阅读书籍：Software Architecture in Practice ，3rd Edition

第八章8. Modifability章节阅读笔记

冯少浦

Modifabilty是什么？

简单来说是软件的可修改性，宇宙中唯一不变的是改变，用户的需求，新的软硬件平台，协议的改变，旧功能的删减都需要程序的更改，因此对软件体系结构来说可修改性很重要。

关于Modifabilty需要关注的4个问题：

1. 什么会改变？（需求，环境，协议等）
2. 改变的几率是多少？
3. 改变合适会发生？由谁造成？
4. 改变的代价是什么？

给出了判断是否值得引入增强Modifabilty的机制的**判断不等式**，假如该不等式成立，说明引入该Mechanism后的Cost小于不引入的Cost，因此值得引入。

*N \* Cost of making change without the mechanism ≤  
Cost of creating the mechanism + (N \* cost of making the change using the mechanism)*

Modifabilty的几个特化的Flavor：

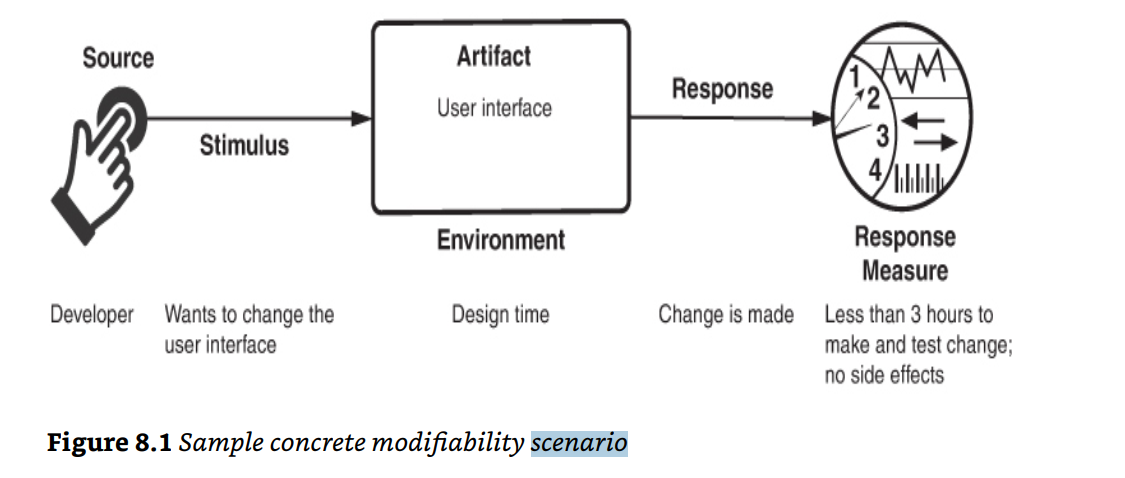
Variability:指系统应对预计计划中某些特征的不同变化的能力。大概的理解就是假如我们设计的程序设计算a+b，那么就要对a+b的所有不同取值都尽量可以计算,强调的是计划中的。

Scalability：主要指的是容量上的可修改性，比如从支持10个用户到支持100个用户皆可。

Portability：指体系结构在不同平台上的可修改性，比如对X86和ARM的兼容性，对Win和Linux的兼容性。

Location independence:比如分布式服务中，对某一地方服务提供方的修改应尽可能减少对其他地方服务提供方的修改。

Modifabilty scenario（典型场景）

**

实现Modifabilty的策略：

1. 强内聚
2. *Split module:将模块分解，分解后的模块必然内聚更强。*
3. *Redistribute responsibilities：假如相似的功能A,B,C分散在不同的模块中，则重新划分模块的职责，减少这种不必要的冗余。*
4. 降低耦合
5. 封装
6. 利用中介
7. 将公共服务抽象化。
8. 限制依赖。
9. 延迟绑定：

大致意思是函数f(a,b)先让比只有一个变量的函数f(a) （假设此时b=0）更加的general,即不提前假设b=0，而是让用户决定，拥有更大的变化空间，因此也就有了更好的modifabilty。

提升Modifability的Pattern(大致与课本中提到的中文的体系结构风格相对应)

1. Client-Server Pattern（客户服务机）
2. Plug-in (微内核) Pattern
3. Layers Pattern（分层的体系结构风格）
4. Publish-Subscribe Pattern(出版-订阅模式)