**敏捷开发**

## 一、敏捷开发的起源

**1 在90年代末期，传统软件开发的方式因**为其繁杂的过程，以及对文档的过于严格的要求，造成了很大程度上的效率下降，也就是人们所说的“重型化危机”。因为这一原因，人们开始反思传统方法的利弊，并对其弊端进行了改进，提出了敏捷方法。

**2 2001年2月，由Martin Fowler，Jim Highsmith等**17位软件开发专家起草的敏捷宣言发表，敏捷联盟成立。敏捷开发作为一种新的方法正式诞生。敏捷宣言中所表述的价值观分为四个方面：

（1）个体和互动 高于 流程和工具

（2）工作的软件 高于 详尽的文档

（3）客户合作 高于 合同谈判

（4）响应变化 高于 遵循计划

**3 同时敏捷宣言还包括12条原则**。这十二条原则是以上四条主要的价值观在实际工作中的体现。总体来说，**敏捷开发作为一种新的软件工程方法，与传统方法相比更加注重人的因素**。不再把开发者当作一个物化的，投入多少时间可以完成相应数量代码的代码开发机器；

**而是注重开发者之间的互动以及开发者和用户之间的互动，同时因为增加了交流和协作使得开发的项目更加灵活和易于修改。**

**4 敏捷开发的主要特点**

1. **敏捷开发的过程有着更强的适应性而不是预设性，**从敏捷宣言的第四条响应变化高于预设计划便可以看出来。因为软件开发过程的本身的不可预见性，很多用户在项目开始时不可能对于这个项目有着一个完整而明确的预期。很多对软件的预期都在后期的修改和完善过程中产生。因此高适应性显然更加符合软件工程开发的实际。而敏捷开发实现其适应性的方式主要在于，

第一，缩短把项目提交给用户的周期；

第二，增加用户，业务人员，开发人员这三者之间的交流；

第三，通过减少重构的成本以增加软件的适应性。

**（2）敏捷开发的过程中，更加的注重人的因素。**在传统软件工程中，个人的因素很少的被考虑到分工中，每个个体都是只是整个代码开发机器的一个小小的螺丝钉，个人的意志和创造力很大程度上的被抹去为了更好的为集体服务。**而在敏捷开发过程中，每个个人的潜力被充分的考虑**，应用什么技术很大程度上直接由在第一线开发的技术人员决定；每个人的特点和创造力都可以充分地发挥，这样开发出来的软件更加的具有生命力，因为他融入了开发者的心血和创意，开发者不再是进行机械的乏味的堆砌，而是创造属于自己的艺术品，这样的条件下产生的代码必然在质量上更占优势。

**（3）在敏捷开发的过程中，整个项目是测试驱动的而不是文档驱动的。**不仅每个模块有着自己的相应的测试单元，开发人员在开发自己的模块的过程中必须保证自己所开发的模块可以通过这一单元的测试，并且集成测试贯穿了整个开发过程的始终。集成测试每天会进行十几次甚至几十次，而不是像传统方法一样只有当各个模块的编码都结束了之后再进行联合调试。这样，在软件开发的进程中每一点改动所引起的问题都容嘉容易暴露出来，使得更加容易在错误刚刚产生的时候发现问题从而解决问题。这样就避免了在最后整个系统完成时错误隐藏的太深给调试造成极大的困难。

**5 敏捷过程模型的一个实例：极限编程**

极限编程过程分为策划、设计、编码和测试四个阶段。

**（1）策划阶段**

首先在策划阶段，用户和开发这进行交流，开发者总结出一系列“用户故事”，描述软件某一部分功能。之后客户对这些功能进行优先级排序，xp团队评估每一个故事的成本。之后客户和xp团队共同决定在开发的下一个版本中将会新增哪些功能。而在版本不断的迭代的过程中，会进行很多次这样的策划过程，每一次客户都可以根据已有的功能来决定是否要新增一些功能，以及要新增哪些功能。

**（2）设计阶段**

在设计阶段，开发人员会根据用户故事，提出这些用户故事的实现方案。设计的过程中主要遵循简洁的原则，**也就是尽量使用简介的表述而不是复杂的表述**。而设计的另一个方面则是重构，重构是一种通过不改变代码的外部功能的情况下改变软件模块的内部结构从而优化软件系统的功能的过程。这是一种改进代码的设计。在设计的这两个层面中，我们可以看到在xp开发过程中，设计和开发是同步进行的。我们在不断实现开始设计的过程中，同时要对到吗进行优化也就是重新设计。**这样，大大的增强了整个软件开发的适应性，而不是始终刻板的实现最开始的第一版设计**。

**（3）编码阶段**

xp开发的第一件任务不是直接对初步的设计和用户故事进行编码，而是针对这些设计全力开发单元测试。完成了单元测试也就确定了开发者要实现的所有功能。这样开发者就只需要全力通过单元测试，而不必在实现什么功能上再浪费不必要的时间和精力。这正体现了敏捷开发的以测试驱动的特点。

而在敏捷开发中，很重要的一个提高效率的方式就是结对编程。在结对编程的过程中，两个开发者共用一台电脑，并各有分工。其中一个人进行实际的编码实现，另一个人在旁边考虑代码在宏观上该如何实现，比如针对什么功能应该使用什么样的算法。这样，在编码者工作遇到问题时，两个人交换位置。这时在旁边思考的人更有可能可以解决这一问题。事实上，结对编程的形式不必拘泥于什么规矩。关键在于，两个人共同开发的过程中，两个人的交流可以使得大部分的问题可以在第一时间解决。并且，因为两个人中只有一个人在进行编程这一项比较疲惫的工作，另一个人较为轻松，这样可以保证开发效率一直保持在一个比较高的状态。

**（4）测试阶段**

每一个模块都通过自己的单元测试之后，开发者会将所有的模块集成到一起进行测试。这样可以及时发现每一模块在最近一次改动之中出现的问题同时避免一些兼容性问题。每一次改动一点小问题要比等到最后一次集中修改所有问题要容易得多。

**6 敏捷开发生态系统**

敏捷开发模型在实际中有着很多表现形式。极限过程开发（xp）时其中的最为广泛应用的一种。还有很多其他的，比如：**自适应软件开发、Scrum、动态系统开发、Crystal、特征驱动开发、精益软件开发、敏捷建模、敏捷统一过程**等。这里只举两个例子介绍一下其主要的特点。自适应软件开发主要从整体上强调软件项目团队具有自我组织的动态性、人与人之间的协作、个人以及团队的学习，从而使团队更有可能取得成功。Scrum开发方法，这个开发方法最大的特征就是每日例会。在每日例会中，每个人交流自己昨天干了什么，今天将要干什么，以及自己在工作中遇到了哪些问题。这样大大地加强了团队成员之间的交流。

我们可以看到，很多人都投入到了敏捷开发的研究和使用中。敏捷开发确实有着非常强大的生命力。

**7 敏捷开发与传统开发方法的比较**

**（1）优势**

敏捷开发的**高适应性**，**以人为本**的特性，和轻量型的开发方法即**以测试为驱动**取代了以文档为驱动，这三个主要的特点，也就是敏捷开发相对与传统开发方式的主要有点。因为他更加的灵活并且更加充分的利用了每个开发者的优势，调动了每个人的工作热情。

**（2）劣势**

与传统开发方式相比，敏捷开发的最主要的劣势在于敏捷开发欢迎新的需求，而在每次新的需求产生时都可能引起整个系统的大幅度的修改。因为开发者在开发上一个版本的时候，完全没有考虑以后的优化将要如何进行。这样的开发方式实际的软件开发过程中，并不一定总是有效的。

而另一个方面，敏捷开发因为缺乏很多在敏捷开发中被认为“不重要”的文档，这样在一个大型项目比如一个操作系统开发的时候，由于其项目周期很长，所以很难保证开发的人员不更换，而没有文档就会造成在交接的过程中出现很大的困难。

**8 首先讲为什么需要敏捷开发。在几万年以前，软件项目的开发都是以年来计算的，这代表什么意思呢 ？**

需求设计了半年多，方案设计做了半年多，开发了三年多，测试了半年多，修改Bug用了半年多。总计花了很长很长的时间，然后上线后发现有很多需求已经不存在了，同时又出现了很多新的需求。怎么办？继续改。这一改又是半年多的时间过去了。马丹用户的需求还再改，怎么办？这是困扰软件开发项目的最大的问题，越大的项目，参与的人越多，风险越大。

**9 传统模式遇到的问题**

**1.需求总是在变动，反复变动，无限拖延。**

2.开发工程师做出来的项目，bug不但多，而且经常改不好。常常是改了一个Bug，出现另一个Bug，好不容易把一个Bug改好了，过了没多久又重现了。原本好好的功能，反而会因为改Bug导致出现的问题更多。

3.做出来的东西完全不是产品经理想要的样子，沟通完之后才发现开发工程师的理解和产品经理的理解是完全不一样的。

4.项目延期不是最坏的结果，最坏的结果是还从不知道项目倒底会延期多少，根本没办法去衡量工作量，团队的成员都在加班加点，然而完全看不出来问题出在什么地方。

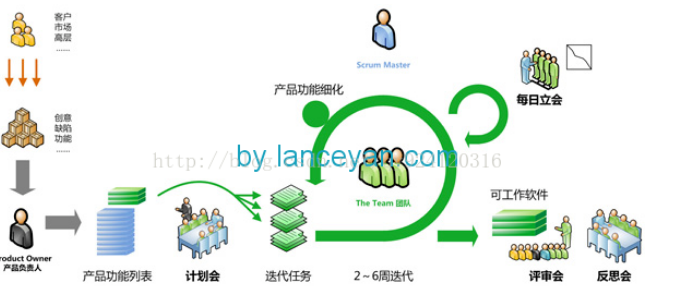
5.开发文档，产品文档，接口文档，测试报告和真实的代码从没有完美契合过。产品经理设计出来的原型和UI设计出来的页面和程序员开发出来的代码完全是一种不同的体系，三位一体的故事从没有真正发生过。代码的实现和接口文档根本不一致，最后索性干脆不看接口文档，完全口头交流。出错的时候各种撕逼扯皮，谁也分不清倒底谁错了。

6.Team的战斗力和凝聚力不强，经常是对着干，对分配的任务总是各种报怨，出现问题之后第一反应是这个不关我的事，不是我的问题，是后端前端设计QAPM的问题。如果你遇到了这种情况，或者说你不甘于这种现状，那么恭喜你，你可以真的需要敏捷开发流程了。

**10 敏捷开发包括了哪些内容？**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **编号** | **活动名称** | **活动内容** | **工期** | **参与人** |
| 1 | 需求分析 | 产品给出版本链接总入口，研发人员和测试人员第一时间熟悉下  个迭代的需求，备注中记录疑问 ，对需求有第一印象的理解。 | 1~2天不等，非全职 | 研发，测试 |
| 2 | 需求评审 | 研发同事带着疑问和产品运营，测试人员三方评审需求，基于目  前的业务体系分析新需求的可行性和实现方案，解决 疑难需求，澄清有歧义的语句，中间环节测试和研发可随时提问。 | 2~4小时不等 | 研发，测试，产品，项目经理 |
| 3 | 需求串讲 | 研发人员经过1~2天对wiki上的需求进行详细分析和把控，把自己对需  求的理解，做成的效果讲给运营确认，如果有理解不一致的地方当场  和测试，产品讨论，把理解思路达成一致。 | 2~4小时不等 | 研发，测试，产品 |
| 4 | 概要设计 | 研发同事把概念模型转换成逻辑模型和物理模型，设计好实现思路，结  合相关的业务场景用最好的设计模式，包含概要设计，接口设计，数据  库设计，业务流程设计等工作。针对同一个功能每人可发表不同的意见  ，最后统一设计思路。 | 1~2天不等，非全职 | 研发 |
| 5 | 设计评审 | 经过1~2天的设计，对设计文档进行评审，找出设计方案可能错误的地方及  时修正，找出疏漏的地方及时补上。同时共同制定相关的编码规范，会议完  成后在wiki写下评审记录。 | 2~4小时不等 | 研发 |
| 6 | 代码开发 | 研发同事把设计转换成代码实现，在研发数据库执行sql脚本，研发对应的  feature分支生成CRUD代码，上传业务代码，绘制流程图，期间有任何疑  问及时抛出来，间歇性跟进研发进度，设定版本里程碑，按照里程碑识别  风险，如果有技术风险第一时间给出方案，如果有进度风险需要加班补偿。 | 根据工作量资源情况， 2~4周不等 | 研发 |
| 7 | 代码评审 | 一般在转测前几天进行代码review,针对代码的基本编码规范和健壮性,稳定性  ,性能性,安全性,可扩展,可复用,可维护,可配置等多维度对代码进行走查，如果  不符合基本规范及时修正，对模块之间的高内聚低耦合需要重视，避免大方向  走偏，针对同一段代码每人可发表不同的意见，最后统一思路。结束后在wiki  记录结果。检查和督促研发人员及时修正。 | 2~4小时不等 | 研发 |
| 8 | 联调自测 | 研发后期对评审的问题进行修复，对功能进行收尾，对复杂的逻辑需要提供  单元测试用例并且有测试结果，涉及多系统配合统一在dev环境进行联调，对  联调的问题及时修复，期间会对测试人员提供的 用例进行评审。 | 2~4天不等 | 研发 |
| 9 | 测试验收 | 首先提前一天从git获取版本对应的脚本变更在测试环境统一执行，部署测试包，  研发同事有半天时间在测试环境测试自己相关的功能和主流程是否阻塞，如果  有阻塞及时修复，第二天开放给测试人员使用，每天给出测试进度百分比，及  时识别风险，晚上研发同事需要相应加班支持，测试阶段后期产品同事加入验  收环节，及时找出问题进行修复。 | 根据工作量资源情况， 1~2周不等 | 研发，测试，产品 |
| 10 | 生产上线 | 测试人员把用例跑完，并且严重问题全部修复验证的情况下 提供提测&上线文档  ，并且对文档进行评审，代码合并到master和dev，检查 erp.properties配置项，  执行脚本，配合运维同事进行版本升级，升级后部署流程图，补充菜单角色等数  据，上线后进行。 | 当天 | 研发，测试，产品 |
| 11 | 上线支持 | 发版之后，测试人员需要建立测试商户，进行主流程的回归测试，有问题研发及  时修正，产品同事对版本功能进行验收，对修复后的问题走bugfix部署。 | 1~2天 | 研发，测试，产品 |
| 12 | 问题分析 | 针对上个版本转测阶段造成的问题进行原因归类和分析，总结出措施，为了更好  的控制 版本质量和尽可能减少不必要的问题。问题原因不排除代码缺陷，设计缺  陷，历史问题，需求优化，小程序，操作问题，数据问题，环境问题，其他引发  ，修改引发等。 | 2~4小时不等 | 研发，测试 |
| 13 | 需求变更 | 产品同事提供jira跟踪单，指明wiki需求地址，研发根据需求涉及的功能和场景评  估工作量，识别给出对本版本的转测时间变动，通知项目经理，产品，测试，给  出新的转测时间，测试同事根据评估的测试时间给出新的上线时间。以此类推。 | 2~4小时不等 | 研发，测试，产品，项目经理 |

## 敏捷团队



1. **产品和开发必须是一个Team**

大家只是分工不同，角色不同，并不是两个对立的团队。如果你的公司是把产品和开发分成两个部门，那么恭喜你，产品和开发之间的纠纷一定无限多。在所有我带的Team中，自始至终强调的理念就是：出了问题，别跟我说这是产品设计出来，这是开发团队实现不了的。我只知道这是你们一个开发小组所有人的责任，这个后果是所有的人都需要承担的。如果我们认真的区分这是什么问题，那么也只是为了避免下次出现同样的情况，用户只会知道是一个公司出了一款垃圾产品，没有人关心到底是产品还是开发的锅。这是做敏捷开发的大前提。

或者不仅仅是产品和开发，责任共担，One Team这个理念是贯穿始终的。这并不是说，大锅饭，而是说，面对不好的结果，所有Team的人都必须共同承担。出现问题的原因仅仅是为了追溯和重现当时的场景，

以避免后续会出现同样的情况。产品和开发必须是一个Team还体现在需求分期上。这一点在讲到需求分期的流程的时候，会提高的。实际上，需求分期如果没做好，敏捷开发只能流于形式。需求分期怎么做，这是MVP的事情，另一个话题。简单来说，每一期都要有一个提前的预测，这一期里要做的所有的功能都只为了检测自己的预测是否正确。并根据结果去不断的调整开发规划。

**2.职责明确，每个人要负责的事情必须清晰无误**，谁该做哪些事情，必须要提前讲清楚。开发团队的推荐角色应该是这样的。

**PM 1个**

**UI 1个**

**CSS/js 1~2个**

**Java 2~4个**

**Android 1~2个**

**IOS 1~2个**

**QA 1个**

**@PM ：** PM的职责并不是画原型，而是去分产品的分期，确定产品要做的功能和优先级。

对于产品来说，最大的职责并不是将原型画出来，而是要证明自己要做的功能是合理的。

如果你证明不了自己要做的功能是合理的，是值的尝试的，就是产品经理的失职。

**做为PM，一定要有这种风险的意识，要知道自己身上担负的责任**，PM花了两周时间设计的原型，8人的开发团队要折腾近三周左右的时间。原型和产品文档都是辅助的东西，我甚至不推荐产品经理去做原型设计，只拆分Story。原型设计交给传统的UI更合适。然而在真实实施的过程中，因为很少有UI具备原型的设计能力，所以实施起来会有一些难度。这个不算特别重要，慢慢培养。

**@小组Leader：**需求评审会的成员应该包括PM组的Leader，前端组的leader,后端组的leader,测试组的Leader，或者是其他公司的中层骨干。这应该是一个公司所有应该为这个项目负责的人的评审会，在评审会上的结论，就应该被坚定的执行下去了。不参与评审会的人，不应该再对需求指手画脚。 需求评审会的目标就是确定原封不动的需求，所以在这里要格外的注意，PM拿出来的方案设计，一定是完整的，而且必须评细节。如果说，一个公司的中层骨干经过需求评审会议，仍然需求乱成一比，

**@开发组成员**：项目进入真正的开发阶段后，开发组的成员就应该是主动去控制项目的进度，和风险，以及主动去测试项目中存在的问题，在这个阶段，除了一些需求不明，或者是发生变动的情况出现，不应该去打扰产品经理。不要让产品经理做开发团队的保姆。开发组的成员的目标就是做好项目的进度控制，有风险就及时反馈给Leader，确保自己理解的需求是明确无误的，确保自己的测试是完整和严谨的，确认自己写出来的代码是可以维护的。一定要理解清楚，一旦PM通过Story讲解，将需求交付给开发组成员，那么开发组成员就应该主动而独立的为这件事情负责。当项目完工以后，开发组成员应该交叉去做CodeReview，并且出性能测试报告，以及组织Demo。

**@测试组成员：**测试级成员的职责不是做功能性的测试，也不是做性能测试。而是应该做边界测试和回归测试。功能性的测试主要应该由开发组成员完成，除了一些特别麻烦的，需要各种极端条件才能复现的，正常的操作过程中出现的问题，都应该是有开发组承担。性能测试同样是开发组人员自行完成，各小组Leader只需要知道一件事情，测试报告是否能够通过。所以测试组的主要做的就是准确的记录，以及bug的统计。也不应该去天催促开发组的成员去改Bug。只需要去反馈给开发组的Leader就好了。整个CTO或者是技术总监应该以此为标准去衡量每个小组Leader的绩效。回归测试是需要做的，但是也不是完全必须要做。如果能够积累足够多的自动化测试用例，就去正常使用它，如果不能，就尽可能少的减少回归测试。这需要跟开发人员沟通的比较清楚，他们往往更清楚，什么地方容易出问题

。接受线上的反馈并且记录也应该是QA的职责，如果Team足够细，可能会有运营或者是客服统一对外收集，然后交给QA，QA再负责录入Bug系统中。基本的敏捷开发流程中的角色的职责大致就是这样的了。