**软件测试内容全貌——全景图 (1)**

在G.J.Myers的经典著作《软件测试之艺术》（The Art of Software Testing）中，给出了测试的定义：“**程序测试是为了发现错误而执行程序的过程**”。这个定义，被业界所认可，经常被引用。除此之外，G.J.Myers还给出了与测试相关的三个重要观点，那就是：

1. 测试是为了证明程序有错，而不是证明程序无错误；
2. 一个好的测试用例是在于它能发现至今未发现的错误；
3. 一个成功的测试是发现了至今未发现的错误的测试。

实际上，这里暗示了“软件测试”在不同侧面上的含义，也就决定了对软件测试不同的定义和不同的理解。根据作者多年的经验和理解，软件测试的不同视野，概括为如下5类：

* 软件测试的狭义论和广义论——静态和动态的测试
* 软件测试的辨证论——正向思维和反向思维
* 软件测试的风险论——测试是评估
* 软件测试的经济学观点——为盈利而测试
* 软件测试的标准论——验证和确认

**1. 软件测试的狭义论和广义论**

G.J.Myers所给出了测试定义——“**程序测试是为了发现错误而执行程序的过程**”，实际是一个狭义的概念，因为他认为测试是**执行程序**的过程，也就是传统意义上的测试——在代码完成后，通过运行程序来发现程序代码或软件系统中错误。但是，这种意义上的测试是不能在代码完成之前发现软件系统需求、发现设计上的问题，把需求、发现设计上的问题遗留到后期，这样就会可能造成设计、编程的部分返工。增加软件开发的成本、延长开发的周期等。需求阶段和设计阶段的缺陷产生的放大效应会加大。这非常不利于保证软件质量。这种狭义论是受软件开发瀑布模型影响。

正是为了更早地发现问题，所以将测试延伸到需求评审、设计审查活动中去，也就是将“软件质量保证”的部分活动归为测试活动。实际上，在软件开发实际操作中，常常将软件测试和质量保证——这两种努力（efforts）合并起来。

延伸后的软件测试，被认为是一种软件测试的广义概念。这就引出软件测试的两个概念“静态测试”和“动态测试”，这样就由静态测试和动态测试构成**一个全过程的、完整的软件测试，**而且静态测试显得更为重要**。**

**2.软件测试的辨证论**

G.J.Myers的第2个观点“测试是为了证明程序有错，而不是证明程序无错误”，引出了软件测试的另外一个争论，软件测试究竟是证明所有软件的功能特性是正确的呢？还是其反向思维——对软件系统进行各种试探和攻击，找出软件系统中不正常或不工作的地方呢？从我个人理解，这两个方面都有一定道理，前者（证明所有软件的功能特性是正确的）是从质量保证的角度来思考软件测试，后者（证明程序有错）从软件测试的直接目标和测试效率来思考，两者应该相辅相成。在后者的思想背景下，我们认为，测试不是为了证明所有的功能可以正常工作，恰恰相反，测试就是为了找出那些不能正常工作、不一致性的地方。也就是说，测试的一般工作就是发现缺陷 (detect bug)，即在软件开发过程中，分析、设计与编码等工作都是建设性的，而测试是带有“破坏性”的工作。

对于不同的应用领域，两者的比重是不一样的，如国防、航天、银行等软件系统，承受不了任何系统失效，因为一次系统的失效完全有可能导致灾难性的损失，所以强调前者以保证非常高的软件质量。而一般的软件服务应用则不同，强调后者，质量目标设置在“用户可接受水平”，不要过分追求质量，从而可以降低软件开发成本。作者建议，在我们实际操作中，可以分阶段实施不同的测试思想，在早期阶段集中在“证明程序有错”—— 发现Bug，后期集中在验证所有特性是否正常工作——降低风险。

    下面就是这两种观点的基本描述：

* **验证软件是验证软件是“工作的”**，以正向思维，针对软件系统的所有功能点，逐个验证其正确性。**其**代表人物是软件测试领域的先驱Dr. Bill Hetzel （代表论著《The Complete Guide to Software Testing》)。
* **证明软件是“不工作的”**，以反向思维方式，不断思考开发人员理解的误区、不良的习惯、程序代码的边界、无效数据的输入以及系统的弱点，试图破坏系统、摧毁系统，目标就是发现系统中各种各样的问题。其代表人物就是上面多次提到的G.J.Myers。他强调，一个成功的测试必须是发现Bug的测试，不然就没有价值。

**3.软件测试的风险论**

测试被定义为“对软件系统中潜在的各种风险进行评估的活动”，这就是软件测试的风险论。软件测试自身的风险性是大家公认的，**测试的覆盖度不能做到100％**。测试的这种风险定义一方面源于这层含义，另外软件测试的标准有时不清楚，“软件规格说明书（Specification/ Spec）”是其中的一个标准，但也不是唯一的，因为Spec中有些内容完全有可能是错误的。所以，我们常常强调**软件测试人员应该站在客户的角度去进行测试，除了发现程序中的错误，还要发现需求定义的错误、设计上的缺陷**，可以针对Spec 去报Bug。但是，测试在大多数时间/情况下,是由工程师完成，而不是客户自己来做，所以又怎么能保证工程师和客户想得一样呢？

有人把开发比作打靶，目标明确，就是按照Spec 去实现系统的功能。而把测试比作捞鱼，目标不明确，自己判断哪些地方鱼多，就去哪些地方捞；如果只捞大鱼（严重缺陷），网眼就可以大些、撒网区域相对比较集中（测试点集中在主要功能-major features）。如果想把大大小小的鱼捞上来，网眼就要小、普遍撒网，不放过任何一块区域（测试点遍及所有功能——all features）。

在“风险”论的框架下，软件测试可以被看作是一个动态的监控过程，对软件开发全过程进行检测，随时发现不健康的征兆，发现问题、报告问题，并重新评估新的风险，设置新的监控基准，不断地持续下去，包括回归测试。这时，软件测试可以完全看作是软件质量控制的过程。

对应这种观点，产生基于风险的测试策略，首先评估测试的风险，功能出问题的概率有多大？哪些是用户最常用的20％功能——Pareto原则（也叫80/20原则）？如果某个功能出问题，其对用户的影响有多大？然后根据风险大小确定测试的优先级。优先级高的测试，优先得到执行，一般来讲，针对用户最常用的20％功能（优先级高）的测试会得到完全执行，而低优先级的测试（另外用户不经常用的80％功能）就不是必要的，如果时间或经费不够，就暂时不做或少做。

**4.软件测试的经济学观点**

“一个好的测试用例是在于它能发现至今未发现的错误”，体现了软件测试的经济学观点**。**实际上**，**软件测试经济学问题至今仍是业界关注的问题之一。经济学的核心就是要盈利，盈利的基础就是要有一个清楚的商业性目标。同样，商业性目标是否正确，直接决定了企业是否盈利的结果。多数情况下，软件测试是在公司内的执行。正是公司的行为目的，决定了软件测试含义或定义的经济性一面。正如，对软件质量的定义不仅仅局陷于“和客户需求的一致性、适用性”，而且要增加其它的要求——“预算内、按时发布、易于维护”。

软件测试也一样，要尽快尽早地发现更多的缺陷，并督促和帮助开发人员修正缺陷。原因很简单：平均而言，**如果在需求阶段修正一个错误的代价是1，那么，在设计阶段就是它的3～6倍，在编程阶段是它的10倍，在内部测试阶段是它的20～40倍，在外部测试阶段是它的30～70倍，而到了产品发布出去时，这个数字就是  40～ 1000倍。修正错误的代价不是随时间线性增长，而几乎是呈指数级增长的**。

**5. 软件测试的标准论**

        如果从标准论来看软件测试，可以定义为软件测试就是“验证（Verification）”和“有效性确认（Validation）”活动构成的整体，即软件测试 = V&V。

 “验证”是检验软件是否已正确地实现了产品规格书所定义的系统功能和特性。验证过程提供证据表明软件相关产品与所有生命周期活动的要求（如正确性、完整性、一致性、准确性等）相一致。相当于，以Spec为标准进行软件测试活动，验证软件产品和Spec的一致性。

“有效性确认”是确认所开发的软件是否满足用户真正需求的活动。相当于，保持对软件需求定义、设计的怀疑，一切从客户出发，理解客户的需求，发现需求定义和产品设计中的问题。这主要通过各种软件评审活动来实现。

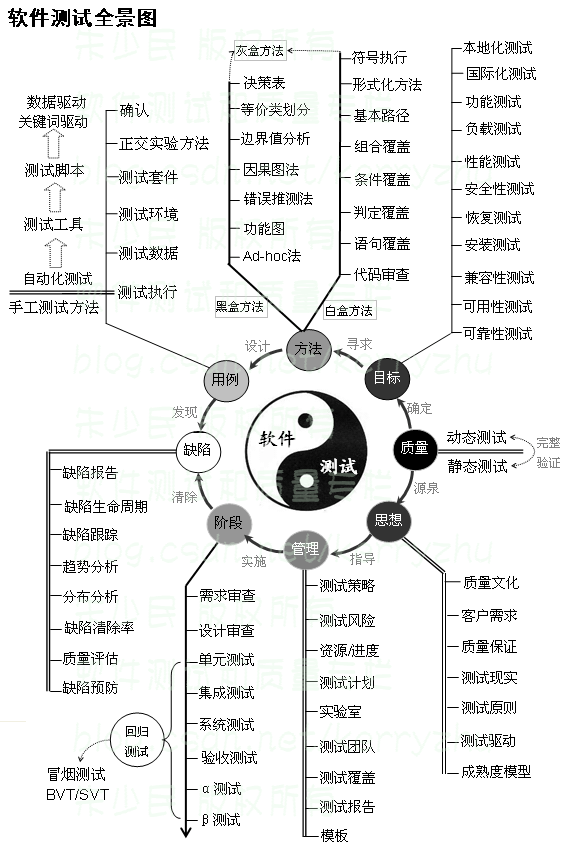
需要说明的是，软件测试的对象是产品（包括阶段性产品，如市场需求说明书、产品规格说明书、技术设计文档、数据字典、程序包、用户文档等），而质量保证和管理的对象集中在软件开发的标准、流程和方法等。

**究竟什么是软件测试呢？综上所述，软件测试的定义为：**

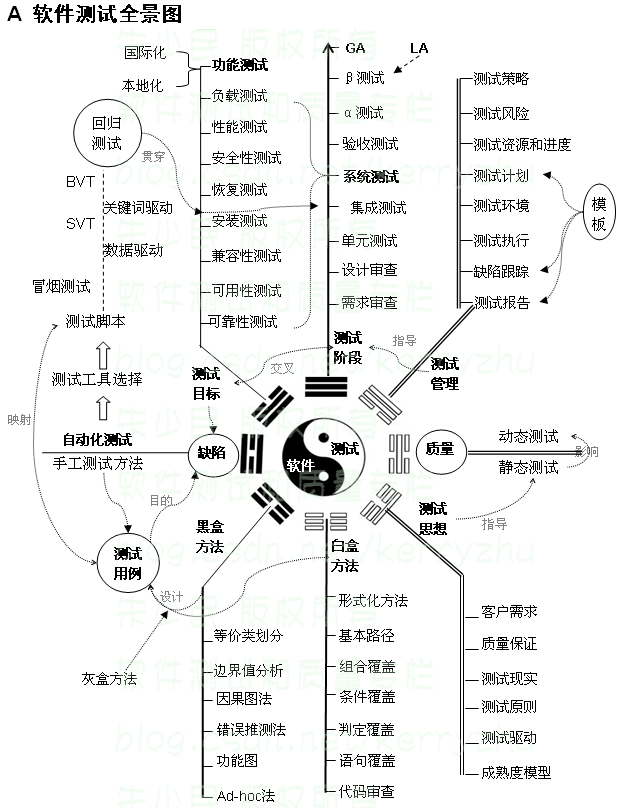
**软件测试是贯穿整个软件开发生命周期、对软件产品（包括阶段性产品）进行验证和确认的活动过程，**

**其目的是尽快尽早地发现在软件产品中所存在的各种问题——与用户需求、预先定义的不一致性。**

**通过全景图**，想使大家对软件测试一目了然。自己做了两张不同的全景图，自己还不能满意，特呈现出来，请大家多提意见，从而构造一完美的软件测试全景图。  
**【全景图一】  
        思路更清楚。**一方面，从质量管理的思想出发，定义测试的目标和测试的范围，然后通过相应的测试方法实现测试目标。这些方法自然被应用于测试用例的设计，而设计出来的测试用例被执行，而执行的手段有手工测试和自动化测试。设计测试用例的目的，就是为了更快、更全面地发现缺陷。另一方面，测试的管理思想也应源于客户的需求、源于组织的质量方针。测试管理要覆盖整个测试生命周期中的各个阶段，每个阶段都会涉及缺陷的报告、跟踪和分析。



**【全景图二】** 这是最初的草稿，基本思路和上面接近，可能更灵活些，而且试图更想说明测试用例、测试脚本和缺陷等之间的关系。理想的情况就是要建立需求、测试用例和缺陷之间的映射关系。也试图通过一些虚线来描述测试管理、测试阶段和测试目标等之间的关心，包括其中回归测试的概念。



**基于过程的软件测试全景图 (2)**

# 基于过程的软件测试全景图，是对基于内容的[软件测试内容全貌——全景图](http://blog.csdn.net/KerryZhu/archive/2007/05/23/1623069.aspx)(1) 的补充，从而对软件测试有一个较完整的描述。借助这张全景图，更好理解从需求、设计验证开始直至产品发布的整个测试过程，以及慢慢体会如何做好测试工作的每一个环节，不漏过任何一个环节，包括测试项目背景的掌控、沟通等等。  http://p.blog.csdn.net/images/p_blog_csdn_net/kerryzhu/209967/o_Software%20Test%20Full%20Panoramic%20Chart%203.gif

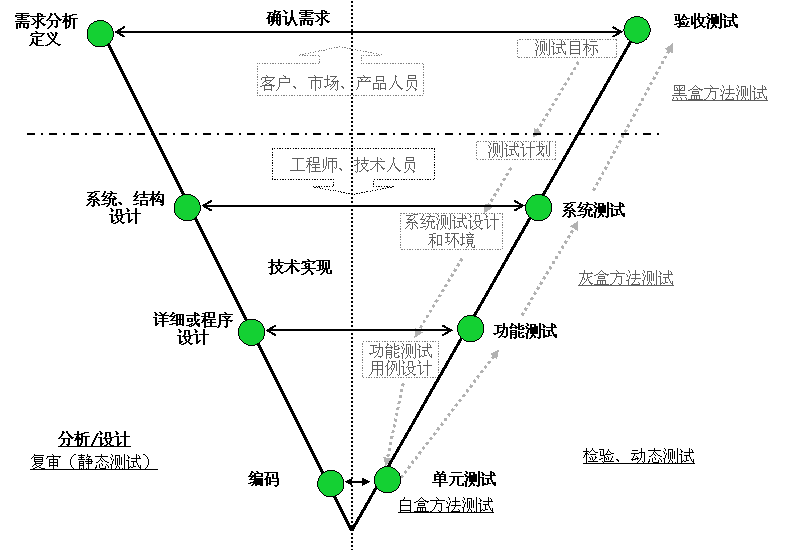
# V模型，我的完整诠释

万事开头难，第一回起头自然比较难，我选择了“V模型，我的完整诠释”作为开始。因为，**软件测试的思想方法是建立在软件开发过程模型（思想）基础之上**，例如测试驱动开发来源于敏捷开发思想。在这里，也是假定V模型是大家更好理解软件测试思想和方法的基础。

现在谈V模型，是否落后于时代？不一定，实际许多软件过程思想是相通的，例如迭代模型、增量模型和螺旋模型都可以归为“分阶段开发”思想这一类。极限编程（XP）对于现在Internet服务模式的软件开发很有效，也只适合软件开发的小团队。V模型适合企业级的软件开发，它更清楚地揭示了软件开发过程的特性及其本质。

V模型是在快速应用开发 (RAD，Rap Application Development)模型基础上演变而来，由于将整个开发过程构造成一个V字形而得名。**V模型强调软件开发的协作和速度，将软件实现和验证有机地结合起来，在保证较高的软件质量情况下缩短开发周期**。

下面通过对这种模型的水平和垂直的关联和比较分析，理解软件开发和测试的关系，理解V模型具有**面向客户、效率高、质量预防意识**等特点，能帮助我们建立一套更有效的、更具有可操作性的软件开发过程。

**图**

**1．** **从水平对应关系看**

左边是设计和分析，是软件设计实现的过程，同时伴随着质量保证活动——审核的过程，也就是静态的测试过程；右边是对左边结果的验证，是动态测试的过程，即对设计和分析的结果进行测试，以确认是否满足用户的需求。如：

         需求分析和功能设计对应验收测试，说明在做需求分析、产品功能设计的同时，测试人员就可以阅读、审查需求分析的结果，从而了解产品的设计特性、用户的真正需求，确定测试目标，可以准备用例(Use Case)并策划测试活动。

         当系统设计人员在做系统设计时，测试人员可以了解系统是如何实现的，基于什么样的平台，这样可以设计系统的测试方案和测试计划，并事先准备系统的测试环境，包括硬件和第三方软件的采购。因为这些准备工作，实际上是要花去很多时间。

         当设计人员在做在做详细设计时，测试人员可以参与设计，对设计进行评审，找出设计的缺陷，同时设计功能、新特性等各方面的测试用例，完善测试计划，并基于这些测试用例以开发测试脚本。

         在编程的同时，进行单元测试，是一种很有效的办法，可以尽快找出程序中的错误，充分的单元测试可以大幅度提高程序质量、减少成本。

从中可以看出，V模型使我们能清楚地看到质量保证活动和项目同时展开, **项目一启动，软件测试的工作也就启动了,避免了瀑布模型所带来的误区——软件测试是在代码完成之后进行。**

**2．** **从垂直方向看**

水平虚线上部表明，其需求分析、定义和验收测试等主要工作是面向用户，要和用户进行充分的沟通和交流，或者是和用户一起完成。水平虚线下部的大部分工作，相对来说，都是技术工作，在开发组织内部进行，主要是由工程师、技术人员完成。

从垂直方向看，越在下面，白盒测试方法使用越多，到了集成、系统测试，更多是将白盒测试方法和黑盒测试方法结合起来使用，形成灰盒测试方法。而在验收测试过程中，由于用户一般要参与，使用黑盒测试方法。

# 软件测试的目标

软件缺陷的产生主要是由软件产品的特点和开发过程决定的，如软件的需求经常不够明确，而且需求变更频繁，开发人员不太了解软件需求，不清楚应该“做什么”和“不做什么”，常常做不合需求的事情，产生的问题最多。同时，软件竞争非常厉害，技术日新月异，使用新的技术，也容易产生问题。而且对于不少软件企业，“争取时间上取胜”常常是其主要市场竞争策略之一，实现新功能、很酷的功能，被认为比质量更为重要，导致日程安排很紧，需求分析、设计等投入的时间和精力远远不够，也是产生软件错误的主要原因之一。

软件错误产生的原因可能还有其他一些原因，例如，软件设计文档不清楚，文档本身就存在错误，导致使用者产生更多的错误。还有沟通上的问题、开发人员的态度问题以及项目管理问题等。《微软开发者成功之路(之一)》概括为有以下七项主要原因：

1. **项目期限的压力**
2. **产品的复杂度**
3. **沟通不良**
4. **开发人员的疲劳、压力或受到干扰**
5. **缺乏足够的知识、技能和经验**
6. **不了解客户的需求**
7. **缺乏动力**

这些原因，会引起下列主要领域的主要错误（缺陷）：

* 需求规格说明书（Requirement Specification or Functional Specification ）包含错误的需求、或漏掉一些需求, 或没有准确表达客户所需要的内容；
* 需求规格说明书中有些功能不可能或无法实现的；
* 系统设计(System Design)中的不合理性；
* 程序设计中的错误、程序代码中的问题，包括错误的算法、复杂的逻辑等。

若能及早排除软件开发中的错误，有效的减少后期工作的麻烦，就可以尽可能的避免付出高昂的代价，从而大大提高系统开发过程的效率。

软件测试的目标，就是为了**更快、更早**地将软件产品或软件系统中所存在的各种问题找出来，并促进开发各类人员尽快地解决问题，最终及时地向客户提供一个**高质量**的软件产品，使软件系统更好地满足用户的需求，同时满足软件组织自身的要求：

**1. 用户的需求**

* 能正常使用全部所需要的功能
* 功能强大，而且界面美观、易用、好用
* 内容健康，有益于生活和工作
* 用户的数据安全、受保护和兼容
* 及时得到新的产品或得到更完美的软件服务
* 软件可靠性很高，使用软件服务没有时间障碍

**2. 软件企业的需求**

* 软件质量是市场竞争的需要，质量好的软件是留住客户的最关键的手段之一，软件企业也必须依靠质量，才能立于不败之地；
* 高质量的软件可以大大降低“质量问题产生的成本”，增加公司的盈利；
* 软件已是国际化的市场，质量是进入国际市场的一个关键门坎；
* 容易维护、移植和扩充，以扩大市场或适应环境的变化。

**这些要求的满足，最终体现在软件产品的质量上：**

* 功能性，软件所实现的功能达到它的设计规范和满足用户需求的程度；
* 可用性，对于一个软件，用户学习、操作、准备输入和理解输出所作努力的程度，如安装简单方便、容易使用、界面友好，并能适用于不同特点的用户，包括对残疾人、有缺陷的人能提供产品使用的有效途径或手段；
* 可靠性，用户使用的根本，在规定的时间和条件下，软件所能维持其正常的功能操作、性能水平的程度；
* 性能，在指定条件下，用软件实现某种功能所需的计算机资源（包括内存大小、CPU占用时间等）的有效程度；
* 容量，系统的接受力、容纳或吸收的能力、或某项功能的最大量或最大限度，有时需要确定系统的特定需求的所能容纳的最大量、所能表现的最大值。如Web系统能承受多少并发用户访问、会议系统可以承受的与会人数等；
* 可测量性，系统某些特性可以通过一些量化的数据指标能描述其当前状态或理想状态；
* 可维护性，在一个运行软件中，当环境改变或软件发生错误时，进行相应修改所做努力的简易程度；可维护性取决于理解软件、更改软件和测试软件的简易程度，可维护性与灵活性密切相关。高可维护性对于那些经历周期性更改的产品或快速开发的产品很重要。
* 兼容性，软件从一个计算机系统或环境移植到另一个系统或环境的容易程度，或者是一个系统和外部条件共同工作的容易程度。兼容性表现在多个方面，如系统的软件和硬件的兼容性、软件的不同版本的系统、数据的兼容性；
* 可扩展性，指将来功能增加、系统扩充的难易程度或能力。

# 软件测试的十大原则

原则是最重要的，方法应该在这个原则指导下进行。软件测试的基本原则是站在用户的角度，对产品进行全面测试，

尽早、尽可能多地发现Bug, 并负责跟踪和分析产品中的问题，对不足之处提出质疑和改进意见。

零缺陷(0-Bug) 是一种理念，足够好（Good-Enough）是测试的基本原则。在软件测试过程中，应注意和遵循的具体原则，可以概括为十大项：

1. **所有测试的标准都是建立在用户需求之上**。正如我们所知，软件测试的目标就是验证产品的一致性和确认产品是否满足客户的需求，所以**测试人员要始终站在用户的角度去看问题、去判断软件缺陷的影响**，系统中最严重的错误是那些导致程序无法满足用户需求的缺陷。
2. **软件测试必须基于“质量第一”的思想去开展各项工作**，当时间和质量冲突时，时间要服从质量。质量的理念和文化（如零缺陷的“第一次就把事情做对”）同样是软件测试工作的基础。
3. **事先定义好产品的质量标准**。有了质量标准，才能依据测试的结果对产品的质量进行正确的分析和评估，例如，进行性能测试前，应定义好产品性能的相关的各种指标。同样，测试用例应确定预期输出结果，如果无法确定测试结果，则无法进行校验。
4. **软件项目一启动，软件测试也就是开始**，而不是等程序写完，才开始进行测试。在代码完成之前，测试人员要参与需求分析、系统或程序设计的审查工作，而且要准备测试计划、测试用例、测试脚本和测试环境，测试计划可以在需求模型一完成就开始，详细的测试用例定义可以在设计模型被确定后开始。应当把“**尽早和不断地测试**”作为测试人员的座右铭。
5. **穷举测试是不可能的**。甚至一个大小适度的程序，其路径排列的数量也非常大，因此，在测试中不可能运行路径的每一种组合，然而，充分覆盖程序逻辑，并确保程序设计中使用的所有条件是有可能的。
6. **第三方进行测试会更客观，更有效。**程序员应避免测试自己的程序，为达到最佳的效果，应由第三方来进行测试。测试是带有 ”挑剔性” 的行为，心理状态是测试自己程序的障碍。同时对于需求规格说明的理解产生的错误也很难在程序员本人测试时被发现。
7. **软件测试计划是做好软件测试工作的前提。**所以在进行实际测试之前，应制定良好的、切实可行的测试计划并严格执行，特别要确定测试策略和测试目标。
8. **测试用例是设计出来的，不是写出来的，**所以要根据测试的目的，采用相应的方法去设计测试用例，从而提高测试的效率，更多地发现错误，提高程序的可靠性。除了检查程序是否做了应该做的事，还要看程序是否做了不该做的事；不仅应选用合理的输入数据，对于非法的输入也要设计测试用例进行测试。
9. **不可将测试用例置之度外，排除随意性。**特别是对于做了修改之后的程序进行重新测试时，如不严格执行测试用例，将有可能忽略由修改错误而引起的大量的新错误。所以，回归测试的关联性也应引起充分的注意，有相当一部分最终发现的错误是在早期测试结果中遗漏的。
10. **对发现错误较多的程序段，应进行更深入的测试。**一般来说，一段程序中已发现的错误数越多，其中存在的错误概率也就越大。错误集中发生的现象，可能和程序员的编程水平和习惯有很大的关系。