**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。敏捷开发在其他业界的应用是否理想不得而知，但以下总结了我所在公司的敏捷开发试验，希望可以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  
可以工作的软件 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  
虽然右项也有价值，但是我们认为左项具有更大的价值。  
  
以上的宣言比较抽象，基于该理念，以下是ThoughtsWork咨询公司的推崇的n个敏捷开发实践：  
**Iteration**  
迭代开发。可以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划分为多个迭代，每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，并基于这个原型在后续的迭代不断晚上。迭代开发的好处是：尽早编码，尽早暴露项目的技术风险。尽早使客户见到可运行的软件，并提出优化意见。可以分阶段提早向不同的客户交付可用的版本。  
**IterationPlanningMeeting**  
迭代计划会议。每个迭代启动时，召集整个开发团队，召开迭代计划会议，所有的团队成员畅所欲言，明确迭代的开发任务，解答疑惑。  
**Story Card/Story Wall/Feature List**  
在每个迭代中，**[架构](http://lib.csdn.net/base/16" \o "大型网站架构知识库" \t "_blank)**师负责将所有的特性分解成多个Story Card。每个Story可以视为一个独立的特性。每个Story应该可以在最多1个星期内完成开发，交付提前测试（Pre-Test）。当一个迭代中的所有Story开发完毕以后，测试组再进行完整的测试。在整个测试过程中（pre-test，test），基于Daily build，测试组永远都是每天从配置库上取下最新编译的版本进行测试，开发人员也随时修改测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，包括测试人员也和开发人员一起办公。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，上面粘贴着所有的Story Card，按当前的开发状态贴在4个区域中，分别是：未开发，开发中，预测试中，测试中。Story Card的开发人员和测试人员根据开发进度在Story Wall上移动Story Card，更新Story Card的状态。这种方式可以对项目开发进度有一个非常直观的了解。  
在开发人员开始开发一个Story时，ta需要找来对应的测试人员讲解Story功能，以便测试人员有一致的理解，同时开始自动化系统测试脚本的开发。  
**Standup Meeting**  
站立会议。每天早上，所有的团队成员围在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，但不浪费所有人的时间立刻解决问题，而是会后个别沟通解决。  
**Pair Programming**  
结对编程是指两个开发人员结对编码。结对编程的好处是：经过两个人讨论后编写的代码比一个人独立完成会更加的完善，一些大的方向不至于出现偏差，一些细节也可以被充分考虑到。一个有经验的开发人员和一个新手结对编程，可以促进新手的成长，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。开发人员每天将编写/修改的代码及时的更新到配置库中，自动化编译程序每天至少一次自动从配置库上取下代码，执行自动化代码静态检查（如PCLint），单元测试，编译版本，安装，系统测试，动态检查（如Purify）。以上这些自动化任务执行完毕后，会输出报告，自动发送邮件给团队成员。如果其中存在着任何的问题，相关责任人应该及时的修改。  
可以看到，整个开发组频繁的更新代码，出现一些问题不可避免。通过测试部又在不停地基于最新的代码进行测试。新增的问题是否能够被及时发现并消灭掉，取决于自动化单元测试和系统测试能力是否足够强大，特别是自动化系统测试能力。如果自动化测试只能验证最简单的操作，则新合入代码的隐患将很难被发现，并遗留到项目后期，形成大的风险。而实际上，提升自动化测试的覆盖率是最困难的。  
**Retrospect**  
总结和反思。每个迭代结束以后，项目组成员召开总结会议，总结好的实践和教训，并落实到后续的开发中。  
**ShowCase**  
演示。每个Story开发完成以后，开发人员叫上测试人员，演示软件功能，以便测试人员充分理解软件功能。  
**Refactoring**  
重构。因为迭代开发模式在项目早期就开发出可运行的软件原型，一开始开发出来的代码和架构不可能是最优的、面面俱到的，因此在后续的Story开发中，需要对代码和架构进行持续的重构。迭代开发对架构师要求很高。因为架构师要将一个完整的版本拆分成多个迭代，每个跌倒由拆分成很多Story，从架构的角度看，这些Story必须在是有很强的继承性，是可以不断叠加的，不至于后续开发的Story完全推翻了早期开发的代码和架构，同时也不可避免的需要对代码进行不断完善，不断重构。  
**TDD**  
测试驱动开发。正如上面讲的，迭代开发的特点是频繁合入代码，频繁发布版本。测试驱动开发是保证合入代码正常运行且不会在后期被破坏的重要手段。这里的测试主要指单元测试。  
  
**敏捷方法反思：**  
自己参与的敏捷开发项目总的来说不是很成功，这可能也是业界遇到的通病：  
1、对于全新的软件，在项目早期测试人员就参与并实现自动化测试脚本，但实际上软件的界面等非常不稳定，导致测试人员返工的工作量很大。  
2、对于全新的软件，资料人员过早参与，后期返工工作量大，原因同第一点。  
3、自动化系统测试工作量大，测试人员投入大量的精力在使测试自动化起来，而没有足够的精力放在真正的测试软件的功能是否正常。即便是这样，自动化系统测试脚本也多流于形式，测不出深层次的问题。  
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，报告中查出来很多告警，但不知如何消除。  
5、由于快速搭建原型，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快速构建、快速交付，团队普遍感觉很疲惫。  
7、敏捷开发不提倡加班，但实际上不管是CMM还是Agile哪一种开发模式跟是否加班都没有必然联系。