想起我刚毕业后，进入一家以软件外包为主的外企做开发。它使用传统的瀑布式的软件开发流程，没有使用任何的敏捷实践。我每天上班打开电脑，拿到自己的任 务，然后从版本控制更新代码，打开工程按下Build，准备进行今天的开发任务。突然发现Build失败（通常是编译不过），大喊一声“谁Break Build啦”，也没有人响应，自己一个人郁闷，接着查看是哪些文件导致编译失败，找到最后的提交人，让他去Fix Build。后来团队里如果某个人Break Build，其他某些团队成员就在MSN的签名上写着“XXX Break Build，今天要请客吃饭”等等。其实Build失败在软件开发过程中会经常出现，不同的程序员实现自己的模块，写单元测试，完成后提交代码，难免会造 成冲突导致Build失败。但是对于开发者来说，应当能够最快的获得当前Build的反馈，如果该Build失败必须在最短的时间内修复它，以免它影响其 他人的开发进度。

  Build具体包括哪些内容呢？它不仅仅指的是编译代码，而是指编译代码，运行所有的测试（包括单元测试，功能测试等），运行代码分析（比如分析代码是否 符合编码规范），部署系统（产生可执行的软件，或者把网站部署到Web服务器上）。Build是一系列的过程用来保证代码能够运行，能够正确的运行，最后 能发布出来。

  在敏捷开发中，有一个很重要的实践叫做持续集成。而什么是持续集成呢？简单来说，持续集成是频繁、持续的在多个团队成员的工作中进行集成，并且给与反馈。一个典型的持续集成周期包括以下几个步骤：

1. 持续集成服务器不断从版本控制服务器上检查代码状态，看代码是否有更新。
2. 如果发现代码有最新的提交，那么就从版本控制服务器下载最新的代码。
3. 等代码完全更新以后，调用自动化编译脚本，进行代码编译。
4. 运行所有的自动化测试。
5. 进行代码分析。
6. 产生可执行的软件，能够提供给测试人员进行测试。

如果其中任何一个步骤失败，就表示该Build失败，持续集成服务器会给予相应的反馈。一般来说，持续集成服务器会有相应的管理界面，可以看到 Build的状态以及相应的信息，如果Build失败，可以查看原因，是编译失败还是测试失败。或者在每次Build后，持续集成服务器发邮件通知，从邮 件中可以看到最新Build的状态。当然，也可以自定义反馈方法，比如在ThoughtWorks中国，有个团队的持续集成反馈就是火山灯，黄色表示正在 进行Build，绿色表示Build成功，而红色则表示Build失败，一旦发现灯变红了，所有人都不能提交代码，而应该尽快修复该Build。还有一个 团队更有创意，用语音来进行反馈。如果Build成功，就会有语言提示说“Build XXXX成功”，如果失败，就会有提示“Build XXXX失败，是由XXX提交的”，被念到名字的成员就得停下来修复这个Build。

  持续集成实践的目的不是减少Build失败的次数，而是尽早发现问题，在最短的时间内解决问题，减少风险和浪费。如果想尝试持续集成，首先需要的是持续集 成服务器，比如CruiseControl或者VSTS；然后需要把现有的Build自动化，比如写Ant脚本；最后就是在持续集成服务器上进行配置，比 如配置版本控制，集成间隔时间，如何部署，如何反馈等。