一、软件测试的定义

软件测试是一个过程或一系列过程，用来确认计算机代码完成了其应该完成的功能，不执行其不该有的操作。

1．软件测试与调试的区别?

（1）测试是为了发现软件中存在的错误；调试是为证明软件开发的正确性。

（2）测试以已知条件开始，使用预先定义的程序，且有预知的结果，不可预见的仅是程序

是否通过测试；调试一般是以不可知的内部条件开始，除统计性调试外，结果是不可预见的。

（3）测试是有计划的，需要进行测试设计；调试是不受时间约束的。

（4）测试经历发现错误、改正错误、重新测试的过程；调试是一个推理过程。

（5）测试的执行是有规程的；调试的执行往往要求开发人员进行必要推理以至知觉的"飞跃

"。

（6）测试经常是由独立的测试组在不了解软件设计的条件下完成的；调试必须由了解详细

设计的开发人员完成。

（7）大多数测试的执行和设计可以由工具支持；调式时，开发人员能利用的工具主要是调

试器。

2．对软件测试的理解?

软件测试就是说要去根据客户的要求完善它.即要把这个软件还没有符合的或者是和客户要求不一样的，或者是客户要求还没有完全达到要求的部分找出来。

（1）首先要锻炼自己软件测试能力，包括需求的分析能力，提取能力，逻辑化思想能力，

即就是给你一个系统的时候，能够把整个业务流程很清晰的理出。

（2）学习测试理论知识并与你锻炼的能力相结合。

（3）想和做。想就是说你看到任何的系统都要有习惯性的思考；做就是把实际去做练习，

然后提取经验。

总结测试用例，测试计划固然重要，但能力和思想一旦到位了，才能成为一名合格的软件测

试工程师。

二、软件测试的分类

1．按照测试技术划分

（1）白盒测试：通过对程序内部结构的分析、检测来寻找问题。检查是否所有的结构及逻

辑都是正确的，检查软件内部动作是否按照设计说明的规定正常进行。--结构测试

（2）黑盒测试：通过软件的外部表现来发现错误，是在程序界面处进行测试，只是检查是

否按照需求规格说明书的规定正常实现。--性能测试

（3）灰盒测试：介于白盒测试与黑盒测试之间的测试。

2．按照是否让备测软件运行划分

（1）静态测试

（2）动态测试

3．按照开发阶段划分

（1）单元测试：模块测试，检查每个程序单元嫩否正确实现详细设计说明中的模块功能等。

（2）集成测试：组装测试，将所有的程序模块进行有序、递增的测试，检验程序单元或部

件的接口关系

（3）系统测试：检查完整的程序系统能否和系统（包括硬件、外设和网络、系统软件、支

持平台等）正确配置、连接，并满足用户需求。

（4）确认测试：证实软件是否满足特定于其用途的需求，是否满足软件需求说明书的规定。

（5）验收测试：按项目任务或合同，供需双方签订的验收依据文档进行的对整个系统的测

试与评审，决定是否接受或拒收系统。

4．按照测试实施组织划分

（1）开发方测试

（2）用户测试

（3）第三方测试

三、软件测试的原则

1．测试用例中一个必需部分是对预期输出或结果的定义；

2．程序员应当避免测试自己编写的程序；

3．编写软件的组织不应当测试自己编写的程序；

4．应该彻底检查每个测试的执行结果；

5．测试用例的编写不仅应当根据有效和预期的输入情况，也应当根据无效和未预料到的输

入情况；

6．检查程序是否“未做其应该做的”仅是测试的一半，测试的另一半是检查程序是否“做

了不应该做的”；

7．应避免测试用例用后既弃，除非软件本身就是一个一次性的软件；

8．计划测试工作时不应默许假定不会发现错误；

9．程序某部分存在更多错误的可能性，与该部分已发现错误的数量成正比；

10．软件测试是一项极富创造性、极具智力挑战性的工作。

四、测试用例的设计

1．测试用例的定义

（1）测试用例是为特定的目的而设计的一组测试输入、执行条件和预期的结果。

（2）测试用例是执行的最小实体。

2．特征：

（1）最有可能抓住错误的；

（2）不是重复的、多余的；

（3）一组相似测试用例中最有效的；

（4）既不是太简单，也不是太复杂。

3．设计测试用例的基本准则

测试用例的代表性、测试结果的可判定性、测试结果的可再现性。

五、黑盒测试

1．等价类划分法

①等价类划分法的设计方法：是把所有可能的输入数据，即程序的输入域划分成若干部分（子集），

然后从每一个子集中选取少量具有代表性的数据作为测试用例。

等价类是指某个输入域的子集合。在该子集合中各个输入数据对于揭露程序中错误都是等效的。并

合理地假定：测试某等价类的代表值就等于对这一类其他值的测试。

有效等价类：对于程序的规格说明来说是合理的、有意义的输入数据构成的集合

无效等价类：对软件规格说明而言，是无意义的、不合理的输入数据所构成的集合

等价类对于测试有两个重要的意义：完备性无冗余性

②等价类划分法的原则

（a）按照区间划分：　　一个有效等价类和两个无效等价类。

（b）按照数值划分：　　n 个有效等价类和一个无效等价类

（c）按照数值集合划分　一个有效等价类和一个无效等价类

（d）按照限制条件或规则划分：可确定一个有效等价类和若干个无效等价类

（e）细分等价类

③等价类划分法的步骤

（a）确定等价类

（b）建立等价类表，列出所有划分出的等价类

（c）从划分出的等价类中按以下的 3 个原则设计测试用例：

·为每一个等价类规定一个唯一的编号

·设计一个新的测试用例，使其尽可能多的覆盖尚未被覆盖的有效等价类，重复这一步，直到所有

的有效等价类都被覆盖为止；

·设计一个新的测试用例，使其仅覆盖一个尚未被覆盖的无效等价类，重复这一步，直到所有的无

效等价类都被覆盖为止。

④确定等价类的方法

（a）先考虑输入数据的类型（合法型和非法型）；

（b）再考虑数据范围（合法型中的合法区间和非法区间）；

（c）最后考虑输出结果，逆向设定输入。

2．边界值分析法

①边界值分析法就是对输入或输出的边界值进行测试

②特点：具有很强的发现程序错误的能力；测试用例来自等价类的边界；

③基本原理：故障往往发生在输入定义域和输出值域的边界上，而不是在其内部。

④方法：（a）首先应确定边界情况.

Y

（b）选取正好等于,刚刚大于或刚刚小于边界的值作为测试数据

c

⑤标准边界值：min、min+、nom、max-、max

健壮边界值： min、min+、nom、max-、max min- max+

d

⑥例：

ab

X

<xnom,ymin><xnom,ymin+><xnom,ymax><xnom,ymax->

有两个变量 x、y 的程序的输入域

<xmin,ynom><xmin+,ynom><xmax,ynom><xmax-,ynom><xnom,ynom>

⑦对于一个含有 n 个变量的程序，只让其中一个变量取极值，让其余的变量取正常值，被保留的变

量依次取 min、min+、nom、max-、max 值，对每个变量都重复进行。n 个变量的程序，边界值分析

测试程序会产生 4n+1 个测试用例。

3．决策表法

①概述：决策表法是黑盒测试方法中最为严格、最具有逻辑性的测试方法。

②什么时候使用?

程序输入输出比较多，输入之间、输出之间相互制约的条件比较多时，可以清楚地表达它们之间的

各种复杂关系。

③决策表通常由四部分组成：　　规则

条件桩 条件项

动作桩 动作项

条件桩： 列出问题的所有条件

条件项：针对条件桩给出的条件列出所有可能的取值

动作桩：给出问题规定的可能采取的操作

动作项：与条件项紧密相关，指出在条件项的各组取值情况下应采取的动作

规则：项中的每一列是一条规则，每一条规则是一组测试用例。

④决策表的化简

（a）合并：如果一个条件项(表中某列中的条件值)和另外一个条件项所产生的动作是相同的,且两

个条件项对应的每一行的值只有一个是不同的,则可以将其合并.合并的项除了不同值变成”不关心”

条目外,其余不变

（b）包含：如果两个条件项的动作是相同的，对任意条件 1 的值和条件 2 中对应的值，

如果满足：

– 如果条件 1 的值是 T(F),则条件 2 中的值也是 T(F).

– 如果条件 1 的值是-(不关心),则条件 2 中的值是 T,F,-,称条件 1 包含条件 2,条件 2 可以撤去.

– 重复 A,B 就可以得到精简的决策表.

Y

Y Y

Y

Y Y

N

-

-

N N

N

N N

N

Y N

-

√ √ √

√ √ √

合并　　包含

⑤构造决策表的步骤：

（a）确定规则的个数；

（b）列出所有的条件桩和动作桩；

（c）填入输入项；

（d）填入动作项,得到初始的决策表；

（e）对初始的决策表化简。

⑥决策表测试法的适用范围

（a）if-then-else 逻辑突出；

（b）输入变量之间存在逻辑关系；

（c）涉及输入变量子集的计算；

（d）输入和输出之间存在因果关系。

4．因果图方法

①概述：如果输入之间有关系，测试时必须考虑输入条件的各种组合，考虑适合于描述对于多种条

件的组合，相应产生多个动作的形式来设计测试用例，这就需要利用因果图。

因果图方法最终生成的就是判定表。适合于检查程序输入条件的各种组合情况。

②因果图法的基本思想： 首先从程序规格说明书的描述中,找出因(输入条件)和果(输出结

果或者程

序状态的改变),然后通过因果图转换为判定表,最后为判定表中的每一列设计一个测试用例.

③基本符号　原因　结果

通常在因果图中用 Ci 表示原因，用 Ei 表示结果，各结点表示状态，可取值“0”或“1”。“0”表示

某状态不出现，“1”表示某状态出现。

c1

C2

恒等： c1 为 1,则 e1 也为 1,否则 e1 为 0. 非： 若 c1 是 1,则 e1 为 0,否则 e1 是 1.

或： 若 c1 或 c2 或 c3 是 1,则 e1 是 1,若三者都不为 1,则 e1 为 0.

与： 若 c1 和 c2 都是 1,则 e1 为 1,否则若有其中一个不为 1,则 e1 为 0.

④约束：实际问题中,输入状态之间可能存在某些依赖关系.

E 约束(异)：a,b 最多有一个可能为 1,不能同时为 1.

I 约束(或)：a,b,c 中至少有一个必须为 1,不能同时为 0.

O 约束(惟一)： a 和 b 必须有一个且仅有一个为 1

R 约束(要求)：a 是 1 时,b 必须是 1,即 a 为 1 时,b 不能为 0

M 约束：对输出条件的约束,若结果 a 为 1,则结果 b 必须为 0.

⑤因果图生成测试用例的基本步骤

（a）找出原因和结果。（b）画出因果图。（c）增加约束。

（d）把因果图转化为判定表，并化简。

（e）把判定表的每一列拿出来作为依据，设计测试用例。

⑥例题

（a）原因： C1：第一个字符是 A； C2：第一个字符是 B；C3：第二个字符是一个数字

字找。

结果： E1：给出信息 L； E2：修改文件； E3：给出信息 M。

（b）因果图。

C1

~

E1

10

v

C2 E2

^

C3 E3

~

(c) 决策表。

1 2 3 4 5 6 7 8

C1 1 1 0 0 0 0

1 1

C2 0 0 1 1 0 0

1 1

C3 1 0 1 0 1 0

1 0

10 1 1 1 1 0 0

E1

E2

√ √ √

E3 √

√ √ √

不可能 √ √

AM BM C2 CM A3 B3

测试用例

A& B* X6 D* A5 B5

（d）设计测试用例

测试用例 1： 输入数据：A3 预期输出：修改文件

测试用例 2： 输入数据：AM 预期输出：给出信息 M

测试用例 3： 输入数据：B3 预期输出：修改文件

测试用例 4： 输入数据：B* 预期输出：给出信息 M

测试用例 5： 输入数据：C2 预期输出：给出信息 L

测试用例 6： 输入数据：CM 预期输出：给出信息 LM

⑦因果图法的优点：

（a）考虑了多个输入之间的相互组合、相互制约关系；

（b）能够帮助我们按一定步骤，高效率地选择测试用例，同时还能为我们指出，程序规格说明描述

中存在着什么问题。

六、白盒测试

1．白盒测试概述：白盒测试也称结构测试或逻辑驱动测试。

2．方法：程序结构分析；逻辑覆盖测试；基本路径测试。

3．原则：

（1）保证一个模块中所有独立路径至少被测试一次；

（2）所有逻辑值均需测试真（True）和假（False）两种情况；

（3）检查程序的内部数据结构，保证其结构的有效性；

（4）在取值上、下边界，即可操作范围内运行所有循环.

4．逻辑覆盖测试:主要是测试覆盖率，以程序内在逻辑结构为基础的测试。

6 种：语句覆盖 判断覆盖 条件覆盖 判定-条件覆盖 条件组合覆盖 路径测试.

①语句覆盖：在测试时，首先设计若干个测试用例，然后运行被测程序，使程序中的每个可执行语

句至少执行一次 。

·判定：整体 控制。 包括：a、单一条件判定；b、符合条件覆盖

·语句覆盖率：已执行的可执行语句占程序中可执行语句总数的百分比

②判定覆盖：设计足够多的测试用例，使程序中的每个判定至少都获得一次“真值”或“假值”。

③条件覆盖:构造一组测试用例，使得每一判定语句中每个逻辑条件的可能值至少满足一次。

满足条件覆盖的不一定满足判定覆盖，反之亦然。两者无直接关系。

④判定/条件覆盖：设计足够的测试用例，使得判定中每个条件的所有可能(真/假)至少出现一次，

并且每个判定本身的判定结果(真/假)也至少出现一次

⑤组合条件覆盖（MCC）：设计足够的测试用例，使得每个判定中条件的各种可能组合都至少出现一

次。

满足组合条件覆盖的测试用例是一定满足判定覆盖、条件覆盖和判定/条件覆盖。

⑥修正条件判定覆盖(MCDC)：需要足够的测试用例来确定各个条件能够影响到包含的判定的结果，

即要求满足两个条件。

七、静态测试

1．静态测试不实际运行软件，主要对软件的编程格式、结构等方面进行评估。可以有人工进行，也

可借助软件工具自动进行。

2．静态测试的方法

（1）代码检查：代码审查 代码走查 桌面检查 同行评分（略）

（2）代码审查：通常由 4 人组成，其中一人是协调人，一人是程序的编写者，其他人员通常是程序

的设计人员以及测试专家。

优点和作用：错误列表、高效、会后修正、增加修改错误清单、较早发现错误。

（3）代码走查：为测试员的人会带着一些书面的测试用例参加会议

（4）桌面检查：（a）完全没有约束（b）开发人员测试自己的程序（c）没有展示自己能力，缺乏良

好的效应。（效果远远逊于代码审查和代码走查）

3．静态结构分析：主要是以图形的方式表现程序的内部结构。

4．代码质量度量：功能性 可靠性 可用性 |有效性 可维护性 轻便性

八、单元测试

1．单元测试的定义

单元测试又称模块测试，是最小单位的测试，其依据是详细设描述，对模块内所有重要的控制路径

设计测试用例，以便发现模块内部的错误。

单元测试多采用白盒测试技术

2．单元测试的对象

·结构化程序，单元测试的单元是指单个子程序、函数或过程

·面向对象程序，单元测试的单元是指类或方法（通常为类）。

3．单元测试的目的

将模块的功能与定义模块的功能规格说明或接口规格说明进行比较，揭示出模块与其规格说明之间

存在的矛盾。

4．单元测试的人员:开发人员

5．单元测试的针对的问题

（1）模块接口： 检查进出程序单元的数据流是否正确。

（2）局部数据结构：必须测试模块内部的数据能否保持完整性。

（3）边界条件测试：主要检查临界数据是否正确处理。

（4）独立路径测试：发现由于不正确的判定或不正常的控制流而产生的错误。

（5）出错处理：要求能预见出错的条件，并设置适当的处理对象，保证其路径的正确

性。

6．单元测试的流程

计划单元测试　设计单元测试　执行单元测试　评估单元测试

7. 计划单元测试

（1）驱动模块(Drive)：用来模拟被测试模块的上一级模块，相当于被测模块的主程序。它接收数

据，将相关数据传送给被测模块，启动被测模块，并打印出相应的结果。

（2）桩模块(Stub)：用来模拟被测模块工作过程中所调用的模块。它们一般只进行很少的数据处理。

8．设计单元测试

（1）需要的信息

·模块的规格说明：模块的输入和输出以及模块的功能。

·模块的源代码。

（2）测试用例的设计方法

·模块测试总体上是面向白盒测试的（静态、动态）

·后续测试针对较大的元素不易进行白盒测试。

·后续测试着眼于发现其他类型的错误，不一定与程序逻辑结构有关。

·使用一种或多种白盒测试方法分析模块的逻辑结构，然后使用黑盒测试方法对照模块的规

格说明

补充测试用例。

9.执行单元测试

（1）设置测试环境

（2）将测试环境初始化

（3）执行测试过程。

10.评估单元测试

（1）测试完备性评估

(2) 代码覆盖率评估

九、集成测试

1．集成测试的定义

集成测试又称组装测试，集成测试是在单元测试的基础上，将所有模块按照设计要求组装成子系统

或系统进行的测试活动。

2．集成测试的目的

确保各单元组合在一起后能够按既定意图协作运行，并确保增量的行为正确，所测试的内容包括单

元间的接口以及集成后的功能。

3．集成测试的层次

（1）模块内集成测试

（2）子系统内集成测试

（3）子系统间集成测试

4．集成测试的流程

5．集成测试的方法

（1）静态测试：只要指对概要设计的测试。

（2）动态测试：以黑盒测试为主，需要了解内部细节时结合白盒测试

6．集成测试策略

（1）非增量式集成：对所有模块进行个别的单元测试后，按照程序结构图将各模块连接起来，把连

接后的程序当作一个整体进行测试。

关键模块的特征：

①满足某些软件需求；

②在程序的模块结构中位于较高的层次（高层控制模块）；

③较复杂、较易发生错误；

④有明确定义的性能要求。

（2）增量式集成：逐次将未曾集成测试的模块和已经集成测试的模块（或子系统）结合成程序包，

再将这些模块集成为较大系统，在集成的过程中边连接边测试，以发现连接过程中产生的问题。

方法：

① 自顶向下增量式测试：深度优先、广度优先。

② 自底向上增量式测试

③混合增量式测试

7．不同集成测试方法的比较

十、系统测试

1．系统测试的目的

将系统或程序与其初始目标进行比较，这意味着系统测试并不局限于系统，系统测试是一个

试图说明程序作为一个整体是如何不满足其目标的过程。如果产品没有一组书面的、可度量

的目标，系统测试也无法进行。

2．系统测试的类型

能力测试，容量测试，强度测试，易用性测试，安全性测试，性能测试，存储测试，配置测

试，兼容性/配置/转换测试，安装测试，可靠性测试，可恢复性测试，适用性测试，文档测

试，过程测试

（1）能力测试

·判断目标文档提及的每一项能力（以区别功能测试中的‘功能’）是否都确实已经实现。

·通常是通过人工检查目标文档中定义了“要做什么”。

（2）容量测试

·是程序经受大容量数据的检验，目的是证明程序不能处理目标文档中规定的数据容量。

·容量测试需要大量的资源，不可进行过多。

·如何使操作系统的作业队列达到饱和容量。

（3）强度测试

·使程序承受高负载或强度的检验。所谓高强度是指在很短的时间间隔内达到的数据或操作的数量峰值。（要与容量测试相区分）

·强度测试涉及时间因素，适用于在可变负载下运行的程序以及交互式程序、实时程序和过程控制程序。基于 Web 的应用程序也是最常接受强度测试的软件之一。

如，1.在很短的时间内是操作系统的作业队列达到峰值;

2.web 应用程序要处理一定容量的并发用户。

注：强度测试是对强度的界定很重要。

（4）易用性测试

·每个用户界面是否都根据用户的智力、教育程度和环境要求进行了调整?

·程序的输出是否有意义、不模糊且无计算机杂乱信息?

·错误诊断信息是否直接，非计算机专业用户是否能够理解（这要求对错误进行精确的预测和详细的分类）？

·整体的用户界面是否在语法、惯例、语义、格式、风格和缩写等方面展现出了相当程度的完整性、一致性和同一性?

·系统是否包含过多或不太可能用到的选项?

·对于所有输入，系统是否返回了即时确认信息?

·程序是否易于使用？如区分大小写的要求用户是否清楚，不同层次菜单之间的浏览是否容易等。

（5）安全性测试

·设计测试用例来突破程序安全检查。例如，可以设计测试用例来规避操作系统的内存保护机制、破坏数据库管理系统的数据安全机制等。

·常用的测试用例设计方法是研究类似系统中已知的安全问题，然后生成测试用例，暴露被测系统中的类似问题

·基于 Web 的应用程序常常比绝大多数程序所需的安全测试级别更高，对于电子商务网站尤其如此。

（6）性能测试

·很多软件都有特定的性能或效率目标，这些特性描述为在特定负载和配置环境下程序的响应时间和吞吐率。应设计测试用例来说明程序不能满足其性能目标。

（7）存储测试

·软件偶尔会有存储目标，例如描述程序使用的内存和辅存的容量以及临时文件或移出文件的大小。应设计测试用例来证明这些存储目标没有得到满足。

（8）配置测试

·很多软件都支持多种硬件配置，可以运行在多种操作系统下，使用多种 web 浏览器。通常可能的配置数量非常之大，以至于无法全面测试，但应该尽可能测试各种配置。

（9）兼容性/配置/转换测试

很多软件不是全新的，而是为了替换某些已有的系统。这样的软件往往涉及与已有系统的兼

容以及从已有系统的转换过程，如升级数据库管理系统。

（10）安装测试

有些软件的安装过程非常复杂，测试安装过程是系统测试的一个重要部分。

（11）可靠性测试

所有测试都是为了提高软件的可靠性，但如果软件的目标中包含了对可靠性的特别描述，就

必须设计专门的可靠性测试用例。

（12）适用性测试

对于软件的适用性和可维护性目标也必须测试。

（13）可恢复性测试

·诸如 OS、DBMS 等软件通常都有可恢复性目标，说明系统如何从硬件失败和数据错误中恢复过来。系统测试的一个目标是证明这些恢复机制不能正确发挥作用。

·可以故意将程序错误植入个系统中，判断系统是否可以从中恢复。

·这些系统的设计目标之一是平均恢复时间（MTTR）最小，测试目标之一就是证明系统不能满足 MTTR 的要求。

（13）文档测试

·系统测试也需要检查用户文档的正确性和清晰性。

（14）过程测试

·很多软件系统不是完全自动化的，其中包括了很多人员操作过程。在系统测试中，必须对所有已规定的人工过程，如系统操作员、最终用户、数据库管理员的操作过程进行测试。

十一、验收测试

·是将程序与其最初的需求及最终用户当前的需要进行比较的过程

·通常是由程序的客户或最终用户来进行，一般不认为是软件开发机构的职责

·最好的方法是设计测试用例，尽力证明程序没有满足合同要求；假如这些测试用例都通过了，就可以接受该程序。