[**软件架构师之拥抱变化**](http://www.cnblogs.com/niyw/archive/2010/08/29/1811665.html)

　　你是否正在被不断变化的需求折磨得焦头烂额？！

　　你是否在为繁冗复杂项目抓耳挠腮？！

　　相信这是很多人现在正面临的问题。我们在学习软件架构时经常能看到拥抱变化的字眼，我们也知道什么是**拥抱变化**，也知道拥抱变化是解决上述问题的最优途径。然而，如何拥抱变化才是解决问题的关键所在。每每此时，各种书本都会把路标指向设计模式，各种架构模式等，大家每个人**看了以后大都恍然大悟，而付诸于实践时则仍旧一脸茫然**。那么如何做到拥抱变化呢？

　　首先，要从**软件架构的根本**说起。我们为什么要进行软件架构设计？！答案很简单，**因为有变化**，并且是很多的不断的往往难以容忍的变化。如果没有变化，世界上就不会没有软件架构，今天很多做软件的人肯定在做从事职业。相信这么说没几个人会持反对意见。

　　其次，**任何软件项目都要解决实际的业务，不解决业务的软件根本不存在，而变化则来自于要解决的业务问题**。也会你认为这是废话，但明白这点很重要。大家都知道，解决问题并不难，发现问题才是最难得。**业务是解决现实世界中的某一类问题，因此它有特定的规律，源子于业务的变化则同样拥有这些规律**。明白这点，我们就向成功迈出了关键的一步。

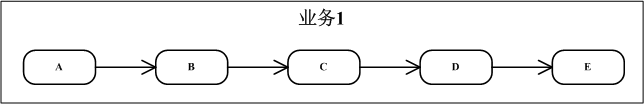
　　第三，既然变化有规律，那么就可以分析总结。因为变化来自业务，所以要从业务入手，**通常业务可以划分为业务流程和业务单元**，如图1：

图1

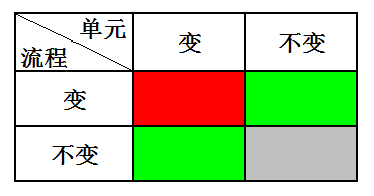
　　图1是一个抽象的业务1，从A到E是一个完整的业务流程，其中A，B，C，D，E是业务中的业务单元。那么我们可以总结出业务的变化规律，如图2：

图2

　　显然，业务变化仅有4种情况。抛除流程和单元均不变得情况（灰色），实际上只有三种，其中最容易遇到的就是一个变化一个不变（绿色）。二者均变化的情况很少，当出现变化时，要么是项目定位出错，注定失败；要么业务划分粒度不够或不合理。因此，通常我们重点把握的变化只有两种，分别是：第1，业务流程不变，业务单元发生变化；第2，业务流程变化，业务单元不变。下面我们就对如何把握这两种情况做一一说明。

**业务流程不变，业务单元发生变化**

　　这种变化最容易理解，举个最简单的例子。当我们去商店/超市购物时，通常的流程都是首先选购商品，然后付账，最后交易完成。像这样的业务流程应该说自商店/超市出现起到现在都没有发生变化过，但是支付方式却发生了很多变化，从最早的金属货币，到后来的纸币，再到信用卡，甚至今天的手机支付等等。如此相信不难理解下图：

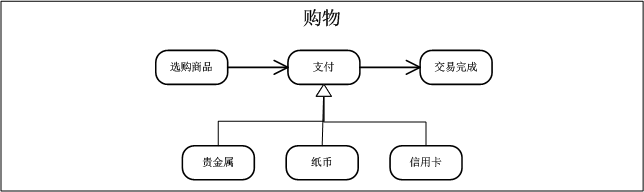


图3

那么此类变化该如何解决呢？通常我们可以为此类业务构建一个比较稳定的框架(Framework)，在可能变化的地方(如图3中的支付)抽象出接口，使用依赖倒置等方法实现可能变化，然后根据条件，调用实际的实现（如选择纸币支付方式）；

**业务流程变化，业务单元不变**

　　以订票为例说明此类变化，如下图：

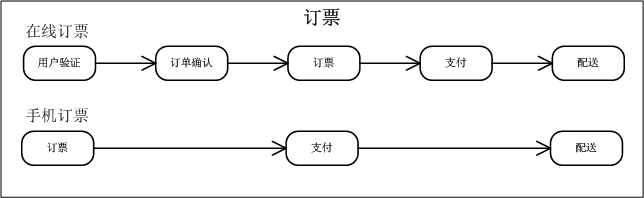


图4

由图4可以看出，对用户来说同样都是订票业务，事实上有不同的实现方式(业务流程)，但是其中的某些业务单元，如订票，支付，配送完全一样。由此可以总结出此类变化的特点，就是在不同的情况下，同样的业务可能要通过不同的组合(业务单元)实现。相信这点不难理解。那么此类变化该如何解决呢？现在很流行的SOA为我们提供了很好的解决方法。我们可以将每个业务单元以服务的形式发布，从而解决了业务单元直接的耦合，在需要的时候装载不同的单元服务，从而实现不同的业务流程，甚至灵活构建新的业务而不必担心对现有业务造成任何影响。

　　最后，通过以上对业务变化的分析，相信大家对软件架构设计中如何拥抱变化有了一些新的认识。事实上，今天很多的软件架构方法都是解决业务变化中的某一类型的变化，如SOA，AOP等等，只是它们的关注点不同。只要把握了它们的关注点，相信拥抱变化不是难事！