# 敏捷

敏捷的业务目标是打造组织更早的交付价值和更灵活地应对变化的能力，软件研发从瀑布模型进化为迭代模式。敏捷实践（scrum、实例化需求、CI、自动化测试、简单设计、重构、特性团队等）服务于该业务目标。

参考：

<https://zhidao.baidu.com/question/2122593100195218107.html?ivk_sa=1022817r>

<https://www.douban.com/group/topic/15528666/>

<https://blog.csdn.net/happylee6688/article/details/22372067>

<https://www.jianshu.com/p/7a8ecb7f7226>

<https://www.cnblogs.com/yanlixin1993/p/3613501.html>

# 精益产品开发

为何有了敏捷还需要精益产品开发？

因为仅仅从关注实现阶段的迭代是不够的，我们需要端到端（用户的问题到交付用户解决方案）快速灵活交付价值。

精益产品开发的目标：顺畅和高质量的交付有用的价值。

精益产品开发的原则：探索和发现有用的价值、聚焦和提升价值流动效率

看板方法：是一种精益管理实践。看板方法的实践步骤：可视化价值流动、显示化流程规则、管理工作流动、建立反馈并持续改进。

# Scrum

Scrum是众多快速开发的敏捷方式之一，scrum是迭代的、增量的流程，迭代周期一般为1-4周，迭代周期是固定的，无论工作是否完成，从不延长。

## 运作

Scrum的三个角色：

1. PO（Product Owner产品所有者）：一般由研发经理/SE承担，PO主要负责管理产品需求及实现，保证团队做正确的事；
2. SM（Scrum Master教练）：SM是团队的组织者，保证团队正确地做事；
3. Team member：负责产品需求而的实现，保证事情完成。

Scrum的四个会议：

1. Sprint计划会议：决定需要完成哪些工作，决定这些工作如何完成；
2. Sprint评审会议：是一种活动，迭代开发过程中团队和客户代表（PO/BA）一起评审Sprint的产出物，尽早识别需求和实现的差异，以便及时调整；
3. Sprint回顾会议：迭代结束后召开，识别和处理迭代中的管理和技术问题，持续改善。通常会结合度量数据，对迭代过程进行回顾；
4. 每日站会：每日固定时间，团队全体成员参与，围绕交付目标，通过每个人的阐述（已完成、计划完成、遇到的阻碍和需要的支持），团队协同确保交付目标的达成

三个产出物：

1. Product Backlog（产品待办事项列表）：一般由PO/BA负责，即产品所有待研发的需求列表，一般包括需求名称、需求类别、需求来源、优先级等
2. Sprint Backlog（迭代待办事项列表）：纳入本迭代研发的需求列表，一般用迭代交付图的方式展示
3. BURN-DOWN CHART（燃尽图）

## 可视化

### 在制品

在制品WIP（work in process）表示已经开始但尚未完成的工作清单，在看板上通常一个卡片代表一个在制品。限制在制品WIP是明确限定在同一个状态下同一时间能有多少工作任务，保持更合理的WIP数量可以降低由任务切换带来的花销。

### 泳道

在看板上添加的横向的分界线称为泳道，设置泳道的目的是对WIP（在制品）进行分类，使可视化更加强烈。

## 特性团队

相对长期稳定、跨组建/智能独立完成端到端客户特性的团队，团队人数建议7±2。

## 组件团队

围绕系统组件进行开发的团队，一个端到端客户特性往往需要1个或者多个组件团队协同完成。

# 需求

## 需求评审/需求澄清

需求评审的目的是确保需求的正确定、完整性、一致性。需求澄清的目的是确保团队对需求的价值、功能、实现准则达成一致的理解。

## 需求分层

史诗epic：是需求定义的产物，epic对应到我们的原始需求，或者说高层次的业务目标，一般都是规划SE录入。

特性feature：feature是研发SE在epic的基础上进行需求分析或建模的产物。Feature是一个完成的产品需求或功能（包括由此引申的其他非功能需求），是逻辑上相关功能的集合，可以为用户带来完整的价值。

故事story：story是研发SE（也可能是领域专家）在feature的基础进行进一步的拆分，一般是组件需求，story不一定对用户有价值，但是必须可验证、颗粒度适中能够有效支撑工作的并行、以及快速反馈。

产品由一系列必选和可选的特性组成，每个特性包含一个或多个相互独立的组件/子特性。特性的颗粒度如果已经足够小，那么可以不拆分故事。

## 需求实例化

需求实例化是一种需求分析方法，采用一系列彼此衔接的处理模式（定系统、找用户、问目的、画场景、找功能）以及产出，帮助我们顺利完成需求分析工作。需求实例化要求以用例驱动、以场景为中心、采用迭代和增量的方法，聚焦于对“用户目的”和“使用场景”的描述。

## 需求价值评估体系与优先级

需求价值评估系统：对需求而进行有计划，有标准的价值评估，评估维度包括需求通用性、实现成本、交付紧迫程序、市场收益、用户体验、产品竞争力、现场应用/启用等。

需求优先级别的定义：根据需求价值评估体系定义需求的优先级。

## 故事点与工作量

故事点是任务的工作规模，工作量是完成任务投入的工作时间。对于相同故事点的任务，不同的人需要的工作量是不同的。

## DoD和AC

DOD是项目价值流中关键活动的完成指标，一般关注重点的DOD有：需求分析DOD、特性\故事开发DOD、特性\故事完成DOD、发布DOD。

AC（Accept Case）是验收的用例。AC可以是DOD的条款之一，可以作为它的必要非重复条件。

DOD范围更广，如开发DOD中可以定义对代码质量的需求，如通过静态检查，通过代码评审等，这些是AC无法覆盖到的。

# 简单设计

敏捷强调的是高效且持续不断地进行设计以达到客户满意的标准，敏捷中的设计是一个过程而不是事件。简而言之，敏捷的设计是一种演进式设计方法，不要求一次性将所有设计细节完成，而是反复进行设计🡪开发🡪检查的过程。这样的好处是在过程中随时响应变化，减少浪费。

## 重构

重构：就是在不改变代码外在行为的前提下，对代码作出修改，以改进代码程序的内结构，提高其可理解性，降低其修改成本；本质上说，重构就是在代码写好之后改进它的设计。

何时触发代码重构行为：代码走查、pha、交流学习、需求开发、圈复杂度检查等都可以触发重构行为，每个迭代都会对代码坏味道进行主动重构，持续改善代码质量。

如何正确的重构：先防护后重构，UT是重构的前提。

## Clean code

即整洁的代码。

# 测试

## MFQ

MFQ：是一种测试设计方法，适用于从不同角度进行测试设计。M—基于模型的简单功能测试分析和设计，F—功能交互测试分析和设计，Q—质量属性测试分析和设计。

步骤：

1. 为测试对象建模
2. 设计基础测试用例
3. 考虑测试数据的变化来确定测试用例
4. 进一步测试

PPDCS：PPDCS是MFQ使用的工具方法：P-流程图，P-决策树/决策表，D-数据表，C-组合图（正交分析法），S-状态图。

## 测试分层

测试分层是建立一种测试体系，把针对需求的测试作为一个整体来规划和执行，各层相互补位，达到质量防护的目的。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 特征 | 外部依赖性 | 运行速度 | 故障定位 | 测试范围 |
| ST | 面向特性的测试 | 存在外部依赖 | 属高层次的测试，比较耗时 | 定位精准失败根源比较困难，修复成本高 | 很宽的测试范畴 |
| FT | 两个或多个模块的交互 | 依赖外部系统 | 较快被运行 | 本身不确定，修复成本中 |  |
| UR | 测试一个代码单元的功能 | 一般不需要外部依赖 | 很快被执行 | 快读被运行，故障快速被定位，修复成本小 | 较小 |

通用原则：各层用例数具有一定的比例，提倡越往下占比越大，每层尽可能的自动化；下层能发现的故障不要留到上一层才发现。

## 测试策略

测试分层策略只是项目测试策略中的一部分，项目测试策略还包括系统测试方案，该方案中还会明确非功能性测试（性能、安全、稳定性、兼容性、易用性）测试策略及使用的工具方法。

## 自动化测试

不是所有的测试都可以自动化测试。例如，可用性测试。

## 测试前移

测试前移不是与原系统测试团队的人员前移到敏捷团队中执行各种测试，测试前移是指的测试活动的前移。

QA是一个角色，可以由团队内任何成员担任，不一定是前移的测试人员，前期一般由前移到团队的原系统测试人员担任。QA需要贯穿研发活动自始至终，从需求分析开始就需要介入（和SE结对明确验收准则）。

# DevOps工具链

DevOps：即开发运维一体化。DevOps致力于面向用户快速交付有用的价值，建立组织快速、频繁、可靠地构建验证、部署发布、运维软件的工程能力和文化环境。Devops体系庞大，落地一般都是从小做起（start small），我们目前就是从Dev模式开始切入，即从研发和测试开始导入，引入持续集成和自动化测试。

DevOps工具链是支持DevOps的工具集：

## CI分层

持续集成CI的出发点是将一切可自动化的内容自动化，为开发者提供持续快速的反馈，包含编译、检查、部署、自动化测试（UT、ST、FT、非功能测试）等。自动化测试是持续集成的最重要的奠基石。

由于交付价值流本身存在多个阶段（例如veriftCI 、mergeCI、dailyCI、性能测试等），每个阶段的关注、质量目标和反馈速度要求不一样，因此需要CI分层。

**持续分层与CI分层**

持续分层的目的是通过各种测试手段相互补位，确保产品得到充分验证。

CI分层的目的是快速反馈，合理组织已有的自动化测试的运行。

测试分层是CI分层的基础。

## Pipeline

Pipeline即软件开发流水线，通过自动化的方式将代码提交、构建、检查、验证等活动连接起来。

## 原子提交

提交一个可以独立验证软件单元，独立验证可以是UT、ST、FT中的一种或多账号。

# 度量

度量的目的是为了改进，在决策场景下，为达成业务目标（可以是业务通电也可以是业务场景，度量的就是你关心的），确保所需要的指示器。