极限编程与敏捷开发

徐景周

在按照我的理解方式审查了软件开发的生命周期后，我得出一个结论：实际上满足工程设计标准的惟一软件文档，就是源代码清单。

-- Jack Reeves

简介

2001 年，为了解决许多公司的软件团队陷入不断增长的过程泥潭，一批业界专家一起概括出了一些可以让软件开发团队具有快速工作、响应变化能力的价值观和原则，他们称自己为敏捷联盟。敏捷开发过程的方法很多，主要有：SCRUM，Crystal,特征驱动软件开发（Feature Driven Development，简称 FDD），自适应软件开发(Adaptive Software Development，简称 ASD)，以及最重要的极限编程(eXtreme Programming,简称 XP)。极限编程(XP)是于 1998 年由 Smalltalk 社群中的大师级人物 Kent Beck 首先倡导的。

极限编程

设计和编程都是人的活动。忘记这一点，将会失去一切。

-- Bjarne Stroustrup

极限编程(XP)是敏捷方法中最著名的一个。它是由一系列简单却互相依赖的实践组成。这些实践结合在一起形成了一个胜于部分结合的整体。

下面是极限编程的有效实践：

1、完整团队

XP 项目的所有参与者（开发人员、客户、测试人员等）一起工作在一个开放的场所中，他们是同一个团队的成员。这个场所的墙壁上随意悬挂着大幅的、显著的图表以及其他一些显示他们进度的东西。

2、计划游戏

计划是持续的、循序渐进的。每 2 周，开发人员就为下 2 周估算候选特性的成本，而客户则根据成本和商务价值来选择要实现的特性。

3、客户测试

作为选择每个所期望的特性的一部分，客户可以根据脚本语言来定义出自动验收测试来表明该特性可以工作。

4、简单设计

团队保持设计恰好和当前的系统功能相匹配。它通过了所有的测试，不包含任何重复，表达出了编写者想表达的所有东西，并且包含尽可能少的代码。

5、结对编程

所有的产品软件都是由两个程序员、并排坐在一起在同一台机器上构建的。

6、测试驱动开发

编写单元测试是一个验证行为，更是一个设计行为。同样，它更是一种编写文档的行为。编写单元测试避免了相当数量的反馈循环，尤其是功功能能验

证方面的反馈循环。程序员以非常短的循环周期工作，他们先增加一个失败的测试，然后使之通过。

7、改进设计

随时利用重构方法改进已经腐化的代码，保持代码尽可能的干净、具有表达力。

8、持续集成

团队总是使系统完整地被集成。一个人拆入（Check in）后，其它所有人责任代码集成。

9、集体代码所有权

任何结对的程序员都可以在任何时候改进任何代码。没有程序员对任何一个特定的模块或技术单独负责，每个人都可以参与任何其它方面的开发。

10、编码标准

系统中所有的代码看起来就好像是被单独一人编写的。

11、隐喻

将整个系统联系在一起的全局视图；它是系统的未来影像，是它使得所有单独模块的位置和外观变得明显直观。如果模块的外观与整个隐喻不符，那么你就知道该模块是错误的。

12、可持续的速度

团队只有持久才有获胜的希望。他们以能够长期维持的速度努力工作，他们保存精力，他们把项目看作是马拉松长跑，而不是全速短跑。

极限编程是一组简单、具体的实践，这些实践结合在形成了一个敏捷开发过程。极限编程是一种优良的、通用的软件开发方法，项目团队可以拿来直接采用，也可以增加一些实践，或者对其中的一些实践进行修改后再采用。

敏捷开发

人与人之间的交互是复杂的，并且其效果从来都是难以预期的，但却是工作中最重要的方面。

-- Tom DeMacro 和 Timothy Lister

敏捷软件开发宣言：

- 个体和交互　　胜过　　过程和工具
- 可以工作的软件 胜过　　面面俱到的文档
- 客户合作　　胜过　　合同谈判
- 响应变化　　胜过　　遵循计划

虽然右项也有价值，但是我们认为左项具有更大的价值。

敏捷宣言遵循的原则：

- 我们最优先要做的是通过尽早的、持续的交付有价值的软件来使客户满意。
- 即使到了开发的后期，也欢迎改变需求。敏捷过程利用变化来为客户创造竞争优势。
- 经常性地交付可以工作的软件，交付的间隔可以从几个星期到几个月，交付的时间间隔越短越好。
- 在整个项目开发期间，业务人员和开发人员必须天天都在一起工作。

- 围绕被激励起来的个体来构建项目。给他们提供所需的环境和支持，并且信任他们能够完成工作。
- 在团队内部，最具有效果并富有效率的传递信息的方法，就是面对面的交谈。
- 工作的软件是首要的进度度量标准。
- 敏捷过程提倡可持续的开发速度。责任人、开发者和用户应该能够保持一个长期的、恒定的开发速度。
- 不断地关注优秀的技能和好的设计会增强敏捷能力。
- 简单是最根本的。
- 最好的构架、需求和设计出于自组织团队。
- 每隔一定时间，团队会在如何才能更有效地工作方面进行反省，然后相应地对自己的行为进行调整。

当软件开发需求的变化而变化时，软件设计会出现坏味道，当软件中出现下面任何一种气味时，表明软件正在腐化。

- 僵化性： 很难对系统进行改动，因为每个改动都会迫使许多对系统其他部分的其它改动。
- 脆弱性：对系统的改动会导致系统中和改动的地方在概念上无关的许多地方出现问题。
- 牢固性： 很难解开系统的纠结，使之成为一些可在其他系统中重用的组件。
- 粘滞性： 做正确的事情比做错误的事情要困难。
- 不必要的复杂性： 设计中包含有不具任何直接好处的基础结构。
- 不必要的重复性： 设计中包含有重复的结构，而该重复的结构本可以使用单一的抽象进行统一。
- 晦涩性： 很难阅读、理解。没有很好地表现出意图。

敏捷团队依靠变化来获取活力。团队几乎不进行预先设计，因此，不需要一个成熟的初始设计。他们更愿意保持设计尽可能的干净、简单，并使用许多单元测试和验收测试作为支援。这保持了设计的灵活性、易于理解性。团队利用这种灵活性，持续地改进设计，以便于每次迭代结束生成的系统都具有最适合于那次迭代中需求的设计。

为了改变上面软件设计中的腐化味，敏捷开发采取了以下面向对象的设计原则来加以避免，这些原则如下：

- 单一职责原则(SRP)
  就一个类而言，应该仅有一个引起它变化的原因。
- 开放-封闭原则(OCP)
  软件实体应该是可以扩展的，但是不可修改。
- Liskov 替换原则(LSP)
  子类型必须能够替换掉它们的基类型。
- 依赖倒置原则(DIP)
  抽象不应该依赖于细节。细节应该依赖于抽象。
- 接口隔离原则(ISP)
  不应该强迫客户依赖于它们不用的方法。接口属于客户，不属于它所在的类层次结构。

- 重用发布等价原则(REP)
  重用的粒度就是发布的粒度。
- 共同封闭原则(CCP)
  包中的所有类对于同一类性质的变化应该是共同封闭的。一个变化若对一个包产生影响，则将对该包中的所有类产生影响，而对于其他的包不造成任何影响。
- 共同重用原则(CRP)
  一个包中的所有类应该是共同重用的。如果重用了包中的一个类，那么就要重用包中的所有类。
- 无环依赖原则(ADP)
  在包的依赖关系图中不允许存在环。
- 稳定依赖原则(SDP)
  朝着稳定的方向进行依赖。
- 稳定抽象原则(SAP)
  包的抽象程度应该和其稳定程度一致。

上述中的包的概念是：包可以用作包容一组类的容器，通过把类组织成包，我们可以在更高层次的抽象上来理解设计，我们也可以通过包来管理软件的开发和发布。目的就是根据一些原则对应用程序中的类进行划分，然后把那些划分后的类分配到包中。

下面举一个简单的设计问题方面的模式与原则应用的示例：

问题：

选择设计运行在简易台灯中的软件，台灯由一个开关和一盏灯组成。你可以询问开关开着还是关着，也可以让灯打开或关闭。

解决方案一：

下面**图 1** 是一种最简单的解决方案，Switch 对象可以轮询真实开关的状态，并且可以发送相应的 turnOn 和 turnOff 消息给 Light。

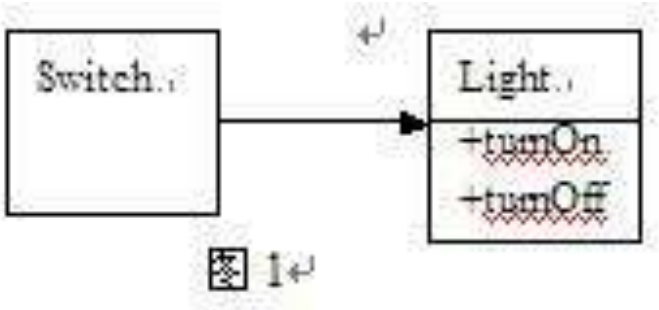

图 1

解决方案二：

上面这个设计违反了两个设计原则：依赖倒置原则(DIP)和开放封闭原则(OCP)，DIP 原则告诉我们要优先依赖于抽象类，而 Switch 依赖了具体类 Light，对 OCP 的违反是在任何需要 Switch 的地方都要带上 Light，这样就不能容易扩展 Switch 去管理除 Light 外的其他对象。为了解决这个方案，可以采用

ABSTRACT SERVER 模式，在 Switch 和 Light 之间引入一个接口，这样就使得 Switch 能够控制任何实现了这个接口的东西，这也就满足了 DIP 和 OCP 原则。如下面图 2 所示：

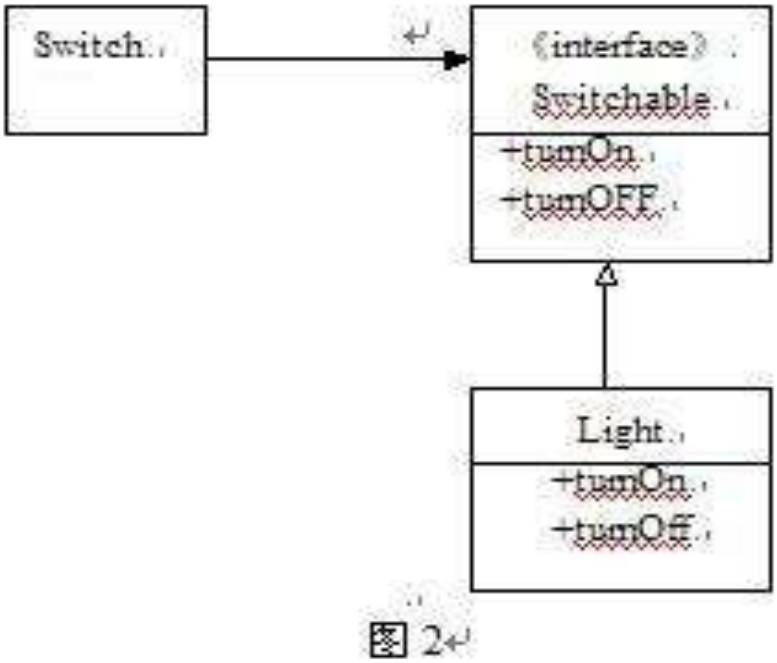

图 2

解决方案三：

上面图 2 所示解决方案，违返了单一职责原则(SRP)，它把 Switch 和 Light 绑定在一起，而它们可能会因为不同的原因而改变。这种问题可以采用 ADAPTER 模式来解决，适配器从 Switchable 派生并委托给 Light，问题就被优美的解决了，现在，Switch 就可以控制任何能够被打开或者关闭的对象。但是这也需要会出时间和空间上的代价来换取。如下面图 3 所示：

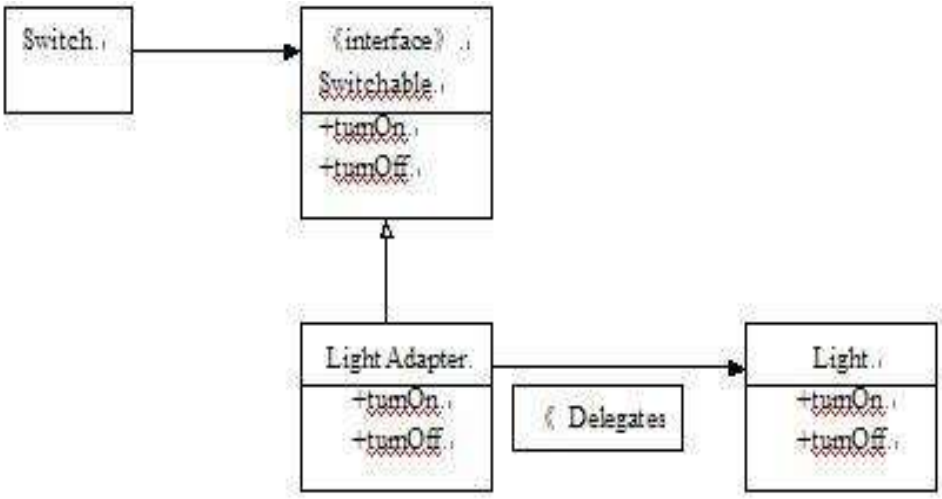

图 3

敏捷设计是一个过程，不是一个事件。它是一个持续的应用原则、模式以及实践来改进软件的结构和可读性的过程。它致力于保持系统设计在任何时间都尽可能得简单、干净和富有表现力。

参考文献

设计模式-可复用面向对象软件的基础 -- 李英军等译
重构-改善既有代码的设计 -- 侯捷等译
敏捷软件开发-原则、模式与实现 -- 邓辉译

联系方式
地址：陕西省西安市劳动路 90 号院(台板厂家属院)六单元
邮编：710082
Email: jingzhou_xu@163.net
未来工作室(Future Studio)