瀑布模型的过程已经在大量软件实践中被证明是不可行的，但在实际开发工作中，仍然被大量的使用。我认真思考这个问题，可能有一个因素导致了这种落后的方法仍在继续，那就是老板的因素。

这个原因就是瀑布模型太简单。

瀑布模型有多简单，只要一个接触过软件开发的人能说完整个过程：

需求分析-->概要设计->详细设计->编码实现->测试

这个过程是完全按找软件的各个工作内容来划分开发阶段的。

这个模型的告诉大家，当软件测试完了，软件就开发完了。最爱听这句话的人是谁？当然是老板了。老板要的是什么？老板要的是管理，要的是成本控制，按照瀑布模型做东西，老板好理解，目标可以看的见，他觉得比较省心。

实际上这里忽略了一个最大的问题，需求的不确定性和软件本身的复杂性。没有一个构架师或系统分析师能在一开始把软件的过程中的所有问题都想清楚的。软件过程中产生的问题是在软件开发的推进过程中发现并解决的，而且软件本身的有些设计也是在软件开发的过程中验证的。所以确定了软件的开发过程中需要适时作出调整，这个调整就是敏捷开发中说的适应变化。

瀑布模型的结症在这里，无法适应变化，当软件发展过程中，发现需求要变更，就需要重新评估一下设计，然后更改已经实现的代码，当然还要通知测试需求测试用例，这样一来需求变化的过程才处理完成，这个过程没有错，只要产生的变化，任何软件的开发方法都是这么做的。问题是这么一来就打乱了我们原定的计划，打乱了我们看似美好的瀑布过程，甚至有的软件根本不知道该怎样该才好，本来觉得良好地

设计变成了垃圾。错在哪里？错不在变化，错不在需求，错在计划，错在瀑布模型，瀑布模型是管理者心中的乌托邦，渴望而不可及。

因为你总是想在事情开始的时候把一切都考虑到，结过最后的结果是什么都有问题。

于是人们提出了不同于瀑布的开发方法，其中很好的一种实践就是敏捷方法。其实敏捷根本不是什么系统的一套过程规范，敏捷是一种思路，是一种解决问题的思路。在我理解敏捷的思维有一下几条：

1、整体思维，不拘泥于细节，就是最需求和做设计的时候，追求一个整体的结果，而不是设计界面元素，通信细节这些琐碎的内容。所以敏捷方法的整体需求文档和设计文档都比较简练，有时候思维结果还很抽象。不过早的进入细节，是这里边最重要的思想。这个思维活动的结果是整体关系，如果探讨细节那一定是在验证整体。

2、小版本计划，大家都说小版本迭代，其实过程的开始时一个小版本计划，在一次迭代过程中，项目还是瀑布模型的。当然这就不可避免的又产生了开头想不清楚，中间产生了变化的问题，但项目计划比较小。所以也更可控。

3、构架师的作用，这个方法中，构架师的整体把控能力及其重要，一切都在变，没有一个优秀的构架师敏捷就变成了混乱。

4、最郁闷的人：敏捷方法中，最郁闷的就是老板了，因为他要花更多的时间来跟踪项目。