【1】

<https://www.cnblogs.com/linbo3168/p/11725048.html>

Agile——敏捷开发，作为CMM神话崩溃后被引入的一套新的软件开发模式，这几年来被广泛引起关注，并被寄予厚望。敏捷开发在其它业界的应用是否理想不得而知，但下面总结了我所在公司的敏捷开发试验，希望能够达到管中窥豹的目的。

**敏捷开发宣言——**

* 个体和交互 胜过 过程和工具
* 能够工作的软件 胜过 面面俱到的文档
* 客户合作 胜过 合同谈判
* 响应变化 胜过 遵循计划

尽管右项也有价值，可是我们觉得左项具有更大的价值。

以上的宣言比較抽象，基于该理念，下面是ThoughtsWork咨询公司的推崇的n个敏捷开发实践：

**Iteration**

* 迭代开发。能够工作的软件胜过面面俱到的文档。因此，敏捷开发提倡将一个完整的软件版本号划分为多个迭代，每一个迭代实现不同的特性。重大的、优先级高的特性优先实现，风险高的特性优先实现。在项目的早期就将软件的原型开发出来，并基于这个原型在兴许的迭代不断晚上。迭代开发的优点是：尽早编码，尽早暴露项目的技术风险。尽早使客户见到可执行的软件，并提出优化意见。能够分阶段提早向不同的客户交付可用的版本号。

**IterationPlanningMeeting**

* 迭代计划会议。每一个迭代启动时，召集整个开发团队，召开迭代计划会议，全部的团队成员畅所欲言，明白迭代的开发任务，解答疑惑。

**Story Card/Story Wall/Feature List**

* 在每一个迭代中，架构师负责将全部的特性分解成多个Story Card。每一个Story能够视为一个独立的特性。每一个Story应该能够在最多1个星期内完毕开发，交付提前測试（Pre-Test）。当一个迭代中的全部Story开发完毕以后，測试组再进行完整的測试。在整个測试过程中（pre-test，test），基于Daily build，測试组永远都是每天从配置库上取下最新编译的版本号进行測试，开发者也随时改动測试人员提交的问题单，并合入配置库。敏捷开发的一个特点是开放式办公，充分沟通，包含測试人员也和开发者一起办公。基于Story Card的开发方式，团队会在开放式办公区域放置一块白板，上面粘贴着全部的Story Card，按当前的开发状态贴在4个区域中，各自是：未开发，开发中，预測试中，測试中。Story Card的开发者和測试人员依据开发进度在Story Wall上移动Story Card，更新Story Card的状态。这样的方式能够对项目开发进度有一个非常直观的了解。

在开发者開始开发一个Story时，ta须要找来相应的測试人员解说Story功能，以便測试人员有一致的理解，同一时候開始自己主动化系统測试脚本的开发。

**Standup Meeting**

* 站立会议。每天早上，全部的团队成员围在Story Wall周围，开一个高效率的会议，通常不超过15分钟，汇报开发进展，提出问题，但不浪费全部人的时间立马解决这个问题，而是会后个别沟通解决。

**Pair Programming**

* 结对编程是指两个开发者结对编码。结对编程的优点是：经过两个人讨论后编写的代码比一个人独立完毕会更加的完好，一些大的方向不至于出现偏差，一些细节也能够被充分考虑到。一个有经验的开发者和一个新手结对编程，能够促进新手的成长，保证软件开发的质量。

**CI/Daily Build**

* 持续集成和每日构建能力是否足够强大是迭代开发是否成功的一个重要基础。基于每日构建。开发者每天将编写/改动的代码及时的更新到配置库中，自己主动化编译程序每天至少一次自己主动从配置库上取下代码，执行自己主动化代码静态检查（如PCLint），单元測试，编译版本号，安装，系统測试，动态检查（如Purify）。以上这些自己主动化任务执行完毕后，会输出报告，自己主动发送邮件给团队成员。假设当中存在着不论什么的问题，相关责任人应该及时的改动。能够看到，整个开发组频繁的更新代码，出现一些问题不可避免。通过測试部又在不停地基于最新的代码进行測试。新增的问题能否够被及时发现并消灭掉，取决于自己主动化单元測试和系统測试能力是否足够强大，特别是自己主动化系统測试能力。假设自己主动化測试仅仅能验证最简单的操作，则新合入代码的隐患将非常难被发现，并遗留到项目后期，形成大的风险。而实际上，提升自己主动化測试的覆盖率是最困难的。

**Retrospect**

* 总结和反思。每一个迭代结束以后，项目组成员召开总结会议，总结好的实践和教训，并落实到兴许的开发中。

**ShowCase**

* 演示。每一个Story开发完毕以后，开发者叫上測试人员，演示软件功能，以便測试人员充分理解软件功能。

**Refactoring**

* 重构。由于迭代开发模式在项目早期就开发出可执行的软件原型，一開始开发出来的代码和架构不可能是最优的、面面俱到的，因此在兴许的Story开发中，须要对代码和架构进行持续的重构。迭代开发对架构师要求非常高。由于架构师要将一个完整的版本号拆分成多个迭代，每一个跌倒由拆分成非常多Story，从架构的角度看，这些Story必须在是有非常强的继承性，是能够不断叠加的，不至于兴许开发的Story全然推翻了早期开发的代码和架构，同一时候也不可避免的须要对代码进行不断完好，不断重构。

**TDD**

* 測试驱动开发。正如上面讲的，迭代开发的特点是频繁合入代码，频繁公布版本号。測试驱动开发是保证合入代码正常执行且不会在后期被破坏的重要手段。这里的測试主要指单元測试。

**敏捷方法反思：**

自己參与的敏捷开发项目总的来说不是非常成功，这可能也是业界遇到的通病：

1. 对于全新的软件，在项目早期測试人员就參与并实现自己主动化測试脚本，但实际上软件的界面等非常不稳定，导致測试人员返工的工作量非常大。
2. 对于全新的软件，资料人员过早參与，后期返工工作量大，原因同第一点。
3. 自己主动化系统測试工作量大，測试人员投入大量的精力在使測试自己主动化起来，而没有足够的精力放在真正的測试软件的功能是否正常。即便是这样，自己主动化系统測试脚本也多流于形式，測不出深层次的问题。
4. 代码动态检查工具执行不理想，流于形式。没有人对Purify有深刻的理解和应用经验，报告中查出来非常多告警，但不知怎样消除。
5. 由于高速搭建原型，没有在架构上进行严谨的设计，导致后期一直堆砌代码。
6. 异地开发模式下无法实现高速构建、高速交付，团队普遍感觉非常疲惫。
7. 敏捷开发不提倡加班，但实际上无论是CMM还是Agile哪一种开发模式跟是否加班都没有必定联系

【2】

<https://baike.baidu.com/item/%E6%95%8F%E6%8D%B7%E5%BC%80%E5%8F%91/5618867?fr=aladdin>

## 敏捷开发团队原则

[编辑](javascript:;)

**最大的分歧**

最大的分歧在于开发人员和测试人员之间。作为敏捷团队的成员，测试人员被期望能编写一点代码，同时开发人员可以做一些测试。各自的强项还是很重要：新的角色要求每个成员成为大家所谓的“通才”。测试人员大多数时间作测试，开发人员大都编写代码，但所有人都分享他们的工作，而且有能力承担他们面前的任务。

**发现中立点**

团队决定作为一个团队需要做什么，如何最好地分配工作。第一步是让团队成员说说他们自己的技能集、优点和缺点。但却不希望他们根据以前角色（如，软件测试员或开发员）来定义自己。所以找到一个中立点，她列出了小型离线会议，和每周工作之外的小时集体活动所需的事项。这样，该团队去当地的农场采摘蓝莓。他们一起上瑜珈课。他们集体在厨房里烤燕麦棒，做果沙。[5]

**正确执行应用程序**

团队找到了让自此感到舒服的新水平。整个项目的工作流程顺利进行，只做一个待办的事情，而不是四个。

## 敏捷开发的原则

[编辑](javascript:;)

**1. 快速迭代**

相对那种半年一次的大版本发布来说，小版本的需求、开发和测试更加简单快速。一些公司，一年发布仅2~3个版本，发布流程缓慢，它们仍采用瀑布开发模式，更严重的是对敏捷开发模式存在误解。

**2. 让测试人员和开发者参与需求讨论**

需求讨论以研讨组的形式展开最有效率。研讨组，需要包括测试人员和开发者，这样可以更加轻松定义可测试的需求，将需求分组并确定优先级。 同时，该种方式也可以充分利用团队成员间的互补特性。如此确定的需求往往比开需求讨论大会的形式效率更高，大家更活跃，参与感更强。

**3. 编写可测试的需求文档**

开始就要用“用户故事”（User Story）的方法来编写需求文档。这种方法，可以让我们将注意力放在需求上，而不是解决方法和实施技术上。过早的提及技术实施方案，会降低对需求的注意力。

**4. 多沟通，尽量减少文档**

任何项目中，沟通都是一个常见的问题。好的沟通，是敏捷开发的先决条件。在圈子里面混得越久，越会强调良好高效的沟通的重要性。

团队要确保日常的交流，面对面沟通比邮件强得多。

**5. 做好产品原型**

建议使用草图和模型来阐明用户界面。并不是所有人都可以理解一份复杂的文档，但人人都会看图。

**6. 及早考虑测试**

及早地考虑测试在敏捷开发中很重要。传统的软件开发，测试用例很晚才开始写，这导致过晚发现需求中存在的问题，使得改进成本过高。较早地开始编写测试用例，当需求完成时，可以接受的测试用例也基本一块完成了