小型软件开发团队的敏捷开发方式

最近几年，敏捷开发得到了越来越多的重视，敏捷的流派很多，本身是个可以剪裁的方法体系，所以也不讲是属于什么流派的，把我能够想到的几个要点说一下，没有先后次序之分。

1. TDD（测试驱动）

测试驱动最起码自动测试是大部分敏捷方法的共有特点。我自己体会，即使不使用其他敏捷方法，测试驱动也是很有价值的。所谓测试驱动就是在写具体的类前，先写对类的测试用例，然后让测试用例跑起来，再实现类。关于什么是测试驱动，就介绍到这里，具体的可以看看《[AGILE JAVA 中文版：测试驱动开发的编程技术](http://www.china-pub.com/30528&ref=browse)》。

测试驱动有什么好处？首先编写测试套件的过程就是对类的设计过程，写测试时就是预先设定类的使用场合和方法，如果测试就很难使用类的界面的话，实际使用也不会好用。

其次是自动测试带来的好处，当你需要对类加以修改，扩展的时候，必须通过原先的测试，同时通过新的测试。这就保证了代码的修改不会带来难以预料的后果。

当然测试要满足一定的条件，比如边界测试等等必须包括在测试套件中。对于Java程序员我推荐一定看看这本书。

1. 增量开发、持续集成、版本管理

增量式开发是所有敏捷方法的共有特点。所谓增量式开发就是先实现基础用例，然后逐步实现辅助性的，枝节性的用例。增量式开发的最大好处是可以避免过度设计。遵循增量式开发实践，一开始实现的只是最基础的功能。当需要增加功能时，可能需要对原来的设计进行重构，然后再添加新的功能。相信我们都有这样的体会，一开始设计时考虑的太复杂了，很多功能其实是没有作用的。

持续集成和版本管理往往结合在一起。所谓持续集成的做法，就是每天都对提交的代码进行系统集成，编译成随时可发布的系统，同时进行回归测试，每天都能够保证能够提供一个可发布的版本。这就必然要求每一步的演进都是足够小的，但是都是可控的。

增量式开发的系统设计包括系统分析都分散到了整个开发周期中，和传统的瀑布模型构成了强烈的对比。增量开发要求的用例一开始是很笼统的，也许就是一个很粗的功能列举，当需要开发具体的某个功能时才开始细化需求用例。

1. 遵守好的代码实践

每一种语言，都有自己的好的实践。这么说吧不同的语言带来不同的代码组织方式，比如Java和C++就很不一样。当然也有共性的地方，比如减少对象间耦合度等等是一致的。这方面的话题有点微妙，主要靠平时的学习、体会，哪些是好的编码实践。这里我推荐一本参考书《[重构：改善既有代码的设计](http://www.china-pub.com/12901&ref=xilie)》。

1. 贯穿质量意识

要克服偷懒意识，在整个开发过程中注意质量，也就是需重构时就重构。如果代码很乱，是不可能有稳定的系统的。

1. 贯穿为用户提供价值意识

应该经常问问自己：用户需要什么功能？用户怎样操作最方便？

建议：

首先罗列一下我们的目标系统的功能，尽量把每个功能都列出来。

然后把功能按照实现的次序进行排队。

细化第一个功能，编写第一个功能的用例。

用一定的时间实现第一个功能。

技术细节：

我建议如果可能，可以研究一下Flex。这是目前我能够看到的在用户体验和开发复杂度方面最平衡的技术。和Dojo等Javascript方案相比，开发起来更简单，用户体验更有优势。

如果可能，建议使用stripes做jsp框架。

如果可能，建议使用ibatis做数据访问层框架。对熟悉JDBC的人而言，ibatis更简单。