 格式
  - INVEST 原则
- Spike & prototype
- git flow and merge request
- DevOps，GitOps, CI & CD
  - TDD, MDD, build & deployment pipeline
- 技术债: 出来混，迟早是要还的

测试冰淇淋和测试金字塔



设计和代码审查是浪费时间吗

2008年对650家公司12500个项目的质量研究表明，设计和代码审查可以消除 85% 的 bug

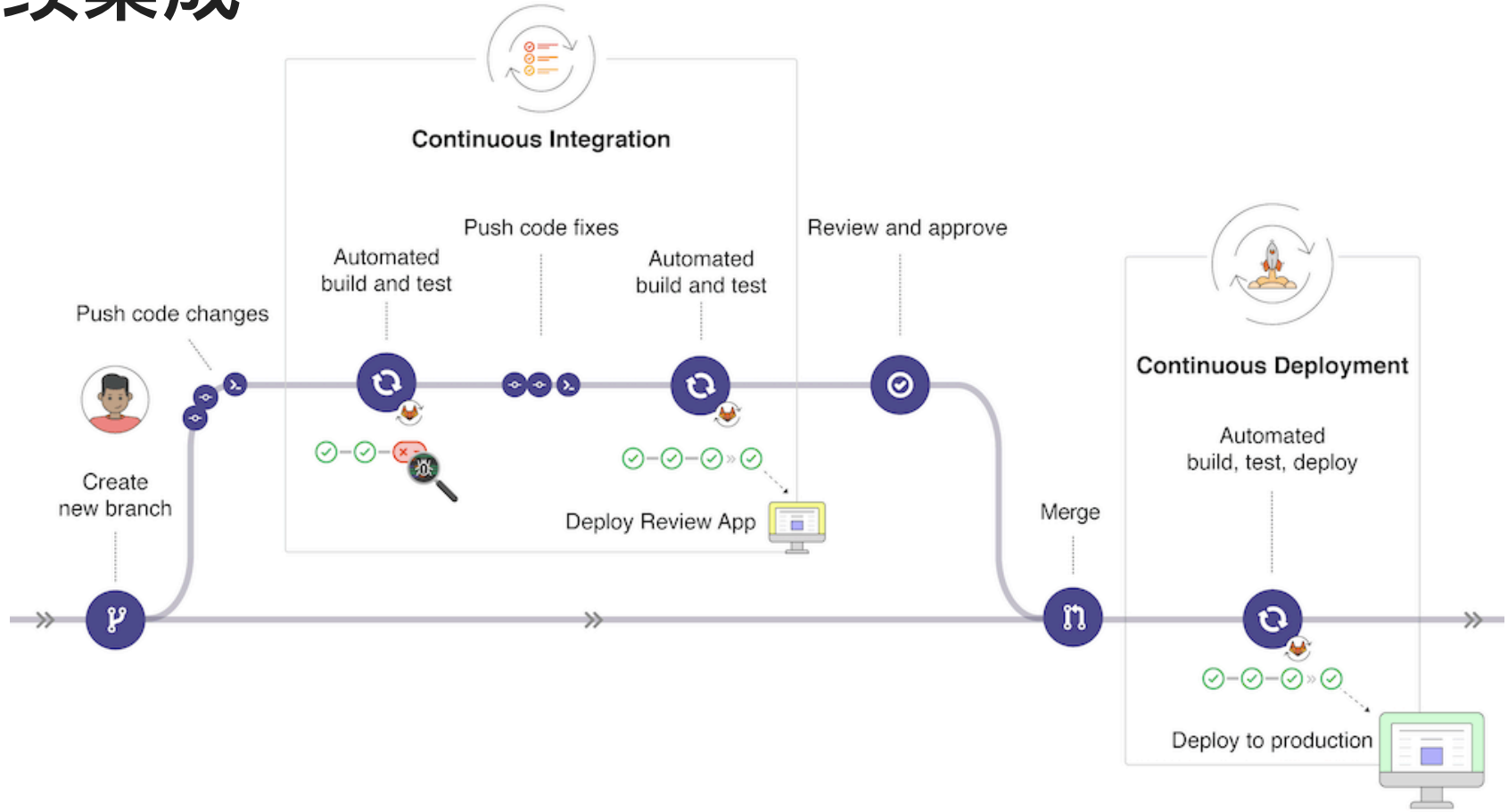
- 尽早发现错误或设计上的缺陷
- 增加做代码改动的责任心和成就感
- 就什么是好代码统一认识，做出示范
- 分享代码中的知识，在你请假时不必给你打电话
- 提高代码的可理解性和可维护性,加快迭代速度

我们可以从无到有，从少到多，从重要的改动开始做设计和代码审查

Git work flow

1. Git-Flow
2. GitHub-Flow
3. Gitlab-Flow
4. Trunk Based Development

持续集成


Continuous Integration
Push code fixes
Automated build and test
Automated build and test
Review and approve
Push code changes
Create new branch
Deploy Review App
Merge
Continuous Deployment
Automated build, test, deploy
Deploy to production

持续交付

- 目标 Goal
  - 每天(随时)都可以部署多次
  - 每个人都可以一键部署 lab, staging, production env
  - 每次部署在一小时之内可完成
- 方法
  - 主干开发 + 持续集成 + 持续交付
  - 图难于其易，为大于其细，频繁提交，逐步迭代
  - 持续分解，快速反馈，持续度量，持续改进

讨论

如何降低沟通的成本，而不是增加?

如何提高工作的效率，而不是降低?

如何成为卓越的工程师?

如何做出卓越的产品?

more...

Q & A

推荐阅读

- 人月神话
- 梦断代码
- 持续交付
- 重构 refactoring
- Scrum Guide
- Release it
- 卓有成效的工程师
- 测试驱动开发 TDD
- 领域驱动设计 DDD
- 度量驱动开发 MDD