如何保证软件设计的质量

凌硕 14126115
北京交通大学软件学院 软件工程
14126115@bjtu.edu.cn

随着信息技术的发展，软件作为信息技术的基石日益发挥着至关重要的作用。软件质量保证作为软件工程的重要研究范畴，正在逐步受到各大互联网公司和软件企业的重视。软件工程学的重点关注点之一即是软件的设计，软件工程的源起之一也是于此。自从软件危机爆发以后，各个公司和信息技术研究机构开始关注如何设计出可靠的软件。换句话说，也就是软件设计的质量该如何保证。但是当时并没有软件质量相关的一整套方法论和实践指南，于是，一代又一代的软件设计师们从纷繁复杂的各式各样的软件设计开发实践当中，总结和归纳了一系列模型和范式，从现象中抽象得出了指导我们软件设计的一整套方法论，从而使得我们的软件设计的质量可以得到保证，软件项目的成功率可以得到极大的提升。在本文接下来的篇幅里，笔者将会详细阐述这些可以保证软件设计质量的方法，以供参考。

首先我们需要明确的一个问题就是什么是软件设计。软件设计一般可以分为两部分，体系结构设计（架构设计）和详细设计。体系结构设计也叫架构设计，是一个高层次的设计。针对软件需求，将需求转化为数据结构或者软件的系统结构，将整个系统拆分为模块或者子系统，并将需求中的联系映射到软件系统对象模块子系统之间的联系上，并做好通信和接口的设计。详细设计是一个在体系结构设计基础之上的细化的设计，将抽象的架构设计具体变成可以编程实现的详细的类设计，数据结构和算法。总而言之，软件设计的基本任务包括架构设计，详细设计，设计文档编纂和设计评审等部分，通过这样的过程，我们的质量保证这样的软件可以具有安全可靠，易于扩展，可移植，可重用等各种优秀软件所具备的特质。

明确了问题的对象，接下来笔者将详细阐述如何保证软件设计的质量。

有调查显示，绝大多数失败的软件项目失败的根源都可以追溯到最开始的软件设计当中，我们提升软件设计质量的方法有如下几条：需求控制和变更管理，使用恰当的设计模式和引入软件质量保证评审控制机制。接下来我将会对这三点进行详细的分析和阐述。

一、需求控制和变更管理

在传统的软件时代里，绝大多数软件开发都遵循着这样的一个范式：确定需求，设计软件，编码，测试，交付，接收新的需求，再次设计……如此循环往复。在需求变化相对不是那么迅速的传统的软件开发时代里，这种模式还是可以勉强接收的。但是随着互联网时代的来临，这样一种模式在瞬息万变的信息时代中，落伍了。在互联网开发中，用户为主，需求是一直在变化的。需求变更和控制就成为了当代软件开发的必备。为了适应这样的变革，我们在设计软件的时候一

定要保证软件的可扩展性。在保证可扩展性的同时，也需要对于扩展的方向进行一定的控制，也就是需求变更管理。

现代的软件项目，不论是传统的业务逻辑项目还是新兴的互联网项目，需求绝不可能在一开始的时候就全部确定，我们在拿到第一版的最初的需求的时候，不要着急去着手设计，首先要对客户提出的这些需求进行分析，找到其核心的部分以及这些核心部分之间的联系。从而对客户提出的需求的业务模型有一个大致的了解。接下来对这个模型进行分析，找到可能会发生变化的点，并进行具体分析。这一个过程也就是需求风险评估的过程。

一般来说，描述不明确的需求，需求主客体限定不明的需求，应该量化而没有的需求，结构流程未原子化的需求，是具有风险的需求。在一开始，做需求分析的人员就要与客户协商，将这些不明确的需求逐一明确化，具体化，并从这样的过程当中分析得出未来可能发生变化的点。这就决定了软件项目的顶层设计。有了良好的基于变更的顶层设计，才能有可扩展的架构设计，才谈得上具有质量保证的软件设计。

二、使用恰当的设计模式

设计模式，是在大量的软件开发实践当中总结出来的最佳实践范式的一个集合。在最开始被提出来的时候，只有二十余种设计模式。随着软件工程的发展和进步，很多新的设计模式在不断地被提出，一些旧的设计模式也在被更新或者被淘汰。设计模式听起来很高大上，看似运用了之后就有点石成金之效，不过事情好像不一定如此。

设计模式的运用与否，和软件架构设计的好坏之间的关系并没有大家想象中的那样具有决定性的作用。具有决定性作用的，在于“恰当”二字上。在需要使用设计模式的时候，选择使用恰当的设计模式，可以极大地提升软件架构的质量，如果滥用设计模式，不但软件质量不会有明显的提升，反而会导致软件的理解难度加大，不利于后期的维护和扩展。

为了提高软件设计的质量，在使用设计模式这一块中，我们应当如何做到恰当的使用呢？

在进行架构设计的第一步，我们对这个项目的子系统和子模块进行分解。分解的时候依据的原则很多，其中我觉得最最重要的是前两个，第一是开放-封闭原则，各个子模块对扩展是开放的，而对于修改是封闭的。对扩展开放，意味着有新的需求或变化时，可以对现有代码进行扩展，以适应新的情况。对修改封闭，意味着类一旦设计完成，就可以独立完成其工作，而不要对类进行任何修改。这个原则的核心是要依赖抽象编程而不是具体。因为抽象的相对稳定的，一旦确定了不会去进行修改的，所以对修改就是封闭的。另一个方面，通过面向对象的继承和对多态机制，我们可以实现对抽象的继承，通过在子类中覆盖实现父类的方法，使得子类具有可扩展性，所以对于扩展就是开放的。这是实施开放封闭原则的基本思路。第二是单一职责原则。每一个子模块，只负责一件事情，或者只保持着一个对象。设计软件的宗旨之一就是将复杂是事务逻辑简化，一个模块一个类一个方法只负责一件事情，这样在未来扩展的时候就可以十分明确而简单。

在明确了架构设计的大体框架之后，我们需要根据具体情况进行细化。我们的架构是否是分层的，如果是分层的，分了几层，每一层应当使用什么样的设计模式简化我们的开发。这一系列问题都需要我们具体问题具体分析。设计模式主要分为三大类，创建型模式，结构型模式和行为模式。在系统架构顶层设计的时候，我们创建系统对象的时候，使用的是创建型模式下面的一些设计模式，例如工厂模式等。在具体进入模块设计的时候，我们采用的更多的是结构型模式下属的，例如适配器，桥接，代理等设计模式，来组织模块之间的联系。具体到模块内部的行为，我们有策略模式，观察者模式等模式，具体的将模块内部的行为进行抽象和封装，以解除模块之间的耦合。

在设计架构的同时，一个非常重要的一点是如何将其简明而又清晰的表达出来。这时候，统一软件建模语言 UML 就起到了至关重要的作用。UML 是一门专门用来描述软件设计的图形语言。它用一整套约定好的图形和符号，描述软件的架构，流程等详细设计的组成部分，旨在建立起开发，设计与客户三者之间的桥梁。在我们进行架构设计的时候，我们需要画出 UML 图，图上往往可以看到一些我们在设计的时候没有考虑到的，没有注意到的一些问题。这时候往往就需要引入评审机制了。

三、引入软件质量保证评审控制机制

在进行架构的初步设计时，我们从事设计的架构师，因为“当局者迷”的缘故，往往看不到其中可能存在的一些问题。这时候我们就需要引入其他人来对设计产生的结果也就是 UML 图进行评审。参加架构设计评审的人，应当有这三类，一是设计软件的同行，二是客户中对业务逻辑十分熟悉的人的代表，三是有经验的高级架构设计师。由这三方对同一个软件设计方案进行评审，并将评审的结果及时反馈给设计师。设计师拿到评审的结果之后，参考给出的建议，对架构设计进行进一步的修改，并再次提交评审。这样反复，次数不宜过多，直到三方都觉得这样的软件架构设计没有什么问题了，就可以确定设计，交给开发人员进行具体的开发了。

综上所述，为了保证软件设计的质量，软件设计师采取了各种方法，去控制变更，去运用前人的智慧和经验总结出来的模型和范式来提升软件的设计，引入评审机制，不断地交流和沟通以保证软件的理解没有二义性。做到了以上几点，这一款软件设计的质量可以说是得到了充分的保证了。

未来我们要走向工作岗位，具体从事软件的设计和开发工作。对软件质量保证的学习和研讨，对于我们的未来，是十分有益处的。在这样一个互联网时代，软件不再是单枪匹马单打独斗的产物，而是在于协调，协作和沟通。在这样的环境下，遵循这上面提到的一系列原则，模型和范式，软件的质量就一定可以得到很好的保证。