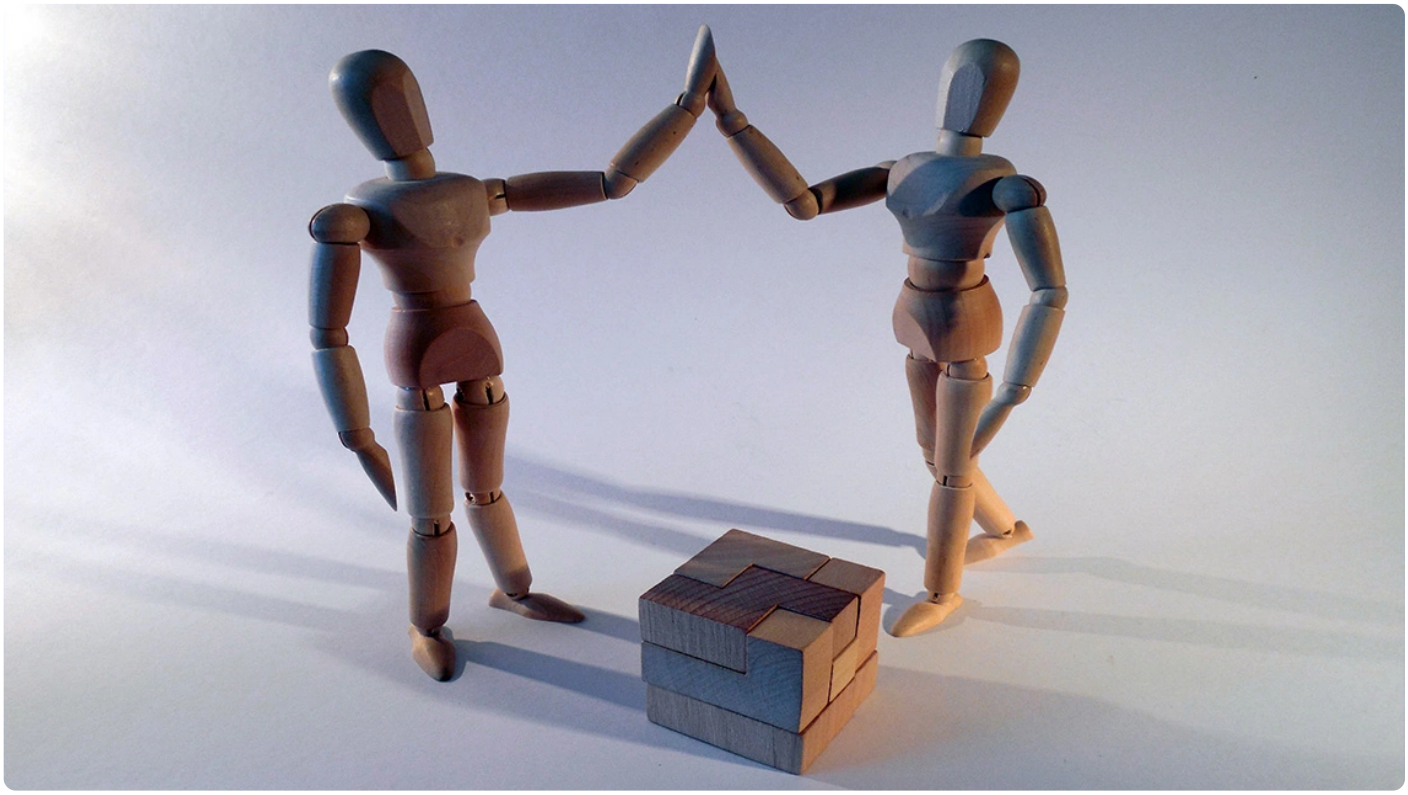
05 | 敏捷开发到底是想解决什么问题?

2019-03-05 宝玉 来自北京

《软件工程之美》



你好，我是宝玉，我今天想跟你聊聊“敏捷开发”。

关于敏捷开发的实际应用，现在无外乎有以下几种常见的情形：

> 很多团队想敏捷开发，但不知道该怎么上手；
>
> 有的团队已经应用了一些敏捷开发的实践，然而效果不理想，不知道是敏捷开发的问题，还是自己实践方式不得当；
>
> 有的团队听说了敏捷开发，但是并不知道它是什么。

为什么会这样呢？今天我们就围绕敏捷开发来谈一谈，看看敏捷开发是什么，能帮助我们解决哪些问题，要不要实施敏捷开发，以及怎么能应用好敏捷开发。

什么是敏捷开发?

那什么是敏捷开发呢？有人认为：

> 敏捷开发就是 Scrum、极限编程；
>
> 敏捷开发就是每天站立会议、每两周一个 Sprint（字面意思是冲刺，可以理解为迭代）；
>
> 敏捷开发就是把需求变成故事，把故事写在便签上贴到白板，然后根据状态移动到不同的列；
>
> 敏捷开发就是用看板软件来管理项目。

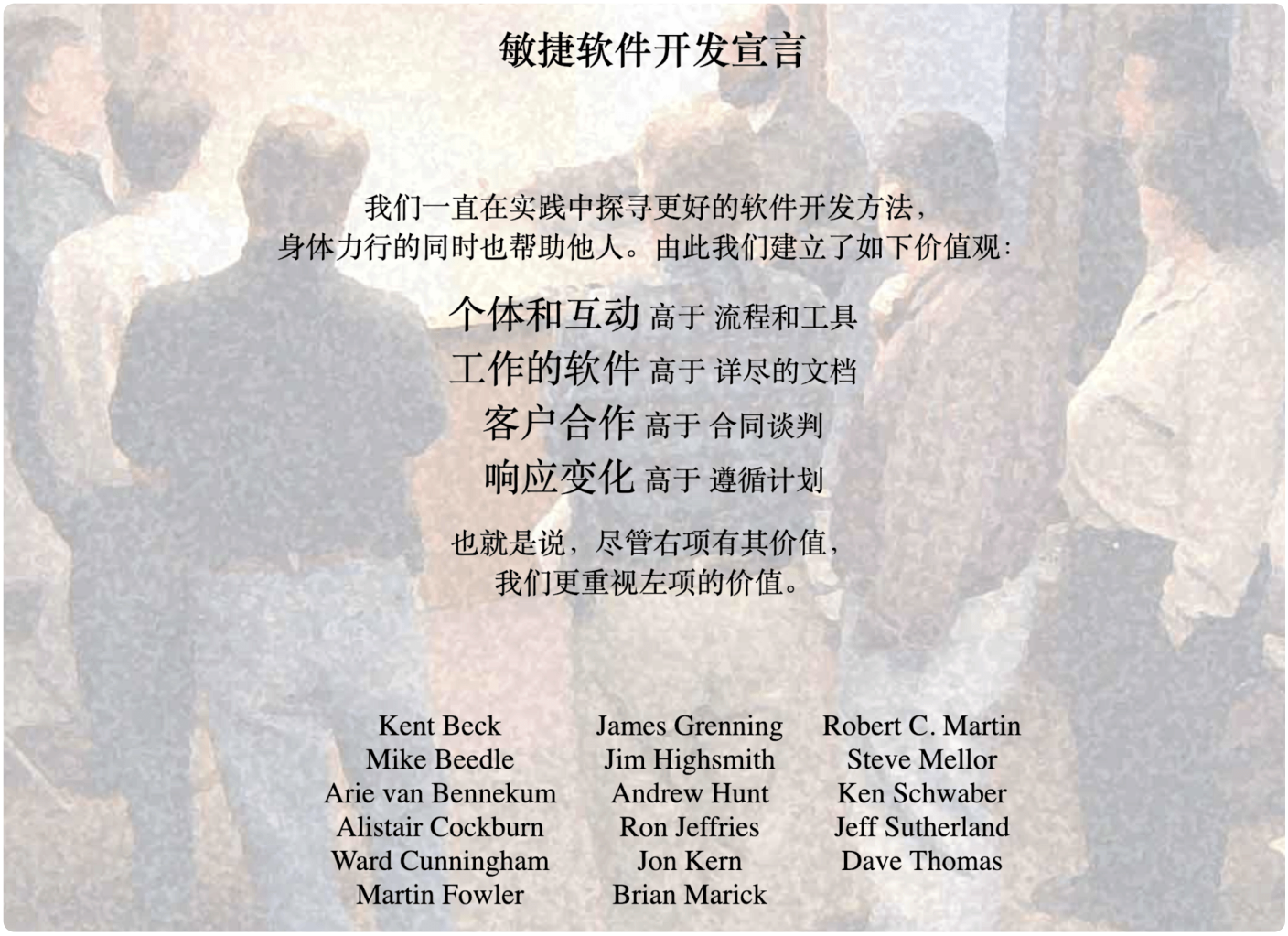
然而，这些是敏捷开发的真正含义吗？

要理解敏捷开发，我们先要了解其诞生背景。在 2001 年那会，瀑布模型还是主流，我们知道，瀑布模型是一种“重型”的开发模式，整个流程走完通常周期很长，少则数月，多则数年。长周期导致风险增加、难以响应变化。

于是由瀑布模型衍生出很多模型，试图去改善瀑布模型存在的问题，我已经在上一篇文章中给你介绍了一些。不过除了介绍的那些以外，在当时还有一些不怎么有名，而现在却如雷贯耳的轻量级开发方法，例如极限编程（Extreme Programming，XP）、Scrum 等。

2001 年初，17 位代表上述各种轻量级软件开发过程流派的领军人物聚集在一起，讨论替代瀑布模型这种重量级软件开发过程的新方法。

但是没能达成一致，所以退而求其次，把大家都认同的理念整理出来，也就是后来的敏捷宣言。这些人还一起成立了敏捷联盟。



图片来源：敏捷开发宣言

我们再回头来看前面大家对敏捷的定义，其实都是在从方法论、工具等方面解释敏捷开发。而敏捷宣言指出：

> 敏捷不是一种方法论，也不是一种软件开发的具体方法，更不是一个框架或过程，而是一套价值观和原则。

现实中关于敏捷的讨论，更多的是在讨论各种方法论和工具。不可否认，这些方法论和工具，能帮助团队“敏捷”起来，但它们和敏捷开发之间的关系，更像是“术”和“道”的关系。

> 各种敏捷框架、方法论和工具，就像是“术”，告诉你敏捷开发的方式，而敏捷则是“道”，是一套价值观和原则，指导你在软件项目开发中做决策。

这么说还是比较抽象，我给你举个例子。

敏捷开发中流行的站立会议，主要目的是为了保证团队成员充分的沟通，遇到困难可以及时寻求帮助。但是如果每天的站立会议流于形式，并不能起到有效的目的，则应该减少频度，甚至取消换成其他方式。

要不要在你的项目开发中使用站立会议，判断的依据就在于这样做是不是符合敏捷的价值观和原则。

也就是说，当你开发做决策的时候，遵守了敏捷开发的价值观和原则，不管你是不是用 Scrum 或者极限编程，那么都可以算是敏捷开发。

敏捷开发想解决什么问题?

如果你仔细读了敏捷宣言，你会发现，宣言中右边的内容其实都是瀑布模型核心的内容：流程和工具、详尽的文档、合同谈判、遵循计划。

虽然敏捷开发并未对瀑布模型的价值进行否定，但也表明了瀑布模型做的还不够好，同时提出了一套自己的价值观。

比如说，我们开始做一个新项目，需要从客户那里收集整理需求，如果按照传统的软件开发模式，我们需要在开发前获得所有需求，然后和客户签订合同，在发布前都不会轻易修改需求。

但是如果我们采用敏捷开发模式来开发项目，那这样做显然违背敏捷的价值观：“客户合作高于合同谈判”。

所以如果是敏捷开发，在每个迭代后，都应该向客户收集反馈，然后在后面的迭代中，酌情加入客户反馈修改的内容。

结合敏捷开发提出的背景，你其实不难发现，敏捷开发就是想解决瀑布模型这样的重型软件开发方法存在的问题，用一种轻量的、敏捷的方法来改善甚至是替代它。

这些年敏捷开发也是一直这么做的。**瀑布模型的典型问题就是周期长、发布烦、变更难，敏捷开发就是快速迭代、持续集成、拥抱变化。**

如果用敏捷的方式盖房子

在讲瀑布模型的时候，我拿盖房子举了个例子，如果改成用敏捷开发的模式盖房子，则会是这样子的：

客户想要盖一栋房子（**初步的想法**）。

产品经理和客户进行了初步的沟通，把用户的需求写成了一个个用户故事（**用简单的用户故事代替繁重的需求文档**），例如：

> 作为一个上班族，我想要一个卧室，以便于休息；
> 作为一个家庭主妇，我想要一个厨房，以便于做饭。

施工人员根据用户故事和客户进一步沟通（**客户合作高于合同谈判**），然后对用户故事进行设计和实现；

每个用户故事开发时，还要给一个测试机器人编写测试脚本，让机器人可以自动测试（**大量采用自动化测试**），并且做好的用户故事可以随时被测试验收（**随时发布，持续集成**）；

每个 Sprint 四个星期时间（**时间盒子，迭代时间固定**）；

第一个 Sprint 搭了个草棚，一张床就是卧室，厕所就挖了一个坑，厨房还来不及搭建（**每个 Sprint 会选择高优先级的用户故事**），屋顶还在漏水（**每个 Sprint 会定期发布，客户可以随时看到可用版本，即使还不完整**）；

第二个 Sprint 有了简易厨房，同时修复了屋顶漏水的毛病（**每个 Sprint 不仅完成用户故事，还会修复 Bug**）；

第三个 Sprint 升级成了小木屋，但是忘记加上窗户（**敏捷推崇自动化测试，但可能会测试不完备**）；

第四个 Sprint 升级成了砖瓦房，窗户也开好了，客户可以入住。但是这时候客户发现一家三口的话，完全不够用，需要扩建到 3 个卧室。于是决定下个迭代改成 3 个卧室（**响应变化高于遵循计划**）；

第五个 Sprint，升级成了 3 个卧室，升级过程中把厨房下水道弄坏了（**迭代过程中可能会导致质量不稳定**）；

第六个 Sprint，修复了下水道的问题，房子也装修好了（**迭代中不断完善**）；

客户验收使用（**上线**）。

用敏捷开发的方式，不再像瀑布模型那样有严格的阶段划分，会在迭代中不断完善；不再写很多文档，而是和客户一起紧密合作；不再抵制需求变更，而是即时响应变更；不再等到测试阶段才发布，而是随时发布，客户随时可以看到东西。

当然，采用敏捷开发的模式也存在一些问题，例如全程需要客户参与，由于测试相对少一些，问题也会相应多一些。

敏捷开发和瀑布模型的差异

由于我大学时学软件工程，那时学的就是瀑布模型，毕业后很多年的项目开发都是以瀑布模型为主的，所以我在刚开始去看敏捷开发，总会以瀑布模型的方式类比敏捷开发，实践的时候也难以摆脱瀑布模型的影响。

直到近些年，我完整的在日常项目中反复实践敏捷开发，才逐步领会到瀑布模型和敏捷开发的一些差别。

这些年敏捷开发，已经逐步发展出一套 “Scrum + 极限编程 + 看板” 的最佳实践，Scrum 主要用来管理项目过程，极限编程重点在工程实践，而看板将工作流可视化。

我将基于 Scrum 和极限编程的实践，来对比一下敏捷开发模型和瀑布模型的差异。

1. 敏捷开发是怎么做需求分析的?

瀑布模型的一个重要阶段就是需求分析，要有严谨的需求分析，产生详尽的需求分析文档。而敏捷开发的需求，主要是来源于一个个小的用户故事，用户故事通常是写在卡片上的一句话，在 Sprint 的开发中，再去确认需求的细节。

比如一个用户登录网站的需求，在用户故事里面就是一句话：

> 作为用户，我想登录网站，这样可以方便浏览。

好处是减少了大量需求文档的撰写，可以早些进入开发。但这个对开发人员在需求理解和沟通的能力上要求更高了。

2. 敏捷开发是怎么做架构设计的?

瀑布模型在需求分析完了以后，就需要根据需求做架构设计。而在敏捷开发中，并不是基于完整的用户需求开发，每个 Sprint 只做一部分需求，所以是一种渐进式的架构设计，当前 Sprint 只做适合当前需求的架构设计。

这种渐进式的架构设计，迭代次数一多，就会出现架构满足不了需求的现象，产生不少冗余代码，通常我们叫它技术债务，需要定期对系统架构进行重构。

3. 敏捷开发怎么保证项目质量?

瀑布模型在编码完成后，会有专门的阶段进行测试，以保证质量。在敏捷开发的 Sprint 中，并没有专门的测试阶段，这就依赖于开发功能的同时，要编写单元测试和集成测试代码，用自动化的方式辅助完成测试。

相对来说，这种以自动化测试为主的方式，质量确实是要有些影响的。

微软的 Windows 就是个很好的例子，在 Windows 10 之前，Windows 的开发模式是传统的类瀑布模型，有很长一段测试的时间，质量有很好的保障，Windows 10 开始，采用的是敏捷开发的模式，每月发布更新，稳定性要稍微差一些。

4. 敏捷开发是怎么发布部署的?

瀑布模型通常在编码结束后，开始部署测试环境，然后在测试阶段定期部署测试环境。测试验收通过后，发布部署到生产环境。

在敏捷开发中，这种持续构建、持续发布的概念叫持续集成，因为整个过程都是全自动化的，每次完成一个任务，提交代码后都可以触发一次构建部署操作，脚本会拿最新的代码做一次全新的构建，然后运行所有的单元测试和集成测试代码，测试通过后部署到测试环境。

持续集成是一个非常好的实践，极大的缩短和简化了部署的流程，而且自动化测试的加入也很好的保证了部署产品的质量。前期搭建整个持续集成环境需要一定技术要求。

5. 敏捷开发的 Sprint 和迭代模型的迭代有什么区别?

在上一章我介绍了增量模型和迭代模型，这两种也是一种快速迭代的方式，那么敏捷开发和迭代模型的区别是什么呢?

我们假设有两个团队，都要实现一个简单的用户系统，一个团队用迭代模型，一个团队用敏捷开发（Scrum），一个迭代 /Sprint 的时间周期都是 2 周（10 个工作日）。

迭代模型所在的团队，产品经理会先花 2 天时间去分析需求，写成需求分析文档，架构师会花 3 天时间来做设计，程序员会花 3 天时间编码，测试再花 2 天时间去测试，最后上线用户系统。

再看敏捷开发的团队，Product Owner（类似于产品经理）会把需求拆分成了几个简单的用户故事：用户登录、用户注册、找回密码、修改资料，然后放到当前 Sprint 的 Backlog（任务清单），Team（开发团队）成员开始从 Backlog 选择用户故事。

程序员 A 选了“用户登录”这个用户故事，他会去找 Product Owner 确认需求细节，之后动手实现这个用户故事。

功能完成后，同时程序员 A 还写了单元测试代码和集成测试代码，对登录的功能写了自动化测试。完成后，通过持续集成工具测试和部署到测试环境。部署完成后，用户登录功能就可以进行使用了。

这个过程，程序员 A 可能花了 4 天时间，做完“用户登录”这个用户故事之后，他又开始继续选取“找回密码”的用户故事来做，4 天时间也完成了。

其他程序员也和程序员 A 一样，他们也会从 Backlog 选择一些用户故事来做。

当团队中第 1 个用户故事部署完之后，测试人员就开始帮助测试，发现的 Bug 都提交到了 Backlog，程序员们在完成用户故事后，开始着手修复这些 Bug，正好在最后 2 天都修复完成。

从上面的例子，你可以看出，迭代模型本质上是一个小瀑布模型，所以在一个迭代里面，需要完整的经历从需求分析，到设计、编码、测试这几个完整的阶段。

所以像瀑布模型一样，刚开始测试的时候是不稳定的，到测试后期才逐步稳定下来，一般迭代前期也会相对轻松一点，而后期测试阶段可能会时间很紧张。

敏捷开发的 Sprint 中，没有像瀑布模型那样严格的开发阶段划分，而是一个个循环迭代的 Sprint。举例来说，一个瀑布模型的项目，可能会按照阶段分成：2 周需求分析，2 周设计，4 周编码，2 周测试，然后上线发布，一共 10 周。如果用敏捷开发的方式来进行，那么可能会是每 2 周一个 Sprint，每个 Sprint 结束后，都会发布上线，每次发布的可能只是完整功能的一部分，但是每次发布的都是一个可用的版本，通过多个 Sprint 的迭代，最终完成项目开发。

具体到每一个 Sprint 的开发周期中，在一个 Sprint 中会有多个小的开发任务，这些开发任务主要是新功能的开发和 Bug 的修复。由于每个 Sprint 周期很短，所以不能像瀑布模型那样有充足的时间去做需求分析、设计和测试，那么敏捷开发中怎么保证质量呢?

在敏捷开发中，通常用“用户故事”这样的方式来代替传统的需求分析，也就是以用户故事的形式，对一个需求进行简单的描述，配合关键的测试用例，并且和需求方的紧密沟通，让开发人员可以理清楚需求；通过“只做刚刚好的设计”来节约设计上的时间；通过“自动化测试”、“持续集成”来提升测试效率。

相对来说，敏捷开发中，整个 Sprint 的节奏是比较恒定，产品也是相对稳定的，即使用户故事没有完成，也不影响版本的发布。

因此，敏捷开发更注重软件开发中人的作用，需要团队成员以及客户之间的紧密协作。

该不该选择敏捷开发?

该不该选择敏捷开发，是很多团队纠结的问题。毕竟关于敏捷，有很多在中国落地失败的例子，是不是这种方法在国内水土不服？

其实，敏捷开发无论国内还是国外，大厂还是小厂，都已经有无数成功案例。这些年，软件工程中一些好的实践，像持续集成、测试驱动开发、结对编程、看板等都来自于敏捷开发。可以肯定，敏捷开发是一种非常好的软件开发模式。

但在应用上，也确实需要满足一些条件才能用好，例如：

团队要小，人数超过一定规模就要分拆；

团队成员之间要紧密协作，客户也要自始至终深度配合；

领导们的支持。敏捷需要扁平化的组织结构，更少的控制，更多的发挥项目组成员的主动性；

写代码时要有一定比例的自动化测试代码，要花时间搭建好源码管理和持续集成环境。

所以在选择敏捷开发这个问题上，你先要参考上面这些条件。

因为敏捷开发对项目成员综合素质要求更高，做计划要相对难一些。如果团队大、客户不配合、领导不支持，再好的敏捷方法也很难有效实践起来。

如果你要实践敏捷开发，建议先找个小项目进行试点，能证明可行了，再进一步推广。有条件的话，可以和一些顾问公司合作，请人做专门的培训和指导。

如果不具备条件，应该考虑先把其中一些好的实践用起来，比如说持续集成、每日站会、自动化测试等。

总结

我们今天一起学习了什么是敏捷开发，也就是敏捷开发是一套价值观和原则。也对比了瀑布模型和敏捷开发，其中的差异还是很大的。

瀑布模型面向的是过程，而敏捷开发面向的是人。敏捷开发要解决的，恰恰是瀑布模型中存在的一些问题。

最后，在要不要用敏捷开发这个问题上，不用过于纠结，看好敏捷开发，那就放心去用，觉得时机还不成熟、还不够了解，就先试点或者只是先借鉴其好的实践。

软件开发，最核心的是人，而不是用什么方法，以前没有敏捷开发只有瀑布模型的时候，也一样诞生了大量伟大的软件，像 Windows、Office。现在有敏捷开发，更多的是让我们多了一些选择。

在下一篇文章，还会再从大厂如何应用敏捷开发的角度，继续讲一讲敏捷开发的应用。

另外，敏捷开发涉及内容还是比较多，如果想有更多了解，可以阅读一些书籍作为专栏的补充。

除了“学习攻略”中推荐的一些书，还有像《用户故事与敏捷方法》《敏捷武士：看敏捷高手交付卓越软件》等这些敏捷实践的书籍也可以辅助看看。

课后思考

实施敏捷开发能给你的项目带来哪些好处？如果要实施，你打算从什么地方入手？如果已经实施了敏捷开发，你觉得用法对吗？有哪些做的好的或者不好的地方？欢迎在留言区与我分享讨论。

感谢阅读，如果你觉得这篇文章对你有一些启发，也欢迎把它分享给你的朋友。

精选留言 (57)

纯洁的憎恶

2019-03-06

流程、工具、文档、合同、计划都是工业化的标志。它们带来了稳定的质量、惊人的效率、

超大规模的协作，对于软件工业也是如此。

然而软件工业具备轻资产、知识密集型、从业人员素质高等特点，充分发挥人的创造力和价值，是其相较传统工业更高阶的要求。加之软件工程面对的不确定性与复杂度更显著。于是“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”的敏捷思想应运而生。

通过用户故事，理解用户需求。在迭代中采用渐进的架构设计。定期重构解决技术债务。功能开发的同时编写自动测试代码。自动化持续构建。

由于淡化了部分工业思维中兼顾稳定、质量、效率、成本的传统手段，敏捷思想的最终落地，需要素质极高的从业人员参与其中，且数量不宜过多，以此来弥补流程上的缺失。同时要团队与客户紧密协作，上级的充分信任，才能够有效发挥其灵活应变，又万变不离其宗的优势。这是大胆的返璞归真，好似回到了瀑布模型前的蛮荒时代，实则是更高级的打法，就像独孤九剑一般。所以，敏捷开发“道”的属性更浓。

敏捷开发具有快速迭代、持续集成、拥抱变化等诱人的特点，但也有苛刻的条件要求。不过，即使无法推行完整的敏捷开发，依旧可以在传统模式下，有针对性的应用敏捷开发的实践方法。

> 作者回复: 又是一篇高质量的分享👍
>
> 我没有什么好补充的，只是代表大家向你表示感谢🙏

67

黄双鹏

2019-03-05

如果是外包项目，作为项目的乙方，如果采用敏捷开发，最初的工作量就很难完整估计，不利于双方的合同签订。不知老师是否有好的建议，谢谢！

> 作者回复: 这个问题通常有两种解决方案：
> 1. 你按照瀑布模型的方式去估算工作量，然后签订合同。开发的时候你需求分析和架构设计还是用瀑布模型的方式，但是编码和测试用敏捷开发。这是一种不错的折中方案；
> 2. 你把所有需求拆分成用户故事，对用户故事进行打分（了解下计划扑克之类的打分方案），然后可以算出来一个总分数。另外按照你以前敏捷开发的经验，可以知道每个Sprint大概能完成多少分，这样你就能大致推算出来工期。

供参考！

27

Y024
2019-03-05

附一个补充材料，供参考：
天下武功，唯快不破—新时代敏捷项目管理之道
https://mp.weixin.qq.com/s/puMNz91hiQgio4wSCIrTgQ

作者回复: 感谢推荐，我看了一下，内容质量很高👍
大家可以参考看看

共 2 条评论 >

24

北有池鱼
2019-03-05

看完以后才发现我们组现在所谓的敏捷开发实际上是迭代开发的伪装！

作者回复: 其实迭代模型已经是很不错的模型了，如果适合你们组项目需要，那就是很好的，不用纠结。

16

alva_xu
2019-03-05

我们现在着手的一个项目，是一个软件框架建设项目，外包给供应商做的。在签合同时，基本需求已经梳理得差不多了。所以按理是可以采用瀑布式开发来进行的。但由于以下原因，所以我们结合了增量开发和Scrum项目管理的模式进行系统建设。
1，基本需求是可以分模块来实现的
2，我们这个项目所依赖的其他部门提供的基础平台也不是一次性可以交付我们使用的
3，我们的使用方(另外一个应用项目）对我们项目的时间要求很急，但可以接受我们分批次交付的模块。
基于以上原因，我们设立了几个大的增量阶段，每个增量阶段我们有分几个sprint来进行开发管理。到目前为止，进展还比较顺利。
但由于我们这个框架建设项目的外部干系人比较多，所以在协调上游平台和下游应用系统的时候，确实遇到了许多沟通方面的问题。由于其他项目没有进行看板管理，所以需要进行例会形

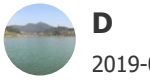
式的沟通来确保关键节点的功能实现。
所以，我认为，开发模式和项目管理模式不可以拘泥于一种形式，关键还是要看是否真正达到了整体的敏捷和精益。对于文中老师提及的scrum管理和极限开发，确实是小团队内部协同作战的比较好的实践。但对于多团队协同作战，就要考虑综合运用各种方法了。
另外，对于文中提及的站会形式，从"道"的角度来说，当然是可以视实际需求来确定是否要开，但往往一种文化的培养，需要有仪式感，需要不断锻炼。所以对于我们来说，我们还是坚持开Scrum中要求的四个重要会议的。

作者回复: 你这对软件工程中各个模型的应用可以说是非常经典的案例了，充分结合了各种模型的优缺点和适用场景，值得大家学习借鉴👍

软件工程知识点其实不算复杂，难的恰恰是如何灵活运用这些知识！

还有你说的"仪式感"也是个很好的点，这些会议看起来形式化，但确实能起到仪式感的效果。

12



D
2019-03-06

"用户故事"这个词好抽象，没太听明白。

老师，请教您一个问题，在敏捷开发过程中如何保证业务的传承？当有新同事加进来，如何让他快速的熟悉整个业务。

作者回复: 限于篇幅，用户故事确实没讲清楚，在文章中有推荐书籍可以作为参考，或者你也可以网上查询一些资料进一步了解，例如：http://www.woshipm.com/user-research/553736.html

这个是个好问题，也是个大问题！

新人的传承，通常我的经验是：
1. 团队要有自己的知识库或WIKI，常用的知识要花时间整理上去，这样新人来了可以自己查
2. 先给他简单的任务，再慢慢稍微复杂一点，给予必要的指导，做中学是最快速有效的
3. 遇到一些典型的问题可以通过结对编程的方式带着一起做

仅供参考

共 2 条评论>

10

不再回头
2019-03-05

敏捷开发，对整个项目的要求都一定的门槛。
团队成员：对需求分析的能力，需求到设计的能力，相对独立思考
自动化：持续集成的自动化部署，环境的搭建，达到持续交付的能力，自动化测试能力
测试验收：功能模块的验收，整体回归

其中立会、看板都是很好的方式方法，已经在执行中。
谢谢老师，请多指导！

作者回复: 总结的很好👍

敏捷开发有很多很好的实践，完全可以借鉴到非敏捷开发的项目中去，先从容易的开始，慢慢增加更多，最终也就"敏捷"起来了。

7

AICC
2019-03-05

这是一个自己每期跟进的专栏，第一次接触软件工程，几节内容下来，感觉对软件工程算是有了体感认知，也感觉到了专栏老师的用心，是目前学过的最走心的专栏了，留言的问题都会回复（其它专栏都是放出留言但少见回复，如果说描述性留言简单呈现能理解，但提问性留言还是只放出没回复，多少让人很郁闷，留言积极性也会下降，毕竟问题没搞明白），此外能从回复中看出来老师的鼓励和引导（这个可以看看老师的一些留言回复），同时也推荐老师的知乎专栏，宝玉的专栏，内容也很不错，像"记录下两个孩子在MineCraft里面还原公寓的经历"也能看出老师的鼓励和引导的教育方式，综上得出老师是一个leader，而不是一个管理者，哈哈哈

作者回复: 谢谢支持，说明我努力没白费呀：）

6

泡泡龙
2019-03-05

像这种情况下，有依赖交叉的用户故事应该怎么做，比如用户系统的数据库该由谁搭建。毕竟注册，登录，修改这些都可能基于一个数据表。表字段这些需要统一，不能一个程序员改一次字段名吧

作者回复: 敏捷开发中有一个迭代0，也就是第一个迭代，就是做这些准备工作、基础架构搭建的。

敏捷团队小，有个好处就在于遇到你说的这种情况，在做之前，大家都在一起开个小会一商量就可以定下来了

6

邢爱明
2019-03-07

对于企业管理的软件，核心需求涉及多个部门，需要反复沟通确认周期很长，这种情况下是否还适合使用用户故事的方式做需求分析呢?
另外，我按照瀑布开发模式的习惯分析，开发人员和po沟通需求后，如果没有文档作为输出物，在开发和测试的时候就没有标准，反而会造成工作返工。这是否意味着，团队成员需要高度的协同和配合?以完成任务为导向，而不是强调各自的分工。

作者回复: 好问题!

敏捷开发这种方式，需要客户紧密配合，也就是可以方便确认需求，否则还是少不了要写需求文档。

另外我在文章中描述用户故事，有些描写不清楚或者歧义的地方，其实用户故事还应该包括验收标准，这样可以解决你说的开发和测试没有标准的问题。

团队成员需要高度的协同和配合那是一定的，尤其是架构和需求两部分。需求简化后，就意味着开发过程中需要反复沟通确认；没有专门的设计阶段，也就意味着每个Sprint开始前，团队要商量有没有要设计或者修改架构的，有就需要有个简单可行的方案对架构进行修改。

如果各自分工，这样的目标就很难达到。

5

王二宝
2019-03-05

每篇都有很大收获，不得不说这是让我受益最大的一个专栏。谢谢宝玉老师
道是价值观，是一种思想，术是一种方法。
段永平说：术是可以学习和模仿的，而道却是需要悟的。

当你理解了道，术是可以万变的。
我订阅了7，8个专栏，老师是第一个把道和术分开讨论的。打心底里佩服老师，谢谢

作者回复: 谢谢支持!

软件工程是一门相当有用的学科，我主要是领着你去学，学了你就会发现真的有价值👍

另外我在学习攻略那一篇还提到一个观点就是教中学，就是你可以先“借道”，把别人总结得道先借过来，再结合一些术和器，去尝试总结或者讲给其他人，这过程中你也一样会有很多感悟和不一样的理解。

5

Felix
2019-03-05

目前在我们团队完全敏捷开发很难，罗马不是一天建成的，我同意老师所说的话，可以把一些好的敏捷实践先用起来

作者回复: 👍
是的，从瀑布到敏捷是个很大的转变，不要急于求成。

5

小老鼠
2019-08-20

1，老师讲的那个房子敢住吗？2、需求十分稳定的项目适合用敏捷吗？3、敏捷中又是单元测试又是自动化测试（功能、性能），工作量是否加大，令不会影响产品发布。4，现在企业人员流动比较厉害，文档细到什么程度比较合适？

作者回复: 1. 如果是按照建筑工程标准验收通过了，当然没问题；

2. 需求稳定的项目一样适用于敏捷开发。尤其是Scrum这样固定迭代周期的敏捷开放方式，可以倒逼你每次迭代先选择优先级高的需求。从已经确定的项目需求，选择最核心的需求，先进行迭代开发，这样你很快就可以看到一个可以运行的版本，然后每次迭代继续增加新的功能，持续发布。

3. 磨刀不误砍柴工，前期投入了功夫在自动化测试，后期会有质量上的回报，敏捷开发会优先砍需求，所以产品可以发布，最坏结果是初期发布时，需求是不完整的，一些优先级不高的需求初期并没有加进去。

4. 项目文档有两个主要目的：1) 帮助写的人理清楚思路; 2) 用来交流沟通。

所以文档的关键不是细，而是达到文档的写作目的。至于粒度，可以站在文档读者的角度，结合日常开发维护场景，去思考这个文档，对于一个需要查阅文档的人来说：文档能给他什么？上面的信息是否有价值？能否达到交流沟通的目的？

举例来说对于一个需求文档，主要的用户场景是否覆盖？输入输出条件是否清晰？交互设计是否完整？异常情况如何处理？

举例来说一个项目交接文档，是否有整体概要的架构图，让人能从整体了解这个项目；是否有各个模块的相关说明，能大致了解各个模块的作用；是否有开发的说明，可以进行开发调试；是否有服务器部署的说明，可以按照文档对服务器进行部署更新……

共 2 条评论 >

4

桃子-夏勇杰

2019-08-19

敏捷与需求分析

应用敏捷之后，并不是说不用做需求分析了，其实敏捷更加鼓励深度的需求分析。传统软件开发经过需求分析之后，就不在接受新的需求了。而敏捷软件开发对需求变更抱持了一种开放的态度，这样更有利于帮助客户实现真正的价值。但是，这个做法并不是我们拖延需求分析到后面开发过程的借口，在敏捷软件开发的前期，对于客户的核心价值，我们要做可能比传统软件开发还要深入的分析。传统软件开发的需求分析范围太广，花费了很多的时间，但是，并不一定能把核心需求分析做到位。

作者回复: 👍敏捷开发这样一个Sprint一个Sprint的迭代，天然的可以将优先级高的需求放在前面。

4

阿神

2019-03-06

敏捷开发里开发也要写集成测试用例吗，那么测试人员主要做手工测试？

作者回复: 对，开发不仅要写单元测试，还要写集成测试。但开发都是用模拟数据，假的API。而测试的自动化测试会用真实的数据，调用真实的API，而且也要做一部分手动测试。

至于比例多少，还得看项目特点

4

williamcai

2019-03-05

能够很快有实现的的功能，及时看到效果，给人成就感，小步慢走，一步一步重构优化，直至完成目标。坏处是走着走着，发现当前的架构不适合待开发的需求，这就需要重构，可能还要数据迁移，增加额外的工作量

作者回复: 👍总结的很好。没有完美的方案，只有适合的方案。

很多时候，及时看到阶段性成果，有成就感还是很重要的，经历过那种很长时间都看不到结果的项目，都不知道什么时候能看到头，每个人压力都很大的，相对来说要做点重构和数据迁移真不算什么了😄

4

hello zero

2019-03-05

老师请教下，如果合同金额一开始就是根据商务阶段了解的情况评估的工作量而确定的，那么在合同执行过程中，如果按敏捷开发的思路，客户不断改需求我们不断地响应，然后工作量甚至已经超过了原先合同的金额，这个时候要如何处理？

作者回复: 这是个好问题，我对这个问题上没有什么经验，但我可以试着帮你分析一下。

你的合同是按照当时的需求签订的，如果后期客户变更需求或者增加新需求，那相当于需要重新签订变更这部分的补充合同。

应用敏捷开发的时候，你也可以让产品经理或者项目经理充当客户的角色，这样他们会更偏重产品需求的解读，而不是重新提出新的需求：）

还有一点，合同执行的时候，这时候你不需要太过于纠结是不是用敏捷还是迭代还是瀑布，而是哪一种开发模式，可以让你高质量高效率的完成，那就是最好的最适合你的开发模式。

4

乐爽
2019-06-13

详细的需求分析是放在迭代内进行的，但此时的需求是一个很小的点，所以不会占据整个迭代太多的时间，是吗？如果在迭代内发现需求方案不合理，放入到下一个迭代，这是否合理呢？

作者回复: 是的，因为一个迭代内的需求不多，所以需求分析相对时间较短。

如果一个需求在一个迭代内做不完，可以延到下一个迭代。

如果一个需求不合理，那么需要重新讨论，讨论清楚了再决定是放当前迭代还是后续迭代。


3

梁中华
2019-03-20

我看你在比较传统瀑布式和敏捷开发的时候，对于架构设计这块基本省略了，现在互联网开发，很多项目都面临大并发，大吞吐量的要求，如果省略了这个步骤，一般开发人员估摸着就开始开发了，这块的技术债会越来越多，甚至一上线就要重构。我感觉2001年的时候，大部分还都是桌面类的软件，互联网项目的并发量也还没这么大，或者在非功能性需求方面要求并不高，文中敏捷开发适用于大部分日常项目。但是上一定规模的项目，设计评审，特别是关键设计的评审这个步骤不能弱化，而是要强化。

作者回复: 谢谢补充，架构设计这块文章中确实没细讲。

也认同你的观点，架构设计是很重要的事。

后面也会有章节继续介绍架构设计。


3

舒偌一
2019-03-08

我们是瀑布+敏捷，整个项目过程采用瀑布管理，中间各环节具体实施采用敏捷。
如果要实施敏捷，首先得统一认识，达成一致；其次是对敏捷有基本的认识；其次是团队人员能力要上去，特别是责任心，其次是客户的深度参与。实施起来好难呢，心累！！！

作者回复: 前期实施确实是有难度，但慢慢运转顺利了就好多了。

很多问题也可以寻找一些折中方案的，例如客户如果不能深度参与，那么你可以考虑由产品经理代替客户的角色。能力的话，依赖于招聘和培训；责任心这些，可以靠流程制服辅助。

如果认定大方向没有做，剩下的就是坚持和不断改进了。 


3