如何保证软件设计的质量

14126176　翟博渊

要保证一个软件设计的质量，我认为主要包括两个方面，第一是如何做好软件设计之前的需求分析，因为只有准确把握住软件用户的需求，我们才能做出用户真正想要的软件，需求做不好，设计出的软件连用户的最基本问题都解决不了，那么这个软件就失去的它最本质的功能和意义，更谈不上质量问题了。第二点，在做好用户需求分析的基础上，如何做出好的软件设计，需要哪些方法和手段，有哪些需要遵守的原则，这也是保证软件质量的一个重要因素。

要做好软件需求分析，我们首先要明确的是什么是软件需求分析。简单来说，软件需求分析就是把软件计划期间建立的软件可行性分析求精和细化，分析各种可能的解法，并且分配给各个软件元素。需求分析是软件定义阶段中的最后一步，是确定系统必须完成哪些工作，也就是对目标系统提出完整、准确、清晰、具体的要求。

开发软件系统最为困难的部分就是准确说明开发什么。最为困难的概念性工作便是编写出详细技术需求，这包括所有面向用户、面向机器和其它软件系统的接口。同时这也是一旦做错，将最终会给系统带来极大损害的部分，并且以后再对它进行修改也极为困难。我们之所以要做软件需求分析，就是为了深入了解和描述软件的功能和性能，确定软件设计的约束和软件同其他系统元素的接口细节，定义软件的其他有效性需求，借助于当前系统的逻辑模型导出目标系统逻辑模型，解决目标系统要“做什么”的问题。做好了软件需求分析，就好像航海前准备好了精确的罗盘，有了需求的指引，软件的设计才能贴合用户的期望，做出有用的软件产品。

需求分析可分为问题获取、分析、编写规格说明和验证四个阶段。这些子项包括软件类产品中需求收集、评价、编写文档等所有活动。需求开发活动包括以下几个方面：

1.确定产品所期望的用户类别；

2.获取每个用户类的需求；

3.了解实际用户任务和目标以及这些任务所支持的业务需求；

4.分析源于用户的信息以区别用户任务需求、功能需求、业务规则、质量属性、建议解决方法和附加信息；

5.将系统级的需求分为几个子系统，并将需求中的一部份分配给软件组件；

6.了解相关质量属性的重要性；

7.商讨实施优先级的划分；

8.将所收集的用户需求编写成文档和模型；

9.评审需求规格说明，确保对用户需求达到共同的理解与认识，并在整个开发小组接受说明之前将问题都弄清楚。

同时我们也需要对需求进行管理，需求管理需要"建立并维护在软件工程中同客户达成的合同" 。这种合同都包含在编写的需求文档与模型中。客户的接受仅是需求成功的一半，开发人员也必须能够接受他们，并真正把需求应用到产品中。通常的需求管理活动包括：

1.定义需求基线（迅速制定需求文档的主体）；

2.评审提出的需求变更、评估每项变更的可能影响从而决定是否实施它；

3.以一种可控制的方式将需求变更融入到项目中；

4.使当前的项目计划与需求一致；

5.估计变更需求所产生影响并在此基础上协商新的承诺，这种承诺具体体现在项目解决方案上；

6.让每项需求都能与其对应的设计、源代码和测试用例联系起来以实现跟踪；

7. 在整个项目过程中跟踪需求状态及其变更情况。

软件需求包括三个不同的层次：业务需求、用户需求和功能需求（也包括非功能需求）。业务需求反映了组织机构或客户对系统、产品高层次的目标要求，它们在项目视图与范围文档中予以说明。用户需求文档描述了用户使用产品必须要完成的任务，这在使用实例文档或方案脚本说明中予以说明。功能需求定义了开发人员必须实现的软件功能，使得用户能完成他们的任务，从而满足了业务需求。在软件需求规格说明书中说明的功能需求充分描述了软件系统所应具有的外部行为。软件需求规格说明在开发、测试、质量保证、项目管理以及相关项目功能中都起了重要的作用。对一个大型系统来说，软件功能需求也许只是系统需求的一个子集，因为另外一些可能属于子系统（或软件部件）。

作为功能需求的补充，软件需求规格说明还应包括非功能需求，它描述了系统展现给用户的行为和执行的操作等。它包括产品必须遵从的标准、规范和合约，外部界面的具体细节，性能要求，设计或实现的约束条件及质量属性。所谓约束是指对开发人员在软件产品设计和构造上的限制。质量属性是通过多种角度对产品的特点进行描述，从而反映产品功能。多角度描述产品对用户和开发人员都极为重要。

做完了软件需求分析，下面就要针对软件的需求开始进行软件设计。软件设计就是从软件需求规格说明书出发，根据需求分析阶段确定的功能设计软件系统

的整体结构、划分功能模块、确定每个模块的实现算法以及编写具体的代码，形成软件的具体设计方案。软件设计是把许多事物和问题抽象起来，并且抽象它们不同的层次和角度。将问题或事物分解并模块化使得解决问题变得容易，分解的越细模块数量也就越多，它的副作用就是使得设计者考虑更多的模块之间耦合度的情况。我们要做的就是如何调节模块化和耦合度之间的平衡，这也是软件质量的关键所在。

除了做好需求分析之外，软件设计要遵循一定的原则，这样设计出的软件才不会太过天马行空，没有结构，才能保证软件设计的质量。常规的软件设计原则有以下几个：

1.开闭原则一个软件实体应当对扩展开发，对修改关闭。说的是，再设计一个模块的时候，应当使这个模块可以在不被修改的前提下被扩展。换言之，应当可以在不必修改源代码的情况下改变这个模块的行为，在保持系统一定稳定性的基础上，对系统进行扩展。这是面向对象设计的基石，也是最重要的原则。

2.里氏代换原则，一个软件实体如果使用的是一个基类的话，那么一定适用于其子类，而且它根本不能察觉出基类对象和子类对象的区别。只有衍生类可以替换基类，软件单位的功能才能不受影响，基类才能真正被复用，而衍生类也能够在基类的基础上增加新功能。

3.依赖倒置原则，要求客户端依赖于抽象耦合。针对接口编程的意思是说，应当使用接口和抽象类进行变量的类型声明，参量的类型声明，方法的返还类型声明，以及数据类型的转换等。不要针对实现编程的意思就是说，不应当使用具体类进行变量的类型声明，参量类型声明，方法的返还类型声明，以及数据类型的转换等。要保证做到这一点，一个具体的类应等只实现接口和抽象类中声明过的方法，而不应当给出多余的方法。

只要一个被引用的对象存在抽象类型，就应当在任何引用此对象的地方使用抽象类型，包括参量的类型声明，方法返还类型的声明，属性变量的类型声明等。

4.接口隔离原则，一个类对另外一个类的依赖是建立在最小的接口上。使用多个专门的接口比使用单一的总接口要好。根据客户需要的不同，而为不同的客户端提供不同的服务是一种应当得到鼓励的做法。就像"看人下菜碟"一样，要看客人是谁，再提供不同档次的饭菜。胖接口会导致他们的客户程序之间产生不正常的并且有害的耦合关系。当一个客户程序要求该胖接口进行一个改动时，会影响到所有其他的客户程序。因此客户程序应该仅仅依赖他们实际需要调用的方法。

5.合成/聚合复用原则，在一个新的对象里面使用一些已有的对象，使之成为新对象的一部分;新的对象通过这些向对象的委派达到复用已有功能的目的。这个设计原则有另一个简短的表述:要尽量使用合成/聚合，尽量不要使用继承。

6.迪米特法则，又叫做最少知识原则，就是说，一个对象应当对其他对象有尽可能少的了解。迪米特法则最初是用来作为面向对象的系统设计风格的一种法则，每一个软件单位对其他的单位都只有最少的知识，而且局限于那些本单位密切相关的软件单位。就是说，如果两个类不必彼此直接通信，那么这两个类就不应当发生直接的相互作用，如果其中的一个类需要调用另一个类的某一个方法的话，可以通过第三者转发这个调用。

7.单一职责原则，就一个类而言，应该仅有一个引起它变化的原因，如果你能想到多于一个的动机去改变一个类，那么这个类就具有多于一个的职责。应该把多于的指责分离出去，分别再创建一些类来完成每一个职责。

做好了准确详细的软件需求分析，并且遵循好的软件设计原则，这样做出的软件设计的质量才会有保证，有较高的健壮性和可行性，从而使最终设计出的软件能够更好的解决实际问题。