论如何提高软件测试质量

学院：软件学院

专业：软件工程

学号：14126164

姓名：杨开欣

摘要：随着软件测试受关注程度越来越高，如何采用技术手段有效提高软件测试质量就成了软件测试领域的一个重要课题。本文从软件测试的基本概念开始，对如何以软件测试性设计为中心、合理运用软件测试技术来提升软件测试质量提出了自己的看法。

关键词：软件测试；测试设计；测试质量

1 软件过程的划分

1.1 软件工程的介绍

软件开发过程可分为：需求，实际，实现和测试 4 个阶段，在开发大型软件系统的漫长过程中，面对纷繁复杂的各种现实情况，人的主观认识和客观现实是之间往往存在差距，开发过程中各类人员之间的交流和配合也往往并不是尽善尽美，所以，在软件生存周期的各个阶段都有可能产生差错。软件测试是为了发现程序中的错误而执行程序的过程。具体的说，软件测试是根据开发个阶段的规格说明和程序的内部结构而精心设计出一批测试用例，并利用测试用例来运行程序，发现程序错误的过程。

1.2 软件质量的介绍

软件产品的质量取决于软件开发过程，软件测试作为软件生存期中的一个重要阶段，受重视程度越来越高。软件测试是保证软件质量和可靠性的关键步骤，也是用来验证软件是否能够完成所期望功能的唯一有效的方法。测试已不仅仅局限于软件开发中的一个阶段，它已开始贯穿整个软件开发过程，进行测试的时间越早，整个软件开发成本下降就越多。大量统计表明，软件测试的工作量往往占到软件开发总量的 40%以上，在极端的情况下，甚至可能高达软件工程其它步骤成本总和的三至五倍，其目的是尽可能的提高软件产品的质量和可靠性。

2 软件测试常见的误区

2.1 软件开发完成后进行软件测试

软件项目要经过以下几个阶段：需求分析，概要设计，详细设计，软件编码，软件测试，软件发布。据此，认为软件测试只是软件编码后的一个过程。这是不了解软件测试周期的错误认识。软件测试是一个系列过程活动，包括软件测试需求分析，测试计划设计，测试用例设计，执行测试。因此，软件测试贯穿于软件项目的整个生命过程。在软件项目的每一个阶段都要进行不同目的和内容的测试活动，以保证各个阶段的正确性。

2.2 软件发布后如果发现质量问题，那是软件测试人员的错误

这种认识很打击软件测试人员的积极性。软件中的错误可能来自软件项目中的各个过程，软件测试只能确认软件存在错误，不能保证软件

没有错误，因为从根本上讲，软件测试不可能发现全部的错误。从软件开发的角度看，软件的高质量不是软件测试人员测出来的，是靠软件生命周期的各个过程中设计出来的。

2.3 软件测试要求不高，随便找个人就可以

很多人都认为软件测试就是安装和运行程序，点点鼠标，按按键盘的工作。这是由于不了解软件测试的具体技术和方法造成的。随之软件工程学的发展和软件项目管理经验的提高，软件测试已经形成了一个独立的技术学科，演变成一个具有巨大市场需求的行业。

2.4 软件测试是测试人员的事情，与程序员无关开发和测试是相辅相成的过程

需要软件测试人员、程序员和系统分析师等保持密切的联系，需要更多的交流和协调，以便提高测试效率。另外，对于单元测试主要应该由程序员完成，必要时测试人员可以帮助设计测试样例。对于测试中发现的软件错误，很多需要程序员通过修改编码才能修复。程序员可以通过有目的的分析软件错误的类型、数量，找出产生错误的位置和原因，以便在今后的编程中避免同样的错误，积累编程经验，提高编程能力。

3 提高测试质量的方法

3.1 采用测试性设计技术

软件测试是目前用来验证软件是否能够完成所期望的功能的唯一有效的方法。但是在测试的实施过程中，由于种种原因导致测试的难度相当大，甚至出现了无法测试的情形。为了提高软件的可测试性，我们在软件设计时应当遵循测试性设计原则，通过改变设计或代码、为软件增加专门测试结构等方法来提高软件的可测试性。

（1）测试驱动设计。这种设计就是直接把软件需求变成测试代码。在确定软件测试性能要求的基础上优先编写测试代码。先写验收测试，再写单元测试，并在开发过程中不断修正。其中具有以下本质优势，第一 TDD 根据客户需求编写测试用例，对功能的过程和接口都进行了设计，而且这种从使用者角度对代码进行的设计通常更符合后期开发的需求。因为关注用户反馈，可以及时响应需求变更，同时因为从使用者角度出发的简单设计，也可以更快地适应变化。第二，出于易测试和测试独立性的要求，将促使我们实现松

耦合的设计，并更多地依赖于接口而非具体的类，提高系统的可扩展性和抗变性。而且 TDD 明显地缩短了设计决策的反馈循环，使我们几秒或几分钟之内就能获得反馈。第三，将测试工作提到编码之前，并频繁地运行所有测试，可以尽量地避免和尽早地发现错误，极大地降低了后续测试及修复的成本，提高了代码的质量。在测试的保护下，不断重构代码，以消除重复设计，优化设计结构，提高了代码的重用性，从而提高了软件产品的质量。第四，TDD 提供了持续的回归测试，使我们拥有重构的勇气，因为代码的改动导致系统其他部分产生任何异常，测试都会立刻通知我们。完整的测试会帮助我们持续地跟踪整个系统的状态，因此我们就不需要担心会产生什么不可预知的副作用了。

（2）每个操作对应一个方法，使方法小型化。使用小型化方法说明和重载带缺省方法参数的方法，使得测试中调用这些方法变的很容易。

（3）显示与控制分离。把代码移到 GUI 视图的外面，各种 GUI 动作就能成了模型上的简单方法调用。这样，在修改程序功能不会影响视图，同时通过方法调用测试功能也比间接地测试功能更容易。

（4）对于可能要作为参数的类，做一个接口。用接口说明外部程序组件或在需要时改变接口形成一个空类作为参数传入。

3.2 选择合适的测试管理模型

模型是系统功能的形式化或半形式化的表示，支持输入状态组合的系统枚举。基于模型的测试主要考虑系统的功能，可以认为是功能测试的一种。测试模型体现了被测试系统的最本质的功能关系。而且要比系统本身更易于开发和分析。一个可测试的模型要能提供足够的信息用来产生测试用例。所以可测试的模型必须满足以下要求：

（1）必须是某种测试实现的完全准确的反映，模型必须表示要检查的所有特征；

（2）是对细节的抽象；

（3）可以表示所有事件和所有的动作；

（4）可以表示系统的各种状态，以便由可知的方法来确定已达到或没有达到什么状态。

4 结论

基于以上原则，我相信能在一定程度上保证测试的质量。在社会高速发展和人才济济的今天，肯定还会发展处更好的方法来保证软件测试的成果。我们要做的是不断学习，不断改进自己对测试的认识和技术，这样才能更好的做出趋于完美的测试，保证测试的质量，进而保证开发出客户满意的质量的软件。