华为IPD和敏捷的交流讨论

目录

- 华为IPD介绍
- 敏捷实践

ipd流程的作用

客户
要求

IPD – Integrated Product Development

客户需求、产品规划、Charter开发 产品开发 上市 生命周期

LTC – Lead To Cash

市场线索、机会、投标、合同订单 制造发货 安装验收 回款

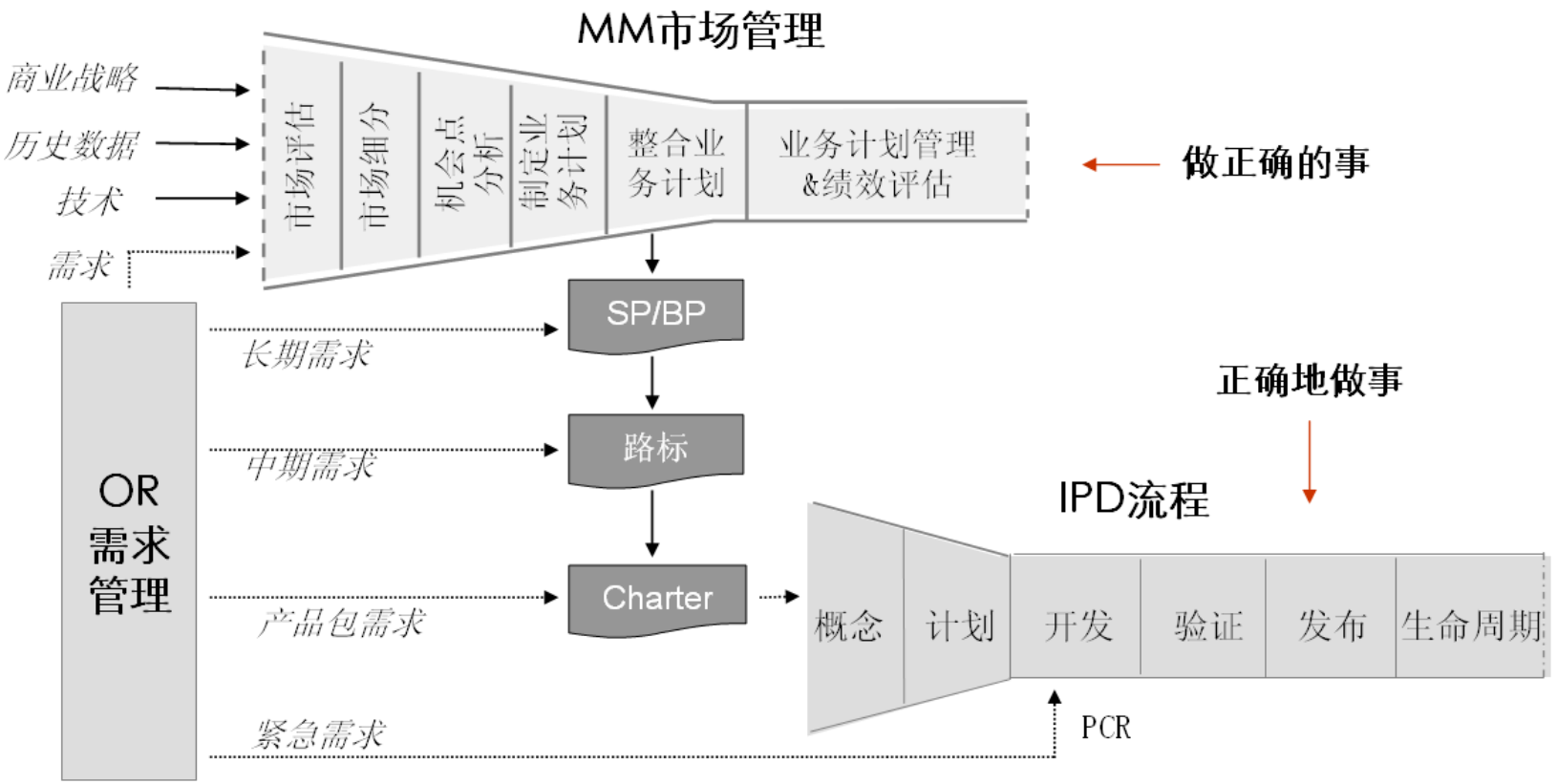
ITR – Issue To Resolution

客户投诉、网上问题 问题解决

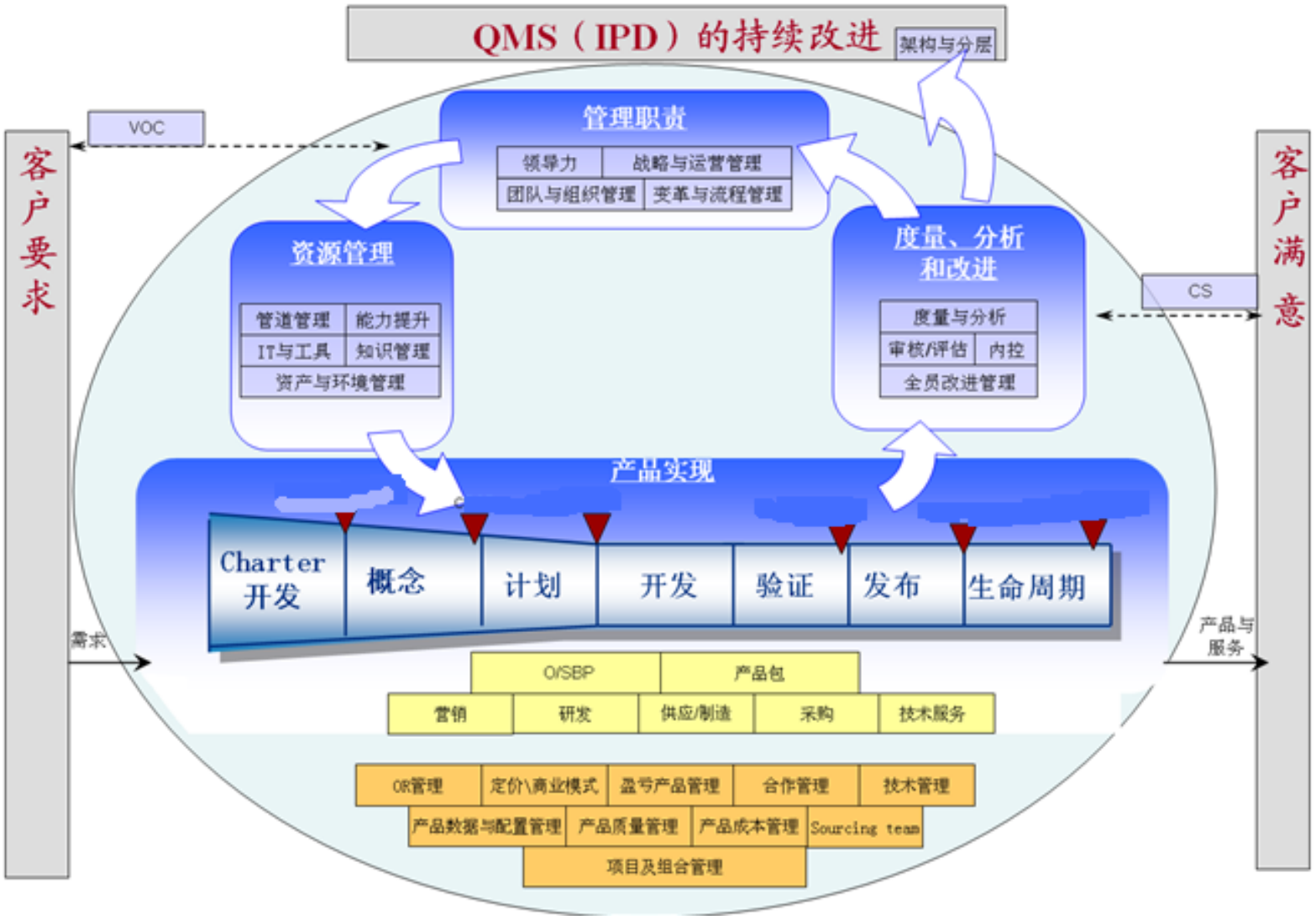
客户
满意

IPD流程

IPD主业务流包括：MM流程、OR流程、IPD流程



QMS和IPD


QMS（IPD）的持续改进
架构与分层
客户要求
VOC
管理职责
领导力
战略与运营管理
团队与组织管理
变革与流程管理
资源管理
管道管理
能力提升
IT与工具
知识管理
资产与环境管理
度量、分析和改进
度量与分析
审核/评估
内控
全员改进管理
CS
客户满意
产品实现
Charter开发
概念
计划
开发
验证
发布
生命周期
需求
产品与服务
O/SBP
产品包
营销
研发
供应/制造
采购
技术服务
OR管理
定价\商业模式
盈亏产品管理
合作管理
技术管理
产品数据与配置管理
产品质量管理
产品成本管理
Sourcing team
项目及组合管理

职能领域和IPD

- 功能领域支撑流程
  - 市场领域
  - 服务领域
  - 供应/制造领域
  - 采购领域
  - 研发领域

敏捷实施案例之一


迭代启动评估
迭代1
迭代2
……
迭代N
敏捷项目组1
敏捷项目组1
敏捷项目组1
敏捷项目组1
敏捷项目组2
敏捷项目组2
敏捷项目组2
敏捷项目组2
敏捷项目组n
敏捷项目组n
敏捷项目组n
敏捷项目组n
持 续 集 成
系统验收

敏捷12项原则

- 1、我们最优先要做的是通过尽早的、持续的交付有价值的软件来使客户满意。
- 2、即使到了开发的后期，也欢迎改变需求。敏捷过程利用变化来为客户创造竞争优势。
- 3、经常性地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。
- 4、在整个项目开发期间，业务人员和开发人员必须天天都在一起工作。
- 5、围绕被激励起来的个体来构建项目。给他们提供所需的环境和支持，并且信任他们能够完成工作。

敏捷12项原则

- 6、在团队内部，最具有效果并且富有效率的传递信息的方法，就是面对面的交谈。
- 7、可以工作的软件是首要的进度度量标准。
- 8、敏捷过程提倡可持续的开发速度。责任人、开发者和用户应该能够保持一个长期的、恒定的开发速度。
- 9、不断地关注优秀的技能和好的设计会增强敏捷能力。
- 10、简单 — 使未完成的工作最大化的艺术 — 是根本的。
- 11、最好的构架、需求和设计出自于“自组织”的团队。
- 12、每隔一定时间，团队会在如何才能更有效地工作方面进行反省，然后响应地对自己的行为进行调整。

敏捷实践概览

团队

- 敏捷团队角色
  - Product Owner（PO）
  - Scrum Master
  - Team
- 完整团队实践

工作件

- 产品Backlog（需求清单）
- 迭代Backlog
- 完成标准

迭代开发

管理实践

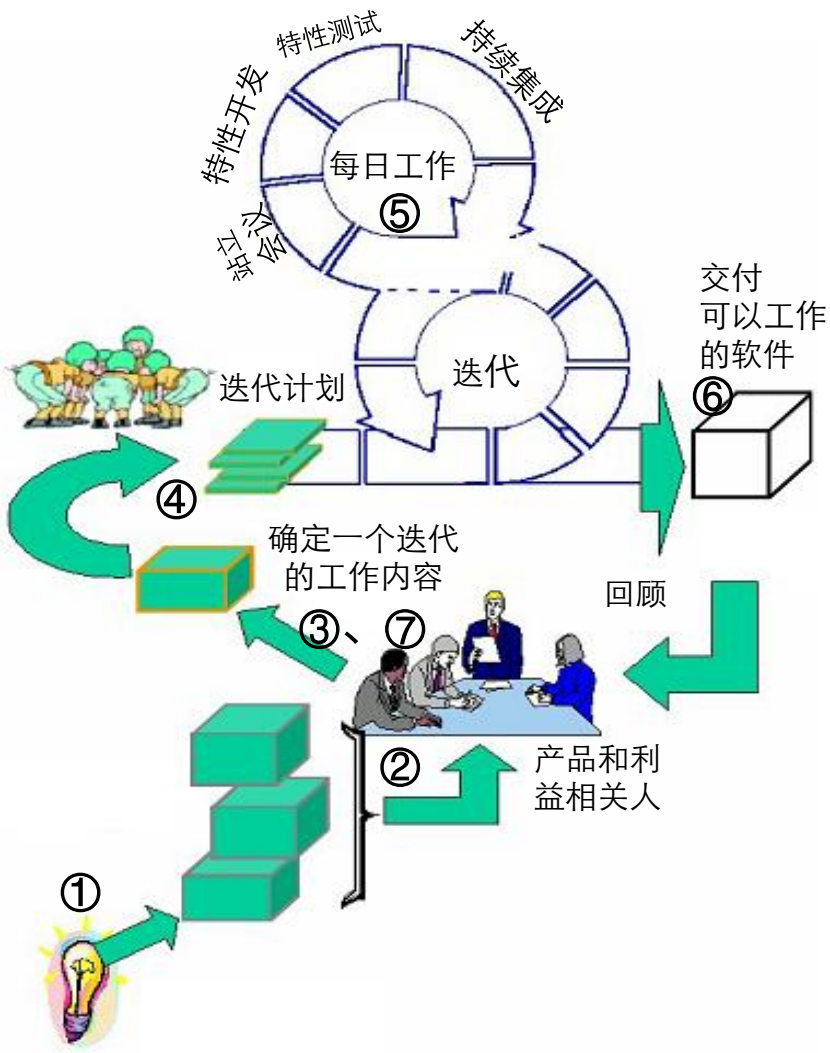
- 迭代计划会议
- 每日站立会议
- 可视化管理
- 迭代验收
- 迭代回顾会议

技术实践

- 用户故事
- 结对编程
- TDD（测试驱动开发）
- 持续集成
- Anatomy系统解剖

敏捷软件开发是以短周期迭代为核心，包含团队、工作件、管理和技术优秀实践的集合

敏捷软件开发典型场景



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

敏捷软件开发核心——迭代开发

什么是迭代式开发

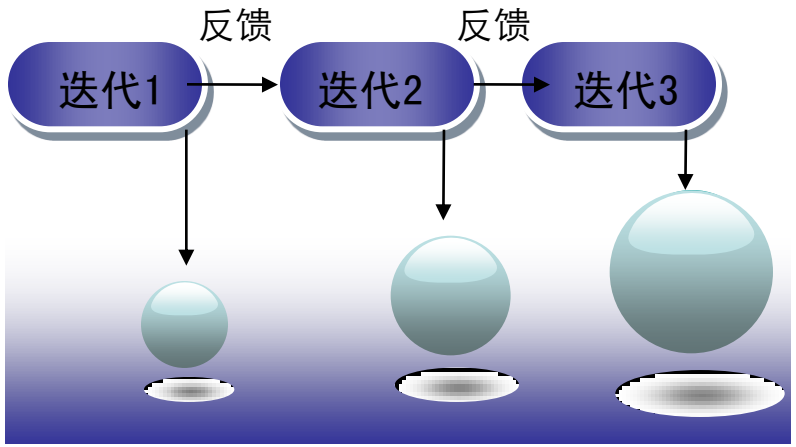
- 迭代开发将整个软件生命周期分成多个小的迭代（一般2-4周），每一次迭代都由需求分析、设计、实现和测试在内的多个活动组成，每一次迭代都可以生成一个稳定和被验证过的软件版本。

迭代式开发的好处

- 通过将高技术风险的需求在早期迭代里实现，有助于尽早暴露问题和及时消除风险
- 通过提供功能渐增的产品，持续从客户获得反馈，根据反馈及时调整，使最终产品更加符合客户的需要
- 通过小批量减少排队，提供更灵活、快速的交付能力
- 平滑人力资源的使用，避免出现瓶颈

迭代式开发的关键要点

- 每一次迭代都建立在稳定的质量基础上，并做为下一轮迭代的基线，整个系统的功能随着迭代稳定地增长和不断完善。
- 每次迭代要邀请用户代表（外部或内部）验收，提供需求是否满足的反馈
- 迭代推荐采用固定的周期（2-4周），迭代内工作不能完成，应当缩减交付范围而不是延长周期



迭代开发是有节奏地小步快跑，但建立在坚实的质量基础上

敏捷团队的角色职责

角色名称	角色定义	角色职责	注意事项
Product Owner（产品负责人）	确保**Team**做正确的事	• 代表利益相关人（如用户、**Marketing**、用服、管理者等），对产品投资回报负责 • 确定产品发布计划 • 定义产品需求并确定优先级 • 验收迭代结果，并根据验收结果和需求变化刷新需求清单和优先级	• 除了客户需求之外，内部任务如重构、持续集成环境搭建等也由**PO**纳入统一管理
Scrum Master（Scrum教练）	确保**Team**正确地做事	• 辅导团队正确应用敏捷实践 • 引导团队建立并遵守规则 • 保护团队不受打扰 • 推动解决团队遇到的障碍 • 激励团队	• 不命令和控制**Team**
Team（开发团队）	负责产品需求实现	• 负责估计工作量并根据自身能力找出最佳方案去完成任务且保证交付质量 • 向**PO**和利益相关人演示工作成果（可运行的软件） • 团队自我管理、持续改进	• 一般由**5-9**名跨功能领域人员组成 • 坐在一起工作 • 有共同的目标，共担责任 • 团队成员严格遵守团队规则

敏捷管理实践：每日站立会议

什么是每日站立会议

- 每日工作前，团队成员的例行沟通机制，由Scrum Master组织，Team成员全体站立参加
- 聚焦在下面的三个主题：
  - 我昨天为本项目做了什么?
  - 我计划今天为本项目做什么?
  - 我需要什么帮助以更高效的工作?

每日站立会议的关键要点

- 准时开始：按计划会议制定的时间地点开会，形成团队成员的自然习惯；
- 高效会议：会议限时15分钟，每个人都保持站立，依次发言，不讨论与会议三个主题无关的事情（如技术解决方案等）；
- 问题跟踪：Scrum Master应该记录下所有的问题并跟踪解决；

每日站立会议的好处

- 增加团队凝聚力，产生积极的工作氛围
- 及时暴露风险和问题；
- 促进团队内成员的沟通和协调。

每日站立会议促进团队沟通协调，及时暴露问题

敏捷管理实践：可视化管理

什么是可视化管理

- 将项目状态 (进度、质量等)通过物理实体（如白板，大屏幕）实时展示，让团队所有成员直观地获取当前项目进展信息。

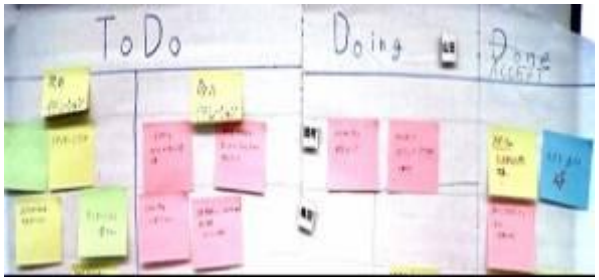
可视化管理的好处

- 简单，一目了然，降低管理成本；
- 实时状态显示，及时暴露问题；
- 信息同源使团队理解一致，提升团队凝聚力；
- 激励先进，鞭策后进，增强团队进取心。

可视化管理的关键要点

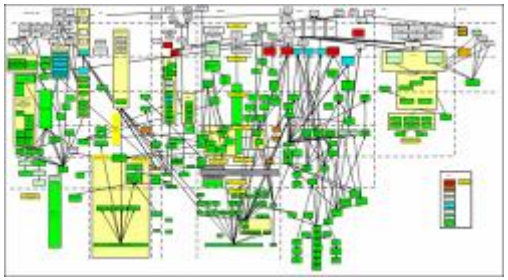
- 物理实体：可视化一定要做到物理上的实体化，大家在公开场所都容易看到，触摸到，（存在电脑中的文件不是可视化的）；
- 内容精简，易懂：信息展示一目了然，切实对团队有帮助，切忌贪多求全，难以分辨；
- 实时刷新：延迟的信息拖延问题暴露，降低运作效率。

Story墙（展示Story进度）



缺陷走势图（展示缺陷解决进展）

Anatomy视图（展示系统集成进展）



可视化管理及时暴露问题，激励团队

敏捷管理实践：迭代验收

什么是迭代验收

- 每次迭代开发结束时举行，通过演示可工作的软件检查需求是否满足客户要求；
- 由Scrum Master组织， PO和用户代表（外部或内部利益相关人）负责验收、Team负责演示可工作软件。

迭代验收的关键要点

- 展示“真实”的产品：Team 应在真实环境中展示可运行的软件，判断是否达到“完成”标准；
- 收集反馈：PO 根据验收情况及客户反馈意见，及时调整产品Backlog。

迭代验收的好处

- 通过演示可工作的软件来确认项目的进度，具有真实性；
- 能尽早的获得用户对产品的反馈，使产品更加贴近客户需求。

迭代验收尽早演示可工作的软件，收集反馈意见

敏捷管理实践：迭代回顾会议

什么是迭代回顾会议

- 在每轮迭代结束后举行的会议，目的是分享好的经验和发现改进点，促进团队不断进步；
- 围绕如下三个问题：
  - 本次迭代有哪些做得好
  - 本次迭代我们在哪些方面还能做得更好
  - 我们在下次迭代准备在哪些方面改进?

迭代回顾会议的关键要点

- 会议气氛：Team全员参加，气氛宽松自由，畅所欲言，头脑风暴发现问题，共同分析根因；
- 关注重点：Team共同讨论优先级，将精力放在最需要的地方（关注几个改进就够了）；
- 会议结论要跟踪闭环：可以放入迭代backlog中。

迭代回顾会议的好处

- 激励团队成员；
- 帮助团队挖掘优秀经验并继承；
- 避免团队犯重复的错误；
- 营造团队自主改进的氛围。

迭代回顾会议是促进团队持续改进的最有效手段

敏捷工程实践：用户故事（user story）

什么是用户故事

- 用户故事是站在用户角度描述需求的一种方式；
- 每个用户故事须有对应的验收测试用例；
- 用户故事是分层分级的，在使用过程中逐步分解细化；
- 典型的描述句式为：作为一个XXX客户角色，我需要XXX功能，带来XXX好处。

故事样例

初始需求： **1.作为网络规划人员，我想要配置一个媒体网关，因为想要增加网络容量和服务**

初次分解： 1.1作为网络规划人员，我想把媒体网关参数上传到管理系统

1.2作为网络规划人员，我想从管理系统下载媒体网关参数

再次分解： 1.2.1作为网络规划人员，我想用文件方式从管理系统下载媒体网关参数
用例：用户在管理系统上选择以文件方式下载媒体网关参数，执行成功后，检查文件是否正确下载到本地且内容正确

1.2.2作为网络规划人员，我想用MML结构方式从管理系统下载媒体网关的参数
用例:……

用户故事的关键要点

- I – Independent，可独立交付给客户
- N – Negotiable，便于与客户交流
- V - Valuable ，对客户有价值
- E - Estimable ，能估计出工作量
- S - Small ，分解到最底层的用户故事粒度尽量小，至少在一个迭代中能完成
- T - Testable，可测试

用户故事的好处

- 用户故事站在用户视角便于和客户交流，准确描述客户需求；
- 用户故事可独立交付单元、规模小，适于迭代开发，以获得用户快速反馈；
- 用户故事强调编写验收测试用例作为验收标准，能促使需求分析人员准确把握需求，牵引开发人员避免过度设计。

用户故事便于团队站在用户角度分解细化需求并制定验收标准

敏捷工程实践：结对编程

什么是结对编程

- 两位程序员在一台电脑前工作，一个负责敲入代码，而另外一个实时检视每一行敲入的代码；
- 操作键盘和鼠标的程序员被称为“驾驶员”，负责实时评审和协助的程序员被称为“领航员”；
- 领航员检视的同时还必须负责考虑下一步的工作方向，比如可能出现的问题以及改进等。

结对编程的关键要点

- 程序员应经常性地在“驾驶员”和“领航员”间切换，保持成员间平等协商和相互理解，避免出现一个角色支配另一个角色的现象；
- 开始一个新Story开发的时候即可变换搭档，以增进知识传播；
- 培养团队成员积极、主动、开放、协作的心态能够增进结对编程效果；
- 实施初期需要精心辅导，帮助团队成员克服个性冲突和习惯差异。

结对编程的好处

- 有助于提升代码设计质量；
- 研究表明结对生产率比两个单人总和低 15%，但缺陷数少 15%，考虑修改缺陷工作量和时间都比初始编程大几倍，所以结对编程总体效率更高（*source: The Economist*）；
- 结对编程能够大幅促进团队能力提升和知识传播。

结对编程提高代码质量和工作效率

敏捷工程实践：测试驱动开发（TDD）

什么是测试驱动开发

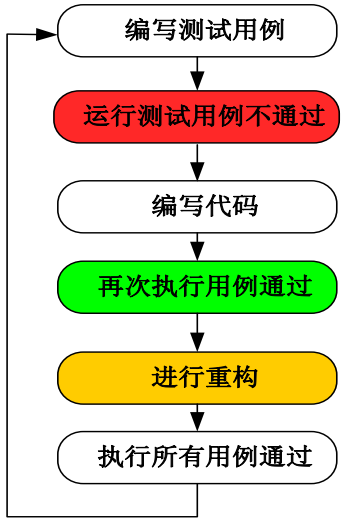
- TDD以测试作为编程的中心，它要求在编写任何代码之前，首先编写定义代码功能的测试用例，编写的代码要通过用例，并不断进行重构优化；
- TDD要求测试可以完全自动化运行。

测试驱动开发的好处

- 和代码同步增长的自动化测试用例，能为代码构筑安全网，保证代码重构的质量；
- TDD有助于开发人员优化代码设计，提高代码可测试性。

测试驱动开发的关键要点

- 测试代码和源代码一样都需要简洁，可读性好；
- 测试用例的设计要保证完备，覆盖被测单元的所有功能；
- 每个测试用例尽量保持独立，减少依赖，提高用例的可维护性；
- 当功能单元较大时，为降低难度，可分解为多个更小的功能单元，并逐一用 TDD 实现。



测试驱动开发保证代码整洁可用（Clean code that works）

敏捷工程实践：持续集成(CI)

什么是持续集成

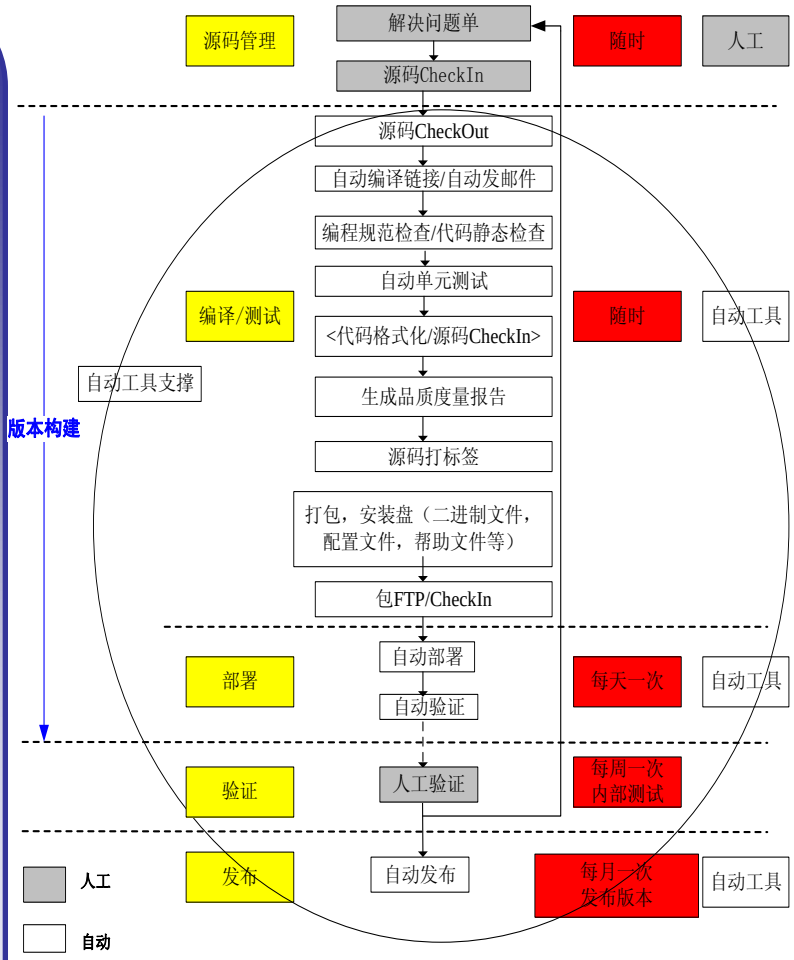
- 持续集成（CI）是一项软件开发实践，其中团队的成员经常集成他们的工作，通常每人每天至少集成一次，每次集成通过自动化构建完成。

持续集成的好处

- 大幅缩短反馈周期，实时反映产品真实质量状态；
- 缺陷在引入的当天就被发现并解决,降低缺陷修改成本；
- 将集成工作分散在平时，通过每天生成可部署的软件；
，避免产品最终集成时爆发大量问题。

持续集成的关键要点

- 持续集成强调 “快速”和“反馈”，要求完成一次系统集成的时间尽量短，并提供完备且有效的反馈信息；
- 自动化测试用例的完备性和有效性是持续集成质量保障；
- 修复失败的构建是团队最高优先级的任务；
- 开发人员须先在本地构建成功，才可提交代码到配置库；
- 持续集成的状态必须实时可视化显示给所有人；
- 大系统持续集成需分层分级，建立各层次统一的测试策略。



参见附件：持续集成解读.ppt

持续集成提供产品质量的快速反馈，保证随时拥有可工作的软件

敏捷工程实践：Anatomy系统解剖

什么是Anatomy

- Anatomy（解剖）来源于电信行业，从用户视角全面展示复杂产品系统的功能依赖关系，让整个系统的功能按自底向上逐步有序地开发和集成。

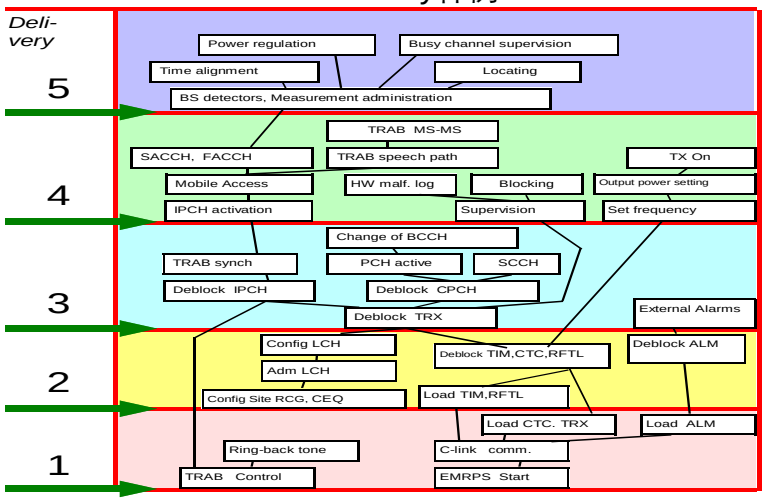
Anatomy的关键要点

- Anatomy不是系统架构视图，图中的Block是表示系统提供给用户使用的一个功能（capability），是站在纯用户视角，不包含设计信息；
- Anatomy中的依赖关系是用户使用系统功能的依赖关系，而不是设计或架构上的依赖关系；
- Anatomy是系统全视图，由最了解系统的工程师绘制出基线，在增量开发时需不断刷新。

Anatomy的好处

- Anatomy是迭代计划制定的重要依据，保证系统按照类似生物自然生长的顺序自底向上有序地开发和集成（Organic integration）；
- Anatomy也可作为可视化工具，通过标识图中每一个功能的状态，使项目整体进展一目了然，并能极大增强团队的信心；
- 有助于团队从增量交付向交付全系统的思维转变；
- 是很好的培训教材，帮助工程师了解全系统。

Anatomy样例



Anatomy帮助团队理解系统全局，制定合理的迭代计划