一、判断分析题

1. 软件测试的目的是尽可能多的找出软件的缺陷。（Y）
2. 软件测试的目的是证明软件没有错误。（N）
3. **测试组负责软件质量。(N)**
4. 程序的效率与程序的复杂性相关。(N)
5. 软件是一种逻辑实体，而不是具体的物理实体，因而它具有抽象性。 (Y)
6. 测试程序仅仅按预期方式运行就行了。(N)
7. 好的测试员不懈追求完美。(Y)
8. 不存在质量很高但可靠性很差的产品。(N)
9. 测试是为了验证该软件已正确地实现了用户的要求。(N)
10. 发现错误多的程序模块，残留在模块中的错误也多。(Y)
11. 程序效率的提高主要应通过选择高效的算法来实现。(Y)
12. **测试人员要坚持原则，缺陷未修复完坚决不予通过。（N）**
13. **项目立项前测试人员不需要提交任何工件。（Y）**
14. 缺陷跟踪系统只针对对测试人员来使用。(N)
15. 从用户软件开发者的角度出发，普遍希望通过软件测试暴露软件中隐藏的错误和缺陷，以考虑是否可接受该产品。 (N)
16. 软件项目在进入需求分析阶段，测试人员应该开始介入其中。(Y)
17. 测试是提高产品质量根本手段。（）
18. **代码评审员一般由测试员担任。（N）**
19. **代码评审是检查源代码是否达到模块设计的要求。（N）**
20. 软件测试员可以对产品说明书进行白盒测试。(N)
21. 静态白盒测试可以找出遗漏之处的问题。(Y)
22. 总是首先设计白盒测试用例。(N)
23. 用黑盒法测试时，测试用例是根据程序内部逻辑设计的。(N)
24. 黑盒测试方法中最有效的是因果图法。(Y)
25. 软件测试按照测试过程分类为黑盒、白盒测试。(N)
26. 白盒测试又称结构测试、逻辑驱动测试或基于程序的测试。（Y）
27. 白盒测试时一般由开发人员兼任测试人员的角色。（Y）
28. 黑盒测试是从用户观点出发的测试。（Y）
29. 白盒测试是从用户观点出发的测试。（N）
30. 白盒测试根据程序外部特征进行测试，黑盒测试根据程序内部逻辑结构进行测试。（N）
31. 程序通过了全面的白盒测试，就不需要再进行黑盒测试了。（N）

32. 对于同一个测试对象，等价类的测试用例数多于边界值的测试用例数。（）
33. 如果输入条件规定了取值范围，则可定义一个有效等价类和两个无效等价类。（Y）
34. 不能对输出值域进行健壮性测试。（）
35. 有 n 个变量的函数的健壮最坏情况测试用例的个数为：5 的 n 次方。（Y）
36. 有 n 个变量的函数的健壮最坏情况测试用例的个数为：7 的 n 次方。（N）
37. 有 n 个变量的函数的健壮性测试用例的个数为：5n+1。（N）
38. 有 n 个变量的函数的最坏情况测试会产生 5 的 n 次方个测试用例。（Y）
39. 有 n 个变量的函数的边界值分析会产生 4n+1 个测试用例（Y）
40. 边界值分析的假设是“多缺陷”假设 。（N）
41. 健壮性测试的主要价值是观察例外情况的处理。（Y）
42. 在设计测试用例时，应包括合理的输入条件和不合理的输入条件。(Y)
43. 弱健壮等价类测试基于多缺陷假设。（N）
44. 强健壮等价类测试是基于多缺陷假设，并考虑了无效值。（Y）
45. 强一般等价类测试考虑了无效值。（Y）
46. 弱一般等价类是基于单缺陷假设。（Y）
47. 弱健壮等价类测试基于单缺陷假设并考虑了无效值。（Y）
48. 等价类测试的弱形式不如对应的强形式的测试全面（Y）
49. 语句覆盖是最弱的逻辑覆盖。（Y）
50. 语句覆盖是最强的逻辑覆盖。（N）
51. 判定覆盖不一定包含条件覆盖。（Y）
52. 条件覆盖不一定包含判定覆盖。（Y）
53. 判定/条件覆盖能同时满足判定、条件两种覆盖标准。（Y）
54. 判定/条件覆盖指满足判定覆盖标准或条件覆盖标准。（N）
55. DD-路径图是一种压缩图。（Y）
56. 详细设计的目的是为软件结构图中的每一个模块确定使用的算法和块内数据结构，并用某种选定的表达工具给出清晰的描述。(Y)
57. 尽量用公共过程或子程序去代替重复的代码段。（N）
58. 对于连锁型分支结构，若有 n 个判定语句，则有 2n 条路径。（Y）
59. 尽量采用复合的条件测试，以避免嵌套的分支结构。（Y）
60. GOTO 语句概念简单，使用方便，在某些情况下，保留 GOTO 语句反能使写出的程序更加简洁。（Y）
61. 单元测试能发现约 80%的软件缺陷。（Y）
62. 单元测试属于动态测试。 (N)
63. 单元测试多采用白盒测试（结构性测试）技术。（Y）

64. 单元测试需要从程序的内部结构出发设计测试用例。（Y）

65. 单元测试需要为每个基本单元开发驱动模块或桩模块。（Y）

66. 在面向对象语言语言中，单元测试是函数或子过程。（）

67. 单元测试又称为模块测试，是针对软件测试的最小单位－程序模块进行正确性检验的测试工作 。（Y）

68. 集成测试计划在需求分析阶段末提交。（N）

69. 自底向上集成需要测试员编写驱动程序。（Y）

70. 进行自底向上集成测试，需要为所测模块或子系统编制相应的驱动模块。（Y）

71. 进行自底向上集成测试，需要为所测模块或子系统编制相应的桩模块。（N）

72. 进行自顶向下集成测试，需要为所测模块或子系统编制相应的驱动模块。（N）

73. 进行自顶向下集成测试，需要为所测模块或子系统编制相应的桩模块。（Y）

74. MM-路径是可执行路径。（Y）

75. 非渐增式集成方式，发现错误难以诊断定位。（Y）

76. 集成测试是检验程序单元或部件的接口关系，逐步集成为符合概要设计要求的程序部件或整个系统。（Y）

77. 系统测试多采用白盒测试（结构性测试）技术。（N）（黑盒测试）

78. 验收测试是由最终用户来实施的。（N）

79. 负载测试是验证要检验的系统的能力最高能达到什么程度。（N）

80. 我们可以人为的使得软件不存在配置问题。（N）

81. 可以发布具有配置缺陷的软件产品。(Y)

82. 所有软件必须进行某种程度的兼容性测试。(Y)

83. 所有软件都有一个用户界面，因此必须测试易用性。(N)

84. β测试是由软件的多个用户在实际使用环境下进行的测试。（Y）

85. 系统测试是在真实或模拟系统运行环境下，检查完整的程序系统能否和相关硬件、外设、网络、系统软件和支持平台等正确配置与连接，并满足用户需求。（Y）

86. Beta 测试是验收测试的一种。（Y）

二、简答题

1. 什么是软件测试？软件测试的目的和作用是什么?

答：利用测试工具按照测试方案和流程对产品进行功能和性能测试，甚至根据需要编写不同的测试工具，设计和维护测试系统，对测试方案可能出现的问题进行分析和评估。

软件测试是在受控制的条件下对系统或应用程序进行操作并评价操作的结果。

软件测试的目的是以最少的时间和人力，系统地找出软件中潜在的各种错误和缺陷。测试是为了证明程序有错，而不是证明程序无错。一个成功的测试是发现了至今未发现的错误的测试。

软件测试的原则包括：所有的测试都应追溯到用户的需求；尽早地和不断地进行软件测试；不可能完全的测试，因为输入量太大，执行路径太多；注意测试中的群集现象；避免测试自己的程序；设计周密的测试用例。

2. 简述软件测试的目的和原则。

答：软件测试的目的是以最少的时间和人力，系统地找出软件中潜在的各种错误和缺陷。测试是为了证明程序有错，而不是证明程序无错。一个成功的测试是发现了至今未发现的错误的测试。

软件测试的原则包括：所有的测试都应追溯到用户的需求；尽早地和不断地进行软件测试；不可能完全的测试，因为输入量太大，执行路径太多；注意测试中的群集现象；避免测试自己的程序；设计周密的测试用例。

3. 软件缺陷产生的原因？

答：A．软件需求说明书编写的不全面，不完整，不准确，而且经常更改 B．软件设计说明书 C．软件操作人员的水平 D．开发人员不能很好的理解需求明书和沟通不足

4. 什么是软件测试，以及软件测试的意义？

答：软件测试是为了发现错误而执行程序的过程。软件测试是根据软件开发阶段的规格说明和程序的内部结构而精心设计的一批测试用例（即输入数据及预期的输出结果），并利用这些测试用例去运行程序，以发现错误的过程。

意义：

1. 对产品质量完成全面的评估，为软件产品发布（如验收测试）、软件系统部署（如性能规划测试）、软件产品鉴定（第三方独立测试）委托方和被委托方纠纷仲裁（第三方独立测试）和其它决策提供信息；
2. 通过持续的测试（包括需求评审、设计评审、代码评审等）可以对产品质量提供持续的、快速的反馈，从而在整个开发过程中不断地、及时地改进产品的质量，并减少各种返工，降低软件开发的成本；
3. 通过测试发现所要交付产品的缺陷，特别是尽可能地发现各种严重的缺陷，降低或消除产品质量风险，提高客户的满意度，扩大市场份额，提高客户的忠诚度。
4. 通过对缺陷进行分析，找出缺陷发生的根本原因（软件过程中的问题，包括错误的行为方式）或总结出软件产品的缺陷模式，避免将来犯同样的错误或产生类似的产品问题，达到缺陷预防的目的

5. 什么是软件测试？什么是测试用例，测试用例必须包括那几部分？

答：

狭义的讲，一个测试用例就是测试人员用以测试被测软件的某个特性或特性组合的 一

组数据。 这组数据可能是从用户处得来的实际的一组数据， 也可能是测试人员专门设计出 来的测试软件某些功能的一组数据。

6. 简述你对测试工作的认识过程、在以后的工作的一些建议。

7. 请辨析软件的质量是“设计出来的”还是“测试出来的”观点。

8. 软件测试与软件开发的关系？

答：软件开发是一个系统的工程。包括需求分析，设计，编码，测试，维护等等几个环节。测试是整个软件开发流程中的一个环节。

9. 在测试生命周期中，测试过程分为几个阶段？各个阶段的含义？以及各阶段的测试依据是什么？

答：软件测试是一个规则的过程，包括测试设计、测试执行以及测试结果比较等。

① 测试设计：根据软件开发各阶段的文档资料和程序的内部结构，利用各种设计测试用例技术精心设计测试用例。

② 测试执行：利用这些测试用例执行程序，得到测试结果。

③ 测试结果比较：将预期的结果与实际测试结果进行比较，如果二者不符合，对于出现的错误进行纠错，并修改相应文档。修改后的程序还要进行再次测试，直到满意为止。如果测试发现不了错误，可能由于测试配置考虑不周到，应考虑重新制定测试方案，设计测试用例。

按照开发阶段划分，软件测试可分为单元测试、集成测试，系统测试和验收测试。

单元测试：针对每个单元的测试， 以确保每个模块能正常工作为目标。

集成测试：对已测试过的模块进行组装，进行集成测试。

系统测试：检验软件产品能否与系统的其他部分协调工作。

验收测试：检验软件产品质量的最后一道工序。

10. 一名优秀的软件测试工程师应具备哪些素质？

答：一个优秀的测试工程师应该具备的基本素质有：责任心、沟通能力、团队精神、自信心、耐心、怀疑精神、洞察力、幽默感等。应具备的专业素质有：有竞争力的测试人员要具有三方面的技能：计算机专业技能、测试专业技能、软件编程技能。

11. 如何做一名合格的测试人员？

测试人员应具备素质如下：

(1) 沟通能力。

(2) 移情能力。

(3) 技术能力。

(4) 自信心。

(5) 外交能力。

(6) 幽默感。

(7) 很强的记忆力。

(8) 耐心。

(9) 怀疑精神。

(10) 自我督促。

(11) 洞察力。

12. 测试计划的目的是什么？

答：软件测试计划是指导测试过程的纲领性文件，包含了产品概述、测试策略、测试方法、测试区域、测试配置、测试周期、测试资源、测试交流、风险分析等内容。借助软件测试计划，参与测试的项目成员，尤其是测试管理人员，可以明确测试任务和测试方法，保持测试实施过程的顺畅沟通，跟踪和控制测试进度，应对测试过程中的各种变更。

13. 什么是黑盒测试?黑盒测试主要采用的技术有哪些?

答：黑盒测试又称为功能测试、数据驱动测试和基于规格说明的测试。它从用户观点出发的测试。用这种方法进行测试时，把被测试程序当作一个黑盒，在不考虑程序内部结构的内部特性、测试者只知道该程序输入和输出之间的关系或程序功能的情况下，依靠能够反映这一关系和程序功能需求规格的说明书，来确定测试用例和推断测试结果的正确性。

黑盒测试的方法包括：边界值分析、等价类测试、基于决策表的测试和因果图等。

14. 简单描述黑盒测试各种方法的特点。

答：黑盒测试的方法主要有边界值分析法、等价类划分法、因果图法、决策表测试法等。

边界值分析利用输入变量的最小值、略大于最小值、输入值域内的任意值、略小于最大值和最大值来设计测试用例。

等价类划分法是把程序的输入域划分为若干部分，然后从每个部分中选取少数代表性数据当作测试用例。经过类别的划分后，每一类的代表性数据在测试中的作用都等价于这一类中的其他值。

因果图方法就是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），将因果图转换为决策表，最后为决策表中的每一列设计一个测试用例。这种方法考虑到了输入情况各种组合以及各个输入情况之间的相互制约关系。

在所有的黑盒测试方法中，基于决策表的测试是最为严格、最具有逻辑性的。在一些数据处理问题当中，某些操作的实施依赖于多个逻辑条件的组合，即：针对不同逻辑条件的组合值，分别执行不同的操作。决策表法很适合测试这类问题。

15. 如果能够执行完美的黑盒测试，还需要进行白盒测试吗？为什么？

答：需要，黑盒测试可根据程序规格检验程序是否完成规定功能，但无法发现程序内部的编码和逻辑错误，白盒测试与之相反互补。

16. 边界值测试有几种方法？

答：边界值分析和健壮性测试

17. 等价分类法的测试技术采用的一般方法?举例说明?

答：

（1） 为每个等价类编号；

（2） 设计一个新的测试方案,以尽可能多的覆盖尚未被覆盖的有效等价类,重复这一步骤,直到所有有效等价类被覆盖为止。

（3） 设计一个新的测试方案,使它覆盖一个尚未被覆盖的无效等价类, 重复这一步骤,直到所有无效等价类被覆盖为止。

18. 什么是等价类？如何划分等价类？等价类测试中有哪些方法？

答：

把单元的输入域化分为几种数据类，每种用来发现一类的错误，每类只用提供一个或几个用例数据。目的是减少用例数量。

19. 请试着比较一下黑盒测试、白盒测试、单元测试、集成测试、系统测试、验收测试的区别与联系。

答：黑盒测试：已知产品的功能设计规格，可以进行测试证明每个实现了的功能是否符合要求。

白盒测试：已知产品的内部工作过程，可以通过测试证明每种内部操作是否符合设计规格要求，所有内部成分是否以经过检查。

软件的黑盒测试意味着测试要在软件的接口处进行。这种方法是把测试对象看做一个黑盒子，测试人员完全不考虑程序内部的逻辑结构和内部特性，只依据程序的需求规格说明书，检查程序的功能是否符合它的功能说明。因此黑盒测试又叫功能测试或数据驱动测试。黑盒测试主要是为了发现以下几类错误：

1、是否有不正确或遗漏的功能?

2、在接口上，输入是否能正确的接受?能否输出正确的结果?

3、是否有数据结构错误或外部信息(例如数据文件)访问错误?

4、性能上是否能够满足要求?

5、是否有初始化或终止性错误?

软件的白盒测试是对软件的过程性细节做细致的检查。这种方法是把测试对象看做一个打开的盒子，它允许测试人员利用程序内部的逻辑结构及有关信息，设计或选择测试用例，对程序所有逻辑路径进行测试。通过在不同点检查程序状态，确定实际状态是否与预期的状态一致。因此白盒测试又称为结构测试或逻辑驱动测试。白盒测试主要是想对程序模块进行如下检查：

1、对程序模块的所有独立的执行路径至少测试一遍。

2、对所有的逻辑判定，取"真"与取"假"的两种情况都能至少测一遍。

3、在循环的边界和运行的界限内执行循环体。

4、测试内部数据结构的有效性，等等。

单元测试**(**模块测试**)**是开发者编写的一小段代码，用于检验被测代码的一个很小的、很明确的功能是否正确。通常而言，一个单元测试是用于判断某个特定条件**(**或者场景**)**下某个特定函数的行为。

单元测试是由程序员自己来完成，最终受益的也是程序员自己。可以这么说，程序员有责任编写功能代码，同时也就有责任为自己的代码编写单元测试。执行单元测试，就是为了证明这段代码的行为和我们期望的一致。

集成测试**(**也叫组装测试，联合测试**)**是单元测试的逻辑扩展。它的最简单的形式是：两个已经测试过的单元组合成一个组件，并且测试它们之间的接口。从这一层意义上讲，组件是指多个单元的集成聚合。在现实方案中，许多单元组合成组件，而这些组件又聚合成程序的更大部分。方法是测试片段的组合，并最终扩展进程，将您的模块与其他组的模块一起测试。最后，将构成进程的所有模块一起测试。

系统测试是将经过测试的子系统装配成一个完整系统来测试。它是检验系统是否确实能提供系统方案说明书中指定功能的有效方法。**(**常见的联调测试**)**

系统测试的目的是对最终软件系统进行全面的测试，确保最终软件系统满足产品需求并且遵循系统设计。

验收测试是部署软件之前的最后一个测试操作。验收测试的目的是确保软件准备就绪，并且可以让最终用户将其用于执行软件的既定功能和任务。

验收测试是向未来的用户表明系统能够像预定要求那样工作。经集成测试后，已经按照设计把所有的模块组装成一个完整的软件系统，接口错误也已经基本排除了，接着就应该进一步验证软件的有效性，这就是验收测试的任务，即软件的功能和性能如同用户所合理期待的那样。

20. 什么是白盒测试?白盒测试主要采用的技术有哪些? 白盒测试有那几种方法？并简单描述各种方法的特点。

答：白盒测试又称为结构测试、逻辑驱动测试或基于程序的测试。它依赖于对程序细

节的严密的检验。针对特定条件和循环集设计测试用例，对软件的逻辑路径进行测试。在程序的不同点检验程序的状态，来进行判定其实际情况是否和预期的状态相一致。

白盒测试包括：逻辑覆盖、基路径测试、数据流测试、程序插装等。

总体上分为静态方法和动态方法两大类

静态：关键功能是检查软件的表示和描述是否一致,没有冲突或者没有歧义

动态：语句覆盖、判定覆盖、条件覆盖、判定条件覆盖、条件组合覆盖、路径覆盖。

21. 常用的逻辑覆盖测试方法有哪几种？并简单描述各种方法的目的。

答：逻辑覆盖可分为：语句覆盖、判定覆盖、条件覆盖、判定/条件覆盖及路径覆盖。

语句覆盖：要求设计若干个测试用例，运行被测程序，使程序中的每个可执行语句至少被执行一次。

判定覆盖：要求设计若干测试用例，运行被测程序，使得程序中每个判断的取真分支和取假分支至少执行一次，即判断的真假值均要被检测。判定覆盖又称为分支覆盖。

条件覆盖：要求设计若干测试用例，执行被测程序，使得程序中每个判断的每个条件的可能取值至少被执行一次。

判定/条件覆盖：要求设计足够的测试用例，执行被测程序，使得判断中每个条件的所有可能取值至少被执行一次，同时每个判断的所有可能判断结果也至少被执行一次。

路径覆盖：要求设计足够多测试用例，覆盖程序中所有可能的路径。

22. 逻辑覆盖中几种主要覆盖的含义?举例说明?

答：逻辑覆盖是一种使用最广泛的结构测试方法。逻辑覆盖以程序内部的逻辑结构为基础设计测试用例，要求对被测程序的逻辑结构有清楚的了解，甚至要能掌握源程序的所有细节。

由于覆盖测试的目标不同，逻辑覆盖可分为：语句覆盖、判定覆盖、条件覆盖、判定/条件覆盖及路径覆盖。

23. 单元测试有那些步骤，各个步骤有那些实施内容。

答：**1.静态检查:用工具 Logiscope 或者人工检查单**

2.动态检查:用工具 PurifyPlus 或者人工调试

3.测试用例执行(工具或人工)

设计测试用例及数据；（提前）

编写测试用例代码、脚本、驱动模块和桩模块；（提前）

运行测试用例，记录结果。（在步骤 **1,2** 之后进行）

24. 非增量式测试与增量式测试

答：非增量式测试的方法是先分散测试，然后集中起来再一次完成集成测试。假如在模块的接口处存在错误，只会在最后的集成测试时一下子暴露出来。

增量式测试是逐步集成和逐步测试的方法，把可能出现的差错分散暴露出来，便于找

出问题和修改。而且一些模块在逐步集成的测试中，得到了较多次的考验，因此，可能会取得较好的测试效果。

结论：增量式测试要比非增量式测试具有一定的优越性。

25. 简述自顶向下增量式测试和自底向上增量式测试两种集成测试方法，并比较两者的优点和缺点。

答：自顶向下增量式测试：

主要优点在于它可以自然的做到逐步求精，一开始就能让测试者看到系统的框架。

主要缺点是需要提供桩模块，并且在输入/输出模块接入系统以前，在桩模块中表示测试数据有一定困难。

自底向上增量式测试：

优点在于，由于驱动模块模拟了所有调用参数，即使数据流并未构成有向的非环状图，生成测试数据也无困难。

主要缺点在于，直到最后一个模块被加进去之后才能看到整个程序（系统）的框架。

26. 简述集成测试的过程。集成测试的方法有那些？

答：系统集成测试主要包括以下过程：

1. 构建的确认过程。
2. 补丁的确认过程。
3. 系统集成测试测试组提交过程。
4. 测试用例设计过程。
5. 测试代码编写过程。
6. Bug 的报告过程。
7. 每周/每两周的构建过程。
8. 点对点的测试过程。
9. 组内培训过程。

27. 比较自顶向下集成测试和自底向上集成测试的优劣?

答：自顶向下集成测试表示逐步集成和逐步测试是按照结构图自上而下进行的，即模块集成的顺序是首先集成主控模块（主程序），然后依照控制层次结构向下进行集成。自底向上集成测试表示逐步集成和逐步测试的工作是按结构图自下而上进行的，由于是从最底层开始集成，所以也就不再需要使用桩模块进行辅助测试。

自顶向下测试的主要优点在于它可以自然的做到逐步求精，一开始就能让测试者看到系统的框架。它的主要缺点是需要提供桩模块。自底向上的优点在于不需要桩模块，需用的驱动模块比较少。它的主要缺点在于，直到最后一个模块被加进去之后才能看到整个程序（系统）的框架。

28. 系统测试计划是否需要同行评审，为什么？

答：需要,系统测试计划属于项目阶段性关键文档,因此需要评审。系统测试计划需要进行同行评审，因为如果对一个系统长时间进行测试可能会出现测试疲劳甚至出现对系统的免疫现象，因此可以进行同行评审，减少对相同系统的疲劳测试。

29. 什么叫α测试（Alpha 测试）？什么叫β测试（beta 测试）？α测试和β测试有什么区别？

答：Alpha 测试 在系统开发接近完成时对应用系统的测试；测试后仍然会有少量的设计变更。这种测试一般由最终用户或其它人员完成，不能由程序或测试员完成。

Beta 测试 当开发和测试根本完成时所做的测试，最终的错误和问题需要在最终发行前找到。这种测试一般由最终用户或其它人员完成，不能由程序员或测试员完成。

30. 什么是单元测试？什么是集成测试？什么是系统测试？他们的测试依据是什么？他们和功能测试，结构性测试有何关系？

31. 阶段评审与同行评审的区别。

答：同行评审目的:发现小规模工作产品的错误,只要是找错误；

阶段评审目的:评审模块 阶段作品的正确性 可行性 及完整性

同行评审人数:3-7 人 人员必须经过同行评审会议的培训,由 SQA 指导

阶段评审人数:5 人左右 评审人必须是专家 具有系统评审资格

同行评审内容:内容小 一般文档 < 40 页，代码 < 500 行

阶段评审内容: 内容多,主要看重点

同行评审时间:一小部分工作产品完成

阶段评审时间: 通常是设置在关键路径的时间点上!

32. 回归测试测试如何进行？

答：

(1). 识别出软件中被修改的部分；

(2). 从原基线测试用例库 T 中，排除所有不再适用的测试用例，确定那些对新的软件版本依然有效的测试用例，其结果是建立一个新的基线测试用例库 T0。

(3). 依据一定的策略从 T0 中选择测试用例测试被修改的软件。

(4). 如果必要，生成新的测试用例集 T1，用于测试 T0 无法充分测试的软件部分。

(5). 用 T1 执行修改后的软件。

第(2)和第(3)步测试验证修改是否破坏了现有的功能，第(4)和第(5)步测试验证 修改工作本身。