敏捷开发初步总结

**为什么要敏捷开发？**

我们的目标是赢得市场，快而准地满足客户的需求。但是有时问题是，客户不清楚要什么？或者客户无法清晰表达需求什么？

我们应该怎么办？

1. 通过不断的有价值软件交付，了解客户的需求
2. 通过不断的有价值软件交付，获得客户的反馈，理解客户真正的需求

我们当前面对的有两种情况，

1. 全新功能开发，面向将来的客户，目前暂无客户
2. 客户项目，但是客户不能清晰给出完整明确的需求

这两种情况，有共同点，不清楚真正的需求是什么，因此需要通过不断的尝试迭代，确定真正的需求是什么。

**敏捷宣言**

1. 个体和互动高于流程和工具。

个人理解：不完全对，但互动（即沟通）是很重要的，个体的能力也是很重要的

1. 可工作的软件高于详尽的文档

个人理解：基本同意，但这笔并不意味着不需要文档，必要的文档记录也是需要的。

1. 客户合作高于合同谈判

个人理解：在新能源汽车领域，合作的趋势要强于传统汽车领域

1. 响应变化高于遵循计划

个人理解：在实际项目中，变化总是无处不在，需要灵活机动的方式做应对

通过持续不断的交付有价值的软件使客户满意。例如每隔几星期或者一两个月，倾向于采用较短的周期。

业务人员（包括市场，产品负责人甚至客户）和开发人员必须相互合作。

欣然面对客户的需求变化，即使在开发后期也一样（但尽量能在前期），为了获得竞争优势，在敏捷过程中掌握变化。

不论团队内外，传递信息效果最好的方式是面对面沟通

可工作的软件是进度的首要度量标准

激发个体斗志，以他们为核心搭建项目，提供所需的环境和资源，辅以信任

可持续发展，产品负责人，开发人员和用户能够共同维持步调

稳定延续，坚持不懈追求卓越和良好设计

定期反思如何提高成效

**组织文化的要求**

分为3个角色

团队成员

* 非独立的个体，而是紧密团结的团队
* 基于承诺，而非监督
* 成员对于负责的工作有选择权（适度选择）
* 通过定期的回顾会议持续改进
* 接受频繁交付
* 开发和测试成为一体，消除对立
* 产品负责人，开发团队陈伟一体，消除对立
* 工作环境更容易相互沟通
* 成员之间彼此了解
* 工作环境：
  + 相邻的座位
  + Scrum工作室，快速/频繁的沟通
  + 正式会议，用于Sprint评审和待办事项梳理

经理

* 提出问题而不是给出方案
* 清除团队的障碍（通常是指行政，资源上的问题）
  + 人：保证全职人员，角色全面
  + 机：工具充足
  + 法：仅提供建议，不直接发布指令
  + 个人觉得：适度的评审也可以考虑
* 认识到团队成员的成长情况，适当放权

执行官

* 需要有这样的意识：在项目开始之前，没有方法能得到一个完善的计划，所谓的计划都是基于重要的假设的，不可能是完善的。（个人想法：计划的完整框架是必须的，最近节点的计划精细化也是必须的，二较为远期的计划则不必强求尽善尽美，因为变数太多）
* 要顶住客户对于敏捷不合理的需求，理解下属团队的工作
* 不能让团队持续筋疲力尽，这样的状态不可持续
* 关注如何管理团队才能让团队充分发挥，包括团队的产品规划，团队的架构，工作的总体计划
* 团队的脚步要缓急有序，缓思考，急行动
* 执行官是团队有序工作的护航员
* 让和敏捷团队的其他团队也敏捷起来
* 考虑一下度量团队的标准，标准不同，则对于团队有着巨大不同的影响

**理解不同类型的敏捷方法**

1. 极限编程（XP）

* 频繁发布，短周期开发

即以短周期快速迭代，这种方式依赖于测试驱动开发（TDD），以自动化测试座位快速迭代的支持。

* 结对编程

2个开发者共同完成代码（个人认为，在以MBD开发模式中，可以以2个人共同开发一个model，而且在model开发过程中频繁进行评审，最后引入第三方评审；目前的做法是，设置一个开发者和对应的评审者，并设置一个共同的第三方评审者）

* 定期构建，集成测试以及快速反馈

对于新开发功能频繁快速构建并作测试，反馈问题。由于是最新的开发内容，如果有问题，开发者比较容易分析和处理问题。

* 总结：XP是一种偏重于技术开发者的工作方法。

1. Scrum

* 把工作按照sprint（通常1-6周左右）划分开发周期，包括了设计（即需求分析和概念设计），开发（建模和编码），测试，用户验证，每一个sprint更新不同的功能。
* 产品负责人负责定义需求，需求按照优先级在不同的sprint中实现
* 每日站会，15min左右
  + 昨天完成了什么工作
  + 今天计划完成什么工作
  + 有哪些障碍
* 使用燃尽图和燃起图跟踪sprint进展
* Scrum是一种更加偏重于项目和团队管理的方法

1. 动态系统开发方法（DSDM）

* 高度的用户参与是重要的
* 团队有决策权
* 专注产品的频繁交付
* 验收标准必须明确（重要！！！）
* 迭代开发/增量交付
* 引入的改变是可逆的？？
* 以概要而非细节描述产品需求
* 测试贯穿于项目周期
* 团队的协作不可少
* 莫斯科（MoSCow）规则：必须有，应该有，可以有，不会有，按照以上4个扽及做优先级的排列

1. 精益开发

* 消除任何不增加用户价值的内容
* 定期测试代码，通过实践学习，通过原型软件获得用户据的反馈
* 尽量延迟决策，等待获得事实而不是做出假设
* 尽快交付，快速投入市场
* 允许团队做关键决定
* 嵌入完整性，个布冯座位整体一起工作？？
* 关注产品的整体而非部分体验

1. 看板

* 可视化的工作情况显示
* 可用于限制正在进行的工作数量
* 制定明确的工作流程
* 没有迭代的特性，但看板本身也不排斥迭代开发
* 没有特别强调时间盒

以上，XP，Scrum，DSDM，Lean，FDD，Kanban可以混合使用，形成适合团队工作的有效方法。

**敏捷方法的不同角色**

1. 产品负责人

* 提出产品愿景
* 把产品愿景转化为产品需求列表
* 产品需求列表按照优先级排序
* 产品需求被拆分为用户故事
* 产品需求优先级排序的原则是”商业价值”
  + 提升收入
  + 扩大市场
  + 降低成本
  + 提升客户满意度
  + 提升效率
  + 提升产品稳定性
  + 产品易用性
* 优先级排序要精细化
* 负责每个sprint的验收，确保完成了所有的目标和工作产品
* 负责大的需求发布，因为有的需求要几个sprint之后才能发布（每个sprint的工作内容不一定都要发布给客户）。
* 如果有足够多的精力，可以负责多个产品；反之，则只关注一个产品
* 如果出现多个产品负责人关注一个产品（通常是非常庞大的产品），需要保证协调的优先级是否合理
* 要考虑组织中其他的项目干系人或者团队
* 产品负责人主要做的工作是，提出做什么。

1. Scrum Master

* 工作原则：领导团队并解决可能遇到的问题
* 清除团队的工作障碍，例如成员的工作被频繁地打断，或者无法得到产品负责人的详尽需求，或者资源不足等
* 沟通团队和其他干系人，例如产品负责人，客户，公司领导等，传递有用的信息
* 保证团队按照敏捷的原则进行开发
* 如果实在较小团队，可以是兼职的角色
* 兼有项目经理（传统方式）的部分工作，也要做技术负责人的工作，要参与到实际的技术开发中
* 产品负责人的主要工作是，在”做什么”和”怎么做”之间做协调，支持，指导。

1. Scrum团队

* 通过工作协议加强团是的相互间的信任和合作
  + 每日立会机制
  + 测试策略
  + 开发策略（个人加的）
  + 对构建和基础设施的计划？？
  + 团队规范（守时，尊重估计，必要时的互助）
  + 在sprint中如何定位问题
  + 产品负责人的投入度
  + 对前面几个sprint的能力预估和计划
* 工作协议的目的是：提高团队的开发效率
* 协议的内容需要定期回顾并且更新
* 五分投票法

所有的成员达到3分以上，团队通过投票

* 自组织

自己选择希望承担的工作（个人认为，完全的自组织可能会引起混乱，但成员有一定的选择权应该是合适的）

* 团队规模在5-9人比较合适

如果是大团队，最好拆分为几个较小的团队（虽然管理复杂度增加，但团队效率可能会增加）

* 保持团队的稳定性是重要的。

1. 鸡和猪的关系

* 猪：为目标做出承诺，并亲身执行工作的成员
* 鸡：关注目标，但不以此为押注

在不同的项目阶段，不同的角色的”猪”和”鸡”的角色不同，例如在需求定义阶段，产品负责人是猪，而开发团队是鸡，而在开发阶段，开发团队是猪，产品负责人是鸡。每个角色不要越过角色，超出所赋予的角色的权利。

1. 其它

* 产品负责人

不要轻易并非其他人左右自己的判断，但也要足够开放，吸收各个方面的信息，为自己做出正确判断做准备。

* Scrum Master

把控团队的敏捷开发过程，不让其跳脱出正常的轨道。但不是一直处于一线工作，二十要让团队尘缘做出一线工作的决策（个人认为，在一些情况下必要的决策干预也是必须的）。Scrum Master的重点在于流程和开发的方法，技能，引导，培训，辅导。

* Scrum团队

避免出现破坏性的成员（可能是无法合作的个性，能力不合格等情况）

* 传统的项目经理的角色由产品负责人/Scrum Master分别承担了一部分工作，工作内容相对较小。
* 角色配置

在较小规模的公司

销售=项目发起人/关键利益干系人

业务分析师=产品负责人

IT经理=Scrum Master/架构师/应用负责人

n名开发人员 + 1名测试人员 = Scrum团队

**收集和记录需求**

1. 史诗

* 史诗是最原始的需求，范围很大，需要在多个sprint完成。（例如远程控制功能）
* 史诗按照what，who，why，when描述
  + What：所需的功能
  + Who：功能的用户
  + Why:为什么用户需要这个功能
  + When:可选项，实现的时间点

例如：作为who，想要达到what, 以便于(why)

* 随着分析，史诗被分解为多个用户故事，需求的细节被充实在用户故事内。