软件质量属性之可修改性

统观软件质量属性，一般分为六大质量属性：可用性分析，可修改性分析，性能分析，安全性分析，可测试性分析，易用性分析。今天我将主要针对可修改性进行阐述。

可修改性可以理解为：指系统或软件的能够快速地以较高的性价比对系统进行变更的能力。比如说：对于一个网站，我们要修改它某一板块的UI界面，当我们对界面进行修改时是否会引起对另一个UI模块的影响，是否会引起后台控制，业务逻辑代码的变更，是否会引起整个网站的崩溃，这体现了一个网站的整个架构的是否具备可修改性。

我们为什么要修改我们的代码呢？原因有很多，但是统一来说就是个需求的问题：它可包含两个方面：1、用户需求，2、系统内在需求。在这解释下需求，成本，修改三者的关系：需求无处不在，时时刻刻产生，判别一个需求的重要性来自于它对系统的成本产生的影响，如果严重影响了系统带来的收益，那必须对系统进行修改，如果某一部分相对于系统收益来说微不足道，甚至不会影响系统的收益，那改不改都“无可厚非”，可能你对这部分修改了还引起系统故障了咋办。例如：一个网站的图标有人觉得不好看，有人觉得不错。使用的人们已经习惯这个图标的表示或者惯性的认为这个图标代表什么。那么是否对这个图标进行修改？我以为没有必要修改，如果你修改了反而可能会引起使用者的错误判断，比如：这个网站是停止运维了吗？好，那我去用其他的系统吧。类似错误的判断会影响到系统的收益，所以不进行修改就挺好的。

以淘宝为说明对象，更具相关资料，初期的淘宝使用的是MySQL作为它的数据库，众所周知MySQL是一个适用于小型企业的数据库，对于起初的淘宝也是适用的，因为没有产生MySQL无法处理的大量数据及相应的实时处理业务。但是近几年来淘宝的数据爆炸产生，在2012年11月30号到达 - 全天访问用户总人数：2亿1千3百万，占中国网民40%；高峰期：每分钟成交订单89678笔.....，这些大量业务数据就会引起系统“自我反馈”（自己对自己的内在需求的反应，回馈（强行解释下）），在这总系统本身产生的需求下，如果不进行系统数据库方面的修改会严重影响到淘宝带来的收益。在此大背景之下，淘宝进行了系统的修改。直至今日，阿里拥有了属于自己的阿里云，在2020年猝不及防的疫情前，这强大的服务器让阿里成为在重大疫情前的龙头。

可修改性战术的目标是控制实现、测试和部署变更的时间和成本。解释为根据相关战术，策略在时间和成本允许的范围内完成系统的相关变更。

两种可修改性战术：局部化修改和防止连锁反应

局部化修改战术，目标是把变更限制在一定范围，在设计期间为模块分配职责，把预期变更限制在一定范围内。关键点就是“限制变更范围”防止变更的内容对其他内容（或者说程序）产生影响。

防止连锁反应，我们平时编程无论是写函数还是写类，都会被其它的类或函数（方法）调用，如何实现对被调用部分的修改不会引起对调用它的部分的影响（有点绕），这就是 - 连锁反应。

无论是局部化修改还是防止连锁反应，都是基于“高内聚，低耦合”思想。

局部化意味着实现“模块化”思想，这里模块这样解释，一个模块只完成一个部分，这就符合设计模式中“单一职责原则”的设计原则，使每一个模块责任单一，防职责过多引起引起整体变更时的繁琐，复杂。

要符合“接口隔离”设计原则，为后期的变更提供接口，当然也要规定接口调用的规范。同时根据“里氏代换”原则将接口的实例化由接口实现类完成，在此之上基于“依赖倒转原则”搭建的上层服务如果需要修改相关服务可以实现局部化修改，不用再对上层服务进行修改。在防止连锁反应的战术中“维持现有接口”就是为上层服务提供一个可使用的接口，当进行变更时对接口的实现类变更即可，无论是实现“适配器”还是“信息隐藏”都可封装在接口实现类中。