**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。

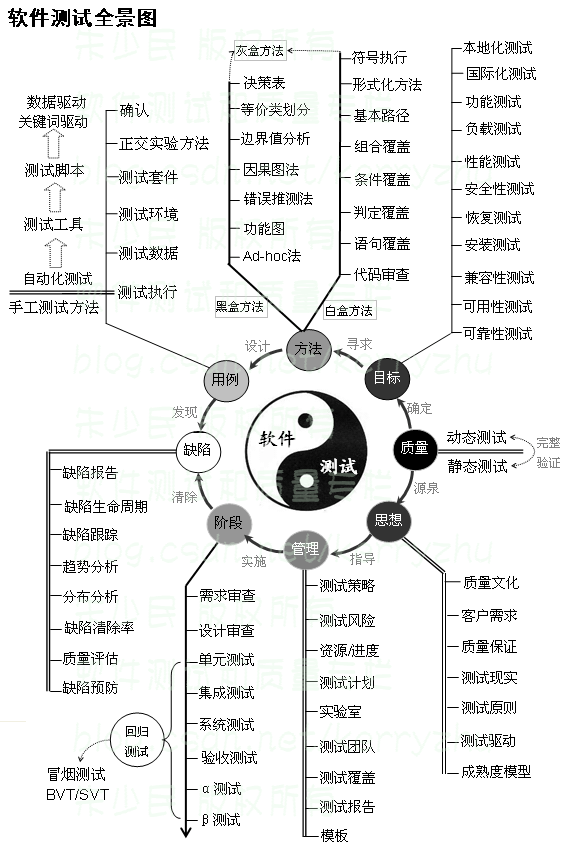
需要说明的是，软件测试的对象是产品（包括阶段性产品，如市场需求说明书、产品规格说明书、技术设计文档、数据字典、程序包、用户文档等），而质量保证和管理的对象集中在软件开发的标准、流程和方法等。

**究竟什么是软件测试呢？综上所述，软件测试的定义为：**

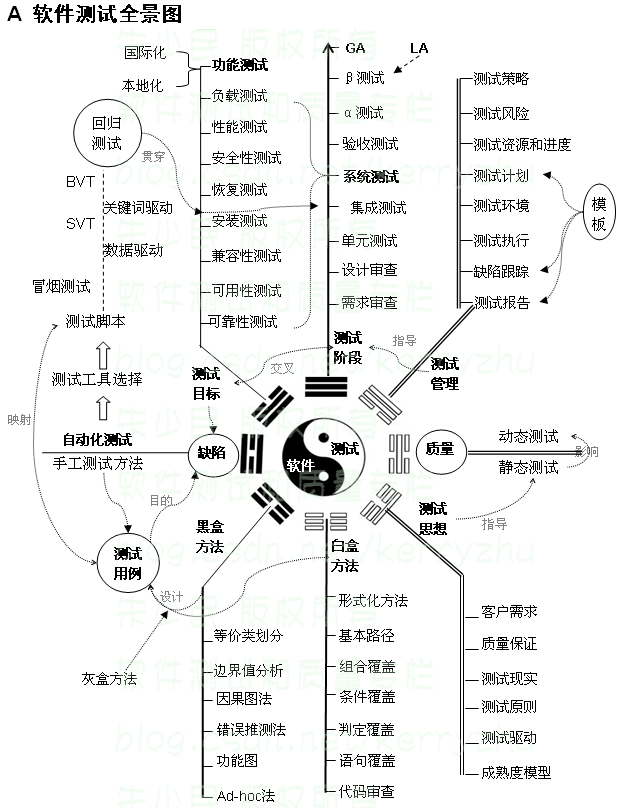
**软件测试是贯穿整个软件开发生命周期、对软件产品（包括阶段性产品）进行验证和确认的活动过程，**

**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。   
