2021 考点

1、在软件测试过程中，应该遵循的原则

1.软件测试是为了证伪而非证真
2.尽早的和不断的进行软件测试
3.重视无效数据和非预期使用习惯的测试
4.程序员应该避免检查自己的程序
5.充分注意测试中的群集现象（28 原则）
6.用例要定期评审
7.应当对每一个测试结果做全面检查
8.测试要保护现场和资料归档
9.软件测试的经济型原则

2、测试用例的设计

2.0“测试用例”概念：

测试用例，英文为 TestCase,缩写为 TC，指的是在测试执行之前设计的一套详细的测试方案，包括测试环境、测试步骤、测试数据和预期结果。

黑白盒都有测试用例

意义：测试用例是测试工作的指导，是软件测试的必须遵守的准则，更是软件测试质量稳定的根本保障。

2.1 依据：

我们编写测试用例的唯一标准就是用户需求,具体的参考资料是《需求规格说明书》，但需要说明的是，用户需求不是一成不变的，而是在一直变化的，这就需要我们根据不断调整变化的需求，来修改和维护我们已写好的测试用例，这个工作量也很大。

2.2 为何需要测试用例?

在开始实施测试之前设计好测试用例，避免盲目测试并提高测试效率，减少测试的不完全性;

2. 测试用例的使用令软件测试的实施重点突出、目的明确;

3. 根据测试用例的多少和执行难度，估算测试工作量，便于测试项目的时间和资源管理与跟踪;

4. 减少回归测试的复杂程度,在软件版本更新后只需修正少量的测试用例便可展开测试工作，降低工作强度、缩短项目周期;

5. 功能模块的测试用例的通用化和复用化则会使软件测试易于开展,并随着测试用例的不断细化其效率也不断攀升;

6. 根据测试用例的操作步骤和执行结果,为分析软件缺陷和程序模块质量提供依据;可以方便地书写软件测试缺陷报告;

7. 可以根据测试用例的执行等级，实施不同级别的测试;

总结:

软件测试是有组织性、步骤性和计划性的，为了能将软件测试的行为转换为可管

理的、具体量化的模式，需要创建和维护测试用例。

2.3 好的测试用例

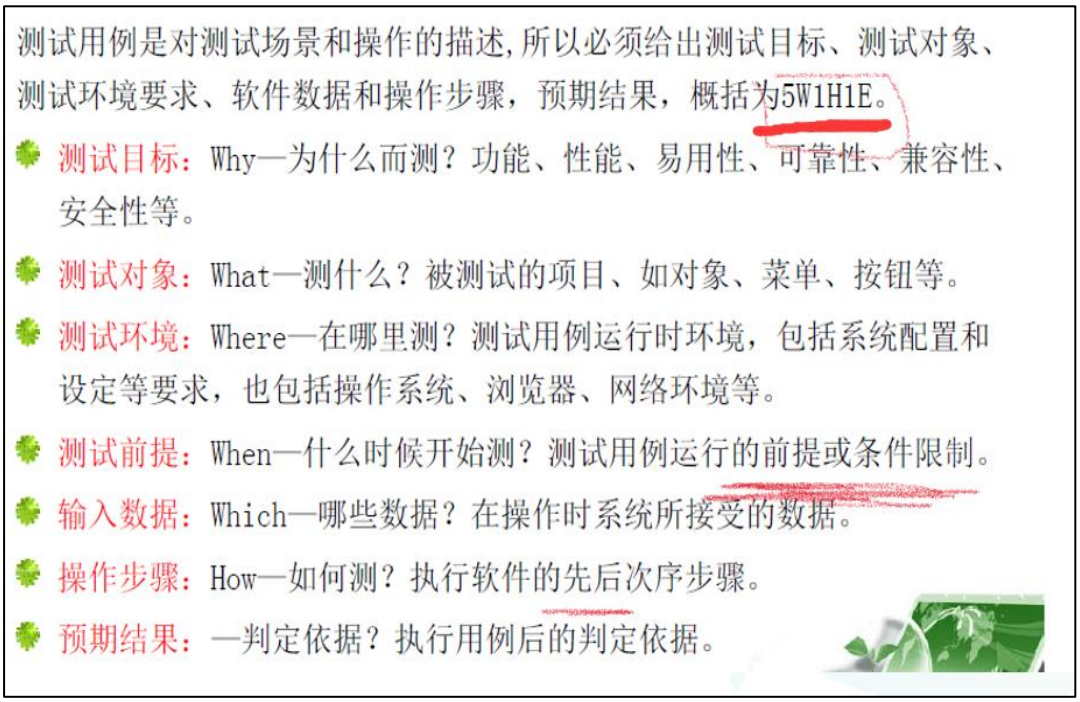
- 可以最大程度地找出软件隐藏的缺陷
- 可以最高效率的找出软件缺陷
- 可以最大程度地满足测试覆盖要求既不过分复杂、也不能过分简单
- 是软件的缺陷的表现可以清楚的判定
- 测试用例包含期望的正确的结果
- 待查的输出结果或文件必须尽量简单明了
- 不包含重复的测试用例
- 测试用例内容清晰、格式一致、分类组织

2.4 影响因素

需求目标，是功能性的需求目标也是非功能性的需求目标。功能性测试比较清楚，正确与否一目了然，而非功能性测试，其相对性比较强，需要从不同角度比照。

2. 用户实际使用场景。从用户的角度来模拟程序的输入，包括用户的操作习惯，使产品更能贴近用户的需求。
3. 软件功能需求规格说明书、产品设计文档。
4. 测试方法对测试用例的设计影响非常大。
5. 测试对象。客户端软件和服务器端系统、分布式系统和集中式系统等。
6. 软件实现所采用的技术。

2.5 测试用例的元素



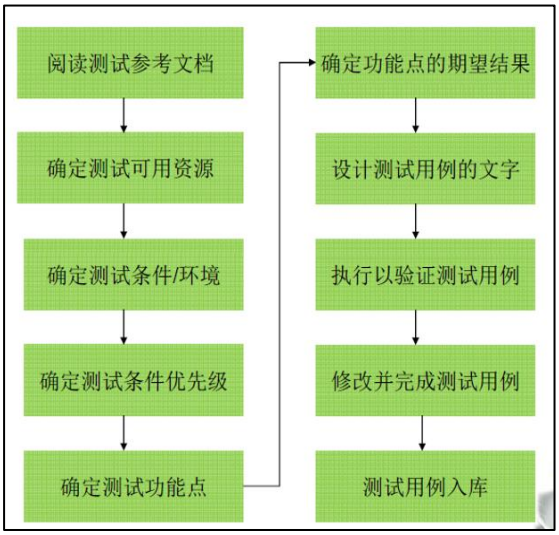
组成：

2.6 测试用例分类

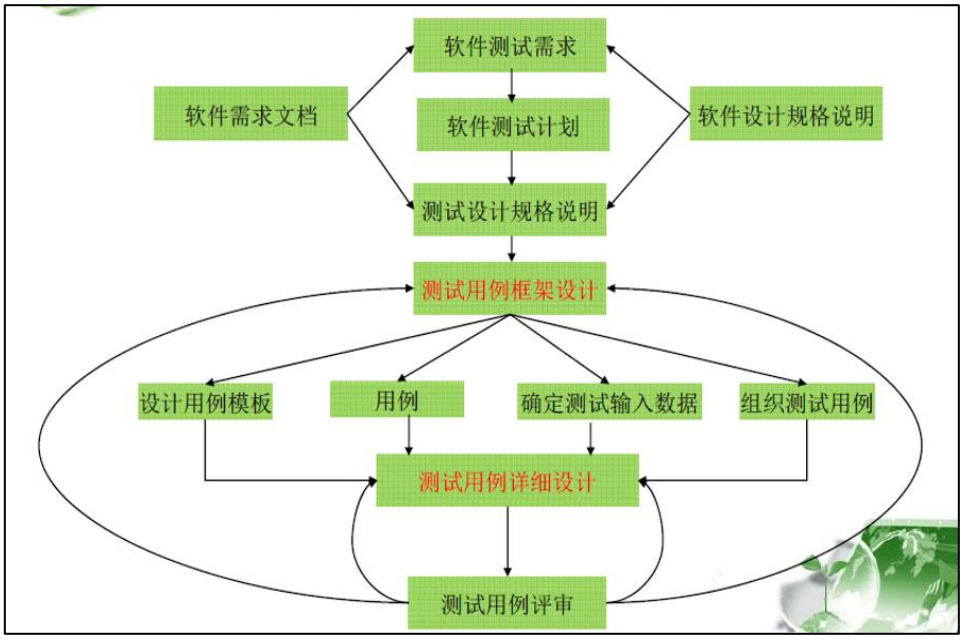
2.7 测试用例设计步骤及方法

联系 5w1h1e 来记忆:

步骤:



方法：



3、测试用例的原则

利用成熟的测试用例设计方法来指导设计

2. 测试用例的针对性
3. 测试用例的代表性
4. 测试用例的可判定性

5. 测试用例的可重现性
6. 足够详细、准确和清晰的步骤
7. 测试用例必须符合内部的规范的要求

4、常用到的软件质量模型

软件质量模型

关于软件质量模型，业界已经有很多成熟的模型定义，比较常见的质量模型有 McCall 模型、Boehm 模型、FURPS 模型、Dromey 模型和 ISO9126 模型。

- *Jim McCall 软件质量模型（1977 年）*
- *Barry W. Boehm 软件质量模型（1978 年）*
- *FURPS/FURPS+ 软件质量模型*
- *R. Geoff Dromey 软件质量模型*
- *ISO/IEC 9126 软件质量模型（1993 年）*
- *ISO/IEC 25010 软件质量模型（2011 年）*

书上的重点介绍：

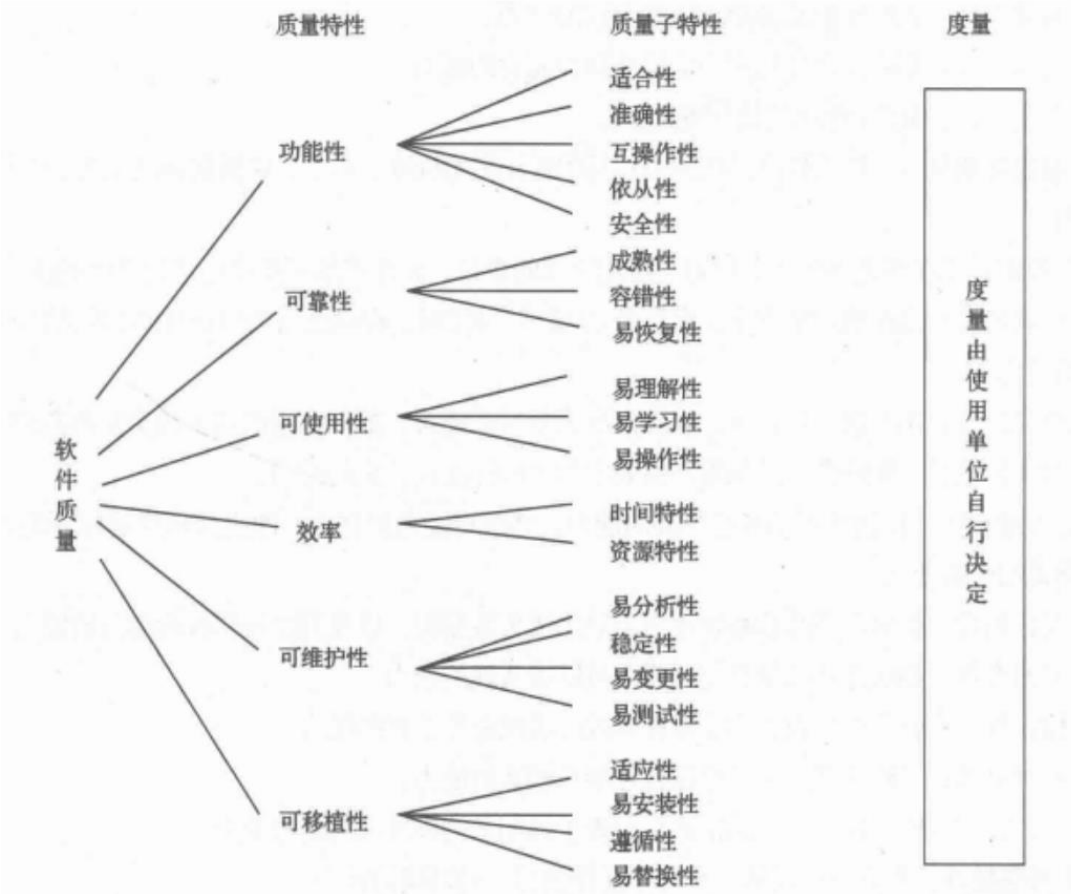


图1-4 ISO软件质量度量模型

功能性：是指当软件在指定条件下使用，软件产品满足明确和隐含要求功能的能力。
适合性：是指软件产品与指定的任务和用户目标提供一组合适的功能的能力。
准确性：是指软件产品具有所需精确度的正确或相符的结果及效果的能力。
互操作性：是指软件产品与一个或多个规定系统进行交互的能力。
安全性：是指软件产品保护信息和数据的能力，以使未授权的人员或系统不能阅读或修改这些
信息和数据，但不拒绝授权人员或系统对其的访问。
依从性：是指软件产品依附与同功能性相关的标准、约定或法规以及类似规定的能力。
2. 可靠性：在指定条件下使用时，软件产品维持规定的性能级别的能力。
成熟性：是指软件产品避免因软件中错误发生而导致失效的能力。
容错性:是指在软件发生故障或违反指定接口的情况下，软件产品维持规定的性能级别的能力。
易恢复性：是指在失效发生的情况下，软件产品重建规定的性能级别并恢复受直接影响的数据的能力。
可靠性依从性：是指软件产品依附与同可靠性相关的标准、约定或法规，以及类似规定的能力。
3. 可使用性：在指定条件下使用时，软件产品被理解、学习、使用和吸引用户的能力。
易理解性：指软件产品使用户能理解软件产品是否合适，以及如何能将软件用于特定的任务和使用环境的能力。

易学习性：指软件产品使用户能学习它的能力。
易操作性：指软件产品使用户能操作和控制它的能力。
吸引性：指软件产品吸引用户的能力。
易用性依从性：指软件产品依附与同易用性相关的标准、约定、风格指南或法规，以及类似规定的能力。
4. 效率：指在规定条件下，相对于所用资源的数量，软件产品可提供适当的性能的能力。
时间特性：指在规定条件下，软件产品执行其功能时，提供适当的响应时间和处理时间，以及吞吐率的能力。
资源特性:指在规定条件下,软件产品执行其功能时,提供合适的数量和类型的资源的能力。
效率依从性：指软件产品依附与同效率相关的标准或约定的能力。
5. 可维护性：指软件产品可被修改的能力，修改可能包括修正，改进或软件适应环境、需求和功能规格说明中的变化。
易分析性：指软件产品诊断软件中的缺陷或失效原因，以及判定待修改的部分的能力。
易改变性：指软件产品使指定的修改可以被实现的能力。
稳定性：指软件产品避免由于软件修改而造成意外结果的能力。
易测试性：指软件产品使已修改软件能被确认的能力。
维护性依从性：指软件产品依附与同维护性相关的标准或约定的能力。
6. 可移植性：指软件产品从一种环境迁移到另一种环境的能力。
适应性：指软件产品无需采用有别于为考虑该软件的目的而准备的活动或手段，就可能适应不同的指定环境的能力。
易安装性：指软件产品在指定环境中被安装的能力。
共存性：指软件产品在公共环境中同与其分享公共资源的其他独立软件共存的能力。
易替换性：指软件产品在环境相同、目的相同的情况下替代另一个指定软件产品的能力。
可移植性依从性：指软件产品依附与同可移植性相关的标准或约定的能力。

5.软件测试计划

概念:

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，风险分析等内容；借助软件测试 计划，参与测试的项目成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。

6.制定软件测试的计划的原则

编写测试计划需要尊重“5W”原则。
“5W”规则指的是“What （做什么）”、"Why （为什么做）”、 “When （何时做）”、"Where（在哪里）”、“How （如何做）”。利用“5W”规则创建软件测试计划，可 以帮助测试团队理解测试的目的(Why),明确测试的范围和内容(What),确定测试的开始和结束日 期(When),指出测试的方法和工具(How),给出测试文档和软件的存放位置(Where)。为了使“5W” 规则更具体化，需要准确理解被测软件的功能特征、应用行业的知识和软件测试技术，在需要测试的 内容中突出关键部分，可以列出关键及风险内容、属性、场景或者测试技术。对测

试过程的阶段划分、 文档管理、缺陷管理、进度管理给出切实可行的方法。

7.制定软件测试的技术的步骤（可能标题有误？？）

4. 软件测试要经过哪些步骤?这些测试与软件开发各阶段之间有什么关系?

需求分析——确认测试

设计阶段——集成测试

编程——单元测试

步骤:

测试计划和控制

2. 测试分析和设计
3. 测试实现和执行
4. 评估出口准则和报告
5. 测试结束活动

8.静态测试、动态测试

静态测试(static testing),是指不实际运行被测软件，而只是静态地检查程序代码、界面或文档中可能存在的错误过程。

- 代码测试：主要测试代码是否符合相应的标准和规范。
- 界面测试：主要测试软件的实际界面与需求中的说明是否相符。
- 文档测试：主要测试用户手册和需求说明是否真正符合用户的实际需求。
- 工具, (Logiscope) Telelogic，可以用作 Java/C++等规范

动态测试(dynamic testing)、是指实际运行被测程序，输入相应的测试数据，检查实际输出结果和预期结果是否相符的过程。

- 动态测试方法为结构和正确性测试
- 动态测试工具 Robot、QTP 等

9.白盒测试、黑盒测试以及二者的关系

分别概念：

白盒测试指的是把盒子打来，去研究里面的源代码和程序结构。 白盒测试又称结构测试或基于程序的测试。该方法将被测对象看作一个打开 的盒子, 允许人们检查其内部结构。测试人员根据程序内部的结构特性，设计和 选择测试用例，检测程序的每条路径是否都按照预定的要求正确地执行。 白盒测试也称结构测试或逻辑驱动测试，是针对被测单元内部是如何进行工 作的测试。它根据程序的控制结构设计测试用例, 主要用于软件或程序验证。

黑盒测试又称数据驱动测试，完全不考虑程序内部结构和内部特性，注重于 测试软件的功能需求。 黑盒测试是一种常用的软件测试方法，在应用这种方法设计测试用例时，把被测程序看成是一个打不开的黑盒, 测试人员在不考虑程序内部结构和内部特性 的情况下,

只根据需求规格说明书，设计测试用例，检查程序的功能是否按照规 范说明的规定正确执行。

关系：

- 是测试用例基本概念
- 软件公司往往采用两种测试相结合的方式
- 整体功能和性能进行黑盒测试
- 源代码采用白盒测试

11.白盒测试的覆盖准则

白盒测试法的覆盖标准有逻辑覆盖、循环覆盖和基本路径测试。其中逻辑覆 盖包括语句覆盖、判定覆盖、条件覆盖、判定、条件覆盖、条件组合覆盖和路径 覆盖。六种覆盖标准发现错误的能力呈由弱到强的变化。

语句覆盖每条语句至少执行一次。

2.判定覆盖每个判定的每个分支至少执行一次。

3.条件覆盖每个判定的每个条件应取到各个可能的值。

4.判定/条件覆盖的同时满足判定覆盖条件覆盖 。

5.条件组合覆盖每个判定中各条件的每一种组合至少出现一次。

6.路径覆盖使程序中每一条可能的路径至少执行一次。

24.常见的黑盒测试用例的设计方法?并分别简单介绍一下各自的思想。

- 等价类划分：从大量数据中划分范围（等价类），然后从每个范围中挑选代表数据，这些代表数据要能反应这个范围内数据的测试结果。
- 边界值划分：程序的很多错误发生在输入或输出范围的边界上，因此针对边界情况设置测试用例，可以发现不少程序缺陷。
- 错误推测法：基于经验和直觉推测推测所有可能存在的错误，有针对性的设计测试用例的方法。
- 因果图法：适合输入条件比较多的情况，测试所有的输入条件的排列组合。
- 正交表实验法：从大量试验点中挑出适量的、有代表性的点，利用“正交表”，合理的安排试验对所有变量对的组合进行典型覆盖。
- 场景图：现在的软件几乎都是用事件触发来控制流程的，事件触发时的情景便形成了场景，而同一事件不同的触发顺序和处理结果就形成事件流。这种在软件设计方面的思想也可以引入到软件测试中，可以比较生动地描绘出事件触发时的情景，有利于测试设计者设计测试用例，同时使测试用例更容易理解和执行。
- 功能图：用功能图形象的描述程序的功能说明，并机械的生成功能图的测试用例。功能图由状态迁移图和逻辑功能模型构成。

12. 白盒测试的常用工具，各适用于什么（不确定）

- 代码分析检查工具：FindBugs、PMD
- 覆盖率工具：jacoco、EMMA
- 白盒测试工具：junit、testng
- 质量管理平台：SonarQube
- 编译工具：Ant、maven
- Mock 工具：Jmockit
- 数据库：mysql
- 持续集成工具：Jenkins
- 代码管理工具：SVN、git

10.软件测试与软件开发的过程的关系

4. 软件测试要经过哪些步骤?这些测试与软件开发各阶段之间有什么关系?

（1）软件测试要经过的步骤

单元测试→集成测试→确认测试 → 系统测试→验收测试

（2）关系

○1 单元测试：对源程序中每一个程序单元进行测试，检查各个模块是否正确实现规定的功能，从而发现模块在编码中或算法中的错误。该阶段涉及编码和详细设计文档。

○2 集成测试：是为了检查与设计相关的软件体系结构的有关问题，也就是检查概要设计是否合理有效。

○3 确认测试：主要是检查已实现的软件是否满足需求规格说明书中确定了的各种需求。分 Alpha 测试和 Beta 测试。

○4 系统测试：是把已确认的软件与其他系统元素（如硬件、其他支持软件、数据、人工等）结合在一起进行测试。以确定软件是否可以支付使用。

或：

软件开发是一个自顶向下、逐步细化的过程，软件计划阶段定义软件作用域；软件需求阶段分析建立了软件信息域、功能和性能需求、约束等；软件设计把设计用某种程序设计语言转换成程序代码。

而测试过程是依相反顺序自底向上，逐步集成的过程。它对每个程序模块进行单元测试，消除程序模块内部逻辑和功能上的错误和缺陷；对照软件设计进行

集成测试，检测和排除子系统或系统结构上的错误；对照需求，进行确认测试；最后从系统全体出发，运行系统，看是否能够满足要求。软件测试与软件开发各阶段的关系如下图所示：

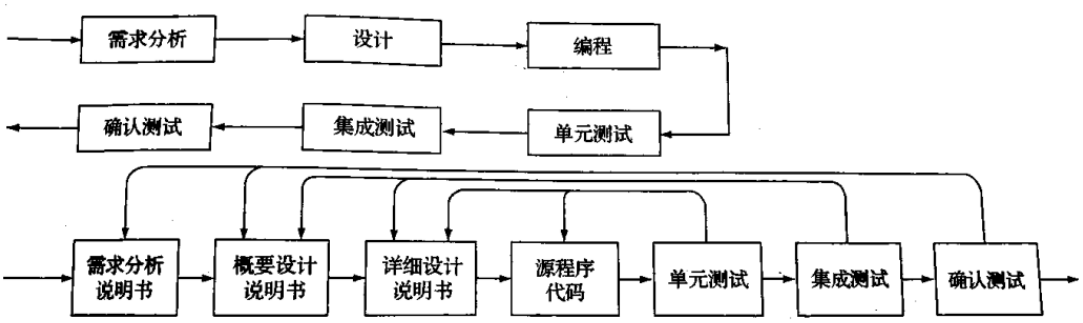


图 3 软件测试与软件开发各阶段的关系

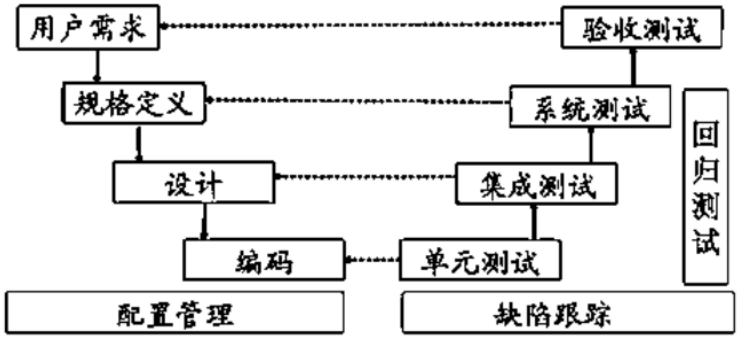


图 4 软件测试与软件开发各阶段的关系

13.单元测试（PPT2.3）

1) 单元测试的原则

- Right. 结果是否正确。
- Boundary. 主要是 CORRECT 原则。
- InReverse. 相反性。
- Cross.
- Error.
- Performance.
- A-trip 原则.

2) 单元测试的重要性及目的

执行单元测试的目的在于验证程序是否像您预期的那样正确工作，并尽早地发现错误。除此之外，单元测试还会给开发人员带来一些益处:

- 编写单元测试可以帮助开发人员书写更高质量的代码。
- 编写单元测试可以使开发人员更有信心重构应用程序，去拥抱变化。

3）单元测试主要测试问题（PPT2.3）

模块接口测试

（1）**模块接口测试**　检查进出程序单元的数据流是否正确，这一过程必须在其他测试之前进行。如果数据不能正确输入和输出，就谈不上进行其他测试。

模块接口测试项目主要包括：

- 调用所测模块时的输入参数与模块的形式参数在个数、属性、顺序上是否匹配；
- 所测模块调用子模块时，它输入子模块的参数与子模块的形式参数在个数、属性、顺序上是否匹配；
- 输出给标准函数的参数在个数、属性、顺序上是否匹配；
- 全局变量的定义在各模块是否一致；

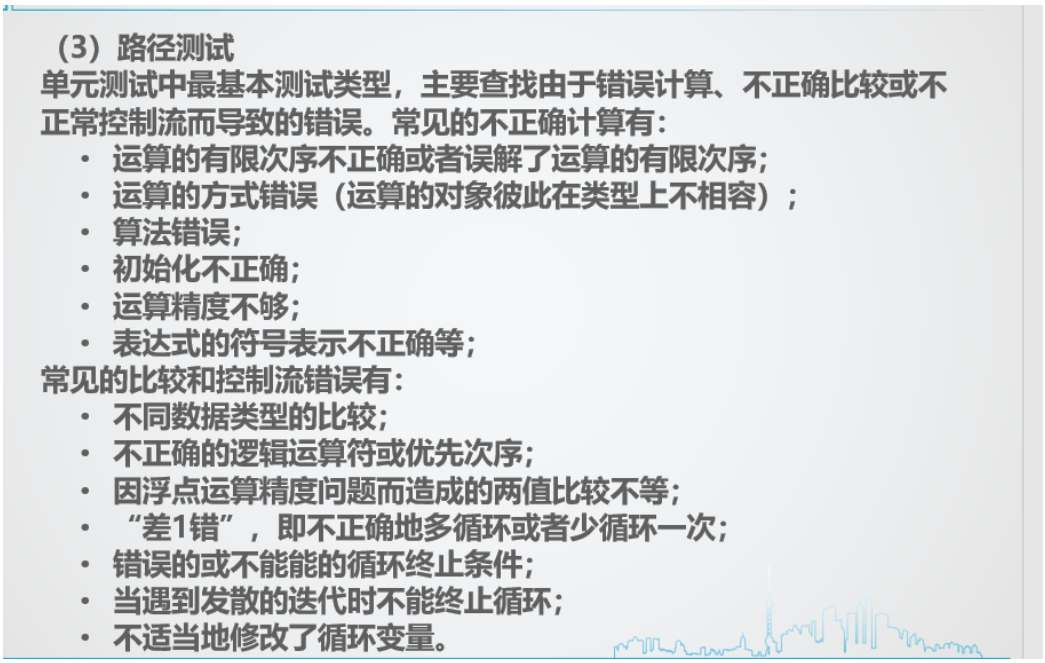
局部数据结构测试

单元测试内容

（2）**局部数据结构测试**　最常见错误来源。主要测试项目包括：

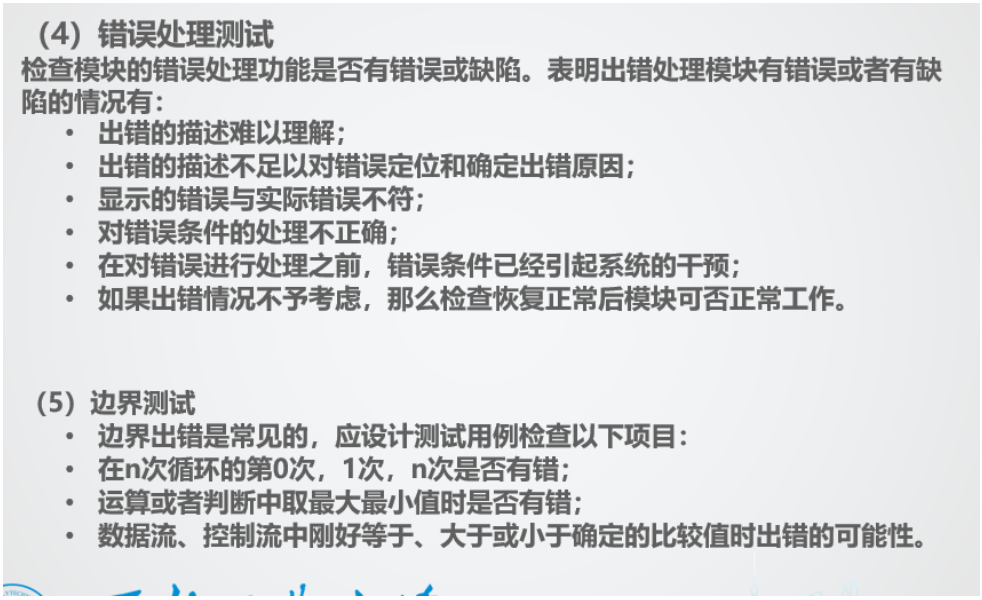
- 检查不正确或者不一致的数据类型说明；
- 使用尚未赋值或尚未初始化的变量；
- 错误的初始值或者默认值；
- 变量名拼写错误或书写错误；
- 不一致的数据类型。

路径测试



错误处理测试

边界测试



14.插桩程序设计

程序插桩简介<CSDN>

一种基本的动态测试方法，向源程序中添加一些语句实现对程序代码的执行、变量的变化等情况的检查，可以获得程序的控制流和数据流信息。如果我们想要了解一个程序在某次运行中可执行语句被覆盖的情况，或是每个语句的实际执行次数，最好的办法就是利用插装技术，它在软件测试技术上占有非常高的地位。最简单的插装：在程序中插入打印语句

printf(“ …”)语句。

插桩位置：a.程序的第一条语句；b.分支语句的开始；c.循环语句的开始；d.下一个入口语句之前的语句；e.程序的结束语句；f.分支语句的结束；g.循环语句的结束。

2.插桩策略：

①语句覆盖探针（基本块探针）：在基本块的入口和出口处，分别植入相应的探针，以确定程序执行时该基本块是否被覆盖。

②分支覆盖探针：c/c++语言中，分支由分支点确定。对于每个分支，在其开始处植入一个相应的探针，以确定程序执行时该分支是否被覆盖。

③条件覆盖探针：c/c++语言中，if, swich,while, do-while, for 几种语法结构都支持条件判定，在每个条件表达式的布尔表达式处植入探针，进行变量跟踪取值，以确定其被覆盖情况。

3.设计插桩程序需要注意的几点：

①探测那些信息；

②在什么位置设置探针；

③设置多少个探测点；

④特定位置插入用以判断变量特性的语句。

———————————————

15.集成测试（PPT2.4）：

1）集成测试的主要任务

集成测试又叫组装测试，通常在单元测试的基础上，将所有程序模块进行有序的、递增的测试。重点测试不同模块的接口部分。

2）集成测试与单元测试，系统测试的区别

单元测试：

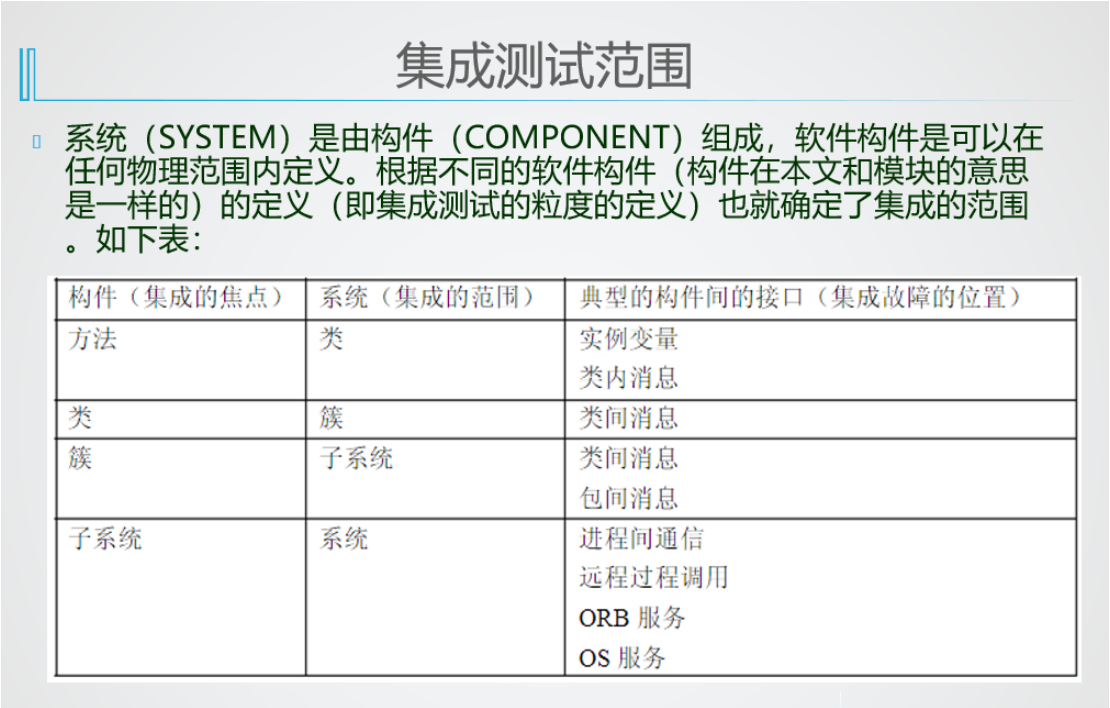
又称模块测试，针对软件设计中的最小单位——程序模块，进行正确性检查的测试工作。

系统测试：

指的是将整个软件系统看为一个整体进行测试，包括对功能、性能、以及软件所运行的软硬件环境进行测试。

目前系统测试主要由黑盒测试工程师在系统集成完毕后进行测试，前期主要测试系统的功能是否满足需求，后期主要测试系统运行的性能是否满足需求，以及系统在不同的软硬件环境中的兼容性等。

3）集成测试的内容



4）集成测试的方法

集成测试有两种不同的方法，非增式测试和增式测试。

a）非增式测试

- 在配备辅助模块的条件下，对所有模块进行个别的单元测试。然后在此基础上，按程序结构图将各模块联接起来，把联接后的程序当作一个整体进行测试。
- 典型的测试方法有大爆炸集成测试。
- 非增式测试的做法是先分散测试，再集中起来一次完成集成测试。如果在模块的接口处存在错误，只会在最后的集成时一下子暴露出来，便于找出问题和修改。

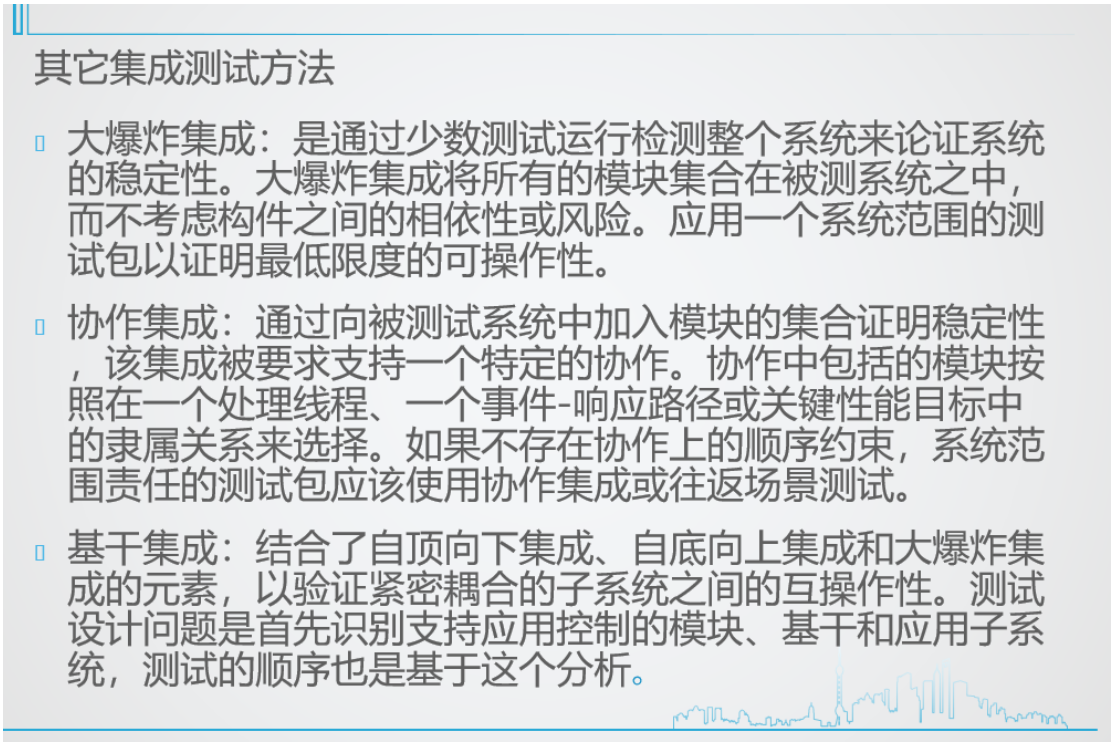
b）增式测试

增式集成是逐步实现的，测试过程使用了较少的辅助模块，也就减少了辅助性测试工作。并且一些模块在逐步集成的测试中，得到了较为频繁的考验，因而可能取得较好的测试效果。

主要有两种实施顺序:

- 自顶向下增式测试表示逐步集成和逐步测试是按结构图自上而下进行的。
- 自底向上增式测试表示逐步集成和逐步测试是按结构图自下而上进行的。

c)



16.系统测试

1)系统测试与用户测试的区别

系统测试：

指的是将整个软件系统看为一个整体进行测试，包括对功能、性能、以及软件所运行的软硬件环境进行测试。

目前系统测试主要由黑盒测试工程师在系统集成完毕后进行测试，前期主要测试系统的功能是否满足需求，后期主要测试系统运行的性能是否满足需求，以及系统在不同的软硬件环境中的兼容性等。

用户测试：属于第三方测试

第三方测试是由开发者和用户以外的第三方进行的软件测试，其目的是为了保证测试的客观性。 狭义的理解是独立的第三方测试机构，如国家级软件评测中心，各省软件评测中心，有资质的软件评 测企业。广义的理解是非本软件的开发人员。如 QA 部门人员测试、公司内部交叉测试。

- 用户测试：很难进行全面的功能性测试，其他的性能、并发等方面的测试比较困难。

2)系统测试的主要内容

包括对功能、性能、以及软件所运行的软硬件环境进行测试。

目前系统测试主要由黑盒测试工程师在系统集成完毕后进行测试，前期主要测试系统的功能是否满足需求，后期主要测试系统运行的性能是否满足需求，以及系统在不同的软硬件环境中的兼容性等。

3)常见的系统测试方法

a)恢复测试

恢复测试指持续超过系统规格负载测试之后，再将负载恢复到规格以内的测试方法，同时，恢复测试还关注导致软件运行失败的各种条件，并验证其恢复过程能否正确执行。在特定情况下，系统需具备容错能力。另外，系统失效必须在规定时间段内被更正，否则将会导致严重的经济损失。

b)安全测试

安全测试用来验证系统内部的保护机制，对信息、数据的保护能力，以防止非法侵入。在安全测试中，测试人员扮演试图侵入系统的角色，采用各种办法试图突破防线。因此系统安全设计的准则是要想方设法使侵入系统所需的代价更加昂贵

c)压力测试

压力测试指一段时间内持续超过系统规格的负载进行测试的一种可靠性测试方法

17.容量测试与压力测试的区别

1）容量测试：目的是通过测试预先分析出某项指标的极限值，系统在极限值状态下能保持正常运行。

容量测试是面向数据的，目的是显示系统可以处理确定的数据容量。

2）压力测试：压力测试是在强负载下的测试，查看应用系统在峰值使用情况下操作行为，从而发现系

统的某项功能隐患、系统是否具有良好的容错能力和可恢复能力。压力测试分为高负载下的长时间

的稳定性压力测试和极限负载情况下导致系统崩溃的破坏性压力测试。

18.验收测试

验收测试指按照项目任务书或合同、供需双方约定的验收依据文档进行的对整个系统的测试与评审，决定是否接收或拒收系统。在系统测试的后期，以用户测试为主或有测试人员等质量保证人员共同参与的测试。

- α 测试：指的是指的是由用户，测试人员、开发人员等共同参与的内部测试。
- β 测试：指的是内测后的公测，即完全交给最终用户测试
- 正式的验收测试

验收测试的重要性：验收签字,收钱。

19.α 测试和 β 测试的不同

α 测试：指的是指的是由用户，测试人员、开发人员等共同参与的内部测试。
β 测试：指的是内测后的公测，即完全交给最终用户测试

20.如何组织软件测试团队
开放题

21 软件测试人员的培养方法
开放题

22.文档测试主要测试内容

文档测试：主要测试用户手册和需求说明是否真正符合用户的实际需求。

- 测试开发过程中生成的文档，以需求规格说明、软件设计、用户手册、安装手册等为主，检验文档是否和实际存在差别。文档测试不需要编写测试用例。

23.软件测试

1． 是什么

IEEE:

1）在特定的条件下运行系统或构件，观察或记录结果，对系统的某个方面做出评价。（操作）

2）分析某个软件项以发现和现存的和要求的条件之差别（即错误）并评价此软件项的特性。（目的）

公式定义

测试 = 检测已知的 + 试验未知的

测试 = 自动化测试（基于确定性模型的、脚本的 ） + 探索式测试（基于搜索的、启发式的 AI 测试，即随机 / 变异的工具测试）

---

2 目的

有前辈的评论 pool

现在总结为：（注记：风险-开发-用户）

以最少的人力、物力、时间找出软件中潜在的各种错误和缺陷，通过修正错误和缺陷提高软件质量，回避潜在的软件错误和缺陷给软件造成的商业风险。

2. 通过分析测试过程中发现的问题可以帮助发现当前开发工作所采用的软件过程的缺陷，以便进行软件过程改进；同时通过对测试结果的分析整理，可修正软件开发规则，并为软件可靠性分析提供相关的依据。

3. 评价程序或系统的属性，对软件质量进行度量和评估，以验证软件的质量满足用户的需求，为用户选择、接受软件提供有力的依据。

---

3 关键问题

注记：zw 尽早重视，避免群集，定期全面检查，保护经济

测试人员应该按照软件测试的原则(Principle)开展测试活动：

软件测试是证伪而非证真

软件测试是为了发现错误而执行程序的过程，软件测试完成并不能说明软件已经不存在问题了。

2. 尽早地和不断地进行软件测试

软件开发各个阶段工作的多样性以及参加开发各层次人员之间工作的配合关系等因素，使得开发的各个环节都可能产生错误。应在软件开发的需求分析和设计阶段就开始测试工作，坚持在各个环节进行技术评审和验证，这样才能尽早发现错误，以较低的代价修改错误。

3. 重视无效数据和非预期使用习惯的测试

测试用例的编写不仅应当根据有效和遇到的输入情况，而且也应当根据无效的和未遇到的输入情况来设计。

4. 程序员应该避免检查自己的程序

程序员与软件产品有着直接的利益关系，有很多理由支持这个原则。测试工作需要严格的作风，客观的态度和冷静的情绪。但是心理学告诉我们，人们具有一种不愿意否定自己的自然性心理，这是做好软件测试的一大心理障碍.

5. 充分注意测试中的群集现象

测试时不要以为找到了几个错误就已经解决，不需继续测试了。经验表明，测试后残存的错误数目与该程序中已经发现的错误数目成正比。根据这个规律，应当对错误群集的程序段进行重点测试

6. 用例要定期评审

测试用例多次重复使用后，其发现缺陷的能力会逐渐降低。为了克服这种现象，测试用例需要进行定期评审和修改，同时需要不断增加新的不同的测试用例来测试软件或者系统的不同部分，从而发现更多的潜在错误

7. 应当对每一个测试结果做全面检查

这是一条最明显的原则，但常常被忽视。有些错误的征兆在输出实例结果时就已经明显地出现了，但是如果不仔细全面地检查测试结果，就会使这些错误或结果被遗漏掉.

8. 测试现场保护和资料归档

出现问题时要保护好现场，并记录足够的测试信息，以备缺陷能够复现。妥善保存测试计划，测试用例，出错统计和最终分析报告，为以后产品的升级测试提供足够的价值信息.

9. 软件测试的经济型原则

软件测试是保证软件质量的一个重要环节，其目的是尽可能多的找出软件中的缺陷，但是穷尽测试又是不可能的。所以在实际项目中，考虑时间、费用、人员等因素，软件测试应该适可而止。

---

4 软件质量保证

概念：是贯穿软件项目整个生命周期的有计划和有系统的活动，经常针对整个项目质量计划执行情况进行评估，检查和改进，向管理者、顾客或其他方提供信任，确保项目质量与计划保持一致。

目的：确保软件项目的过程遵循了对应的标准及规范要求且产生了合适的文档和精确反映项目情况的报告，其目的是通过评价项目质量建立项目达到质量要求的信心。

活动：软件质量保证活动主要包括评审项目过程、审计软件产品，就软件项目是否真正遵循已经制定的计划、标准和规程等，给管理者提供可视性项目和产品可视化的管理报告。

25.恢复测试

恢复测试，英文是Recovery testing。

恢复测试是测试一个系统从如下灾难中能否很好地恢复，如遇到系统崩溃、硬件损坏或其他灾难性问题。恢复测试指通过人为的让软件（或者硬件）出现故障来检测系统是否能正确的恢复，通常关注恢复所需的时间以及恢复的程度。

恢复测试主要检查系统的容错能力。当系统出错时，能否在指定时间间隔内修正错误并重新启动系统。恢复测试首先要采用各种办法强迫系统失败，然后验证系统是否能尽快恢复。对于自动恢复需验证重新初始化（reinitialization）、检查点(checkpointing mechanisms)、数据恢复(data recovery)和重新启动 (restart)等机制的正确性；对于人工干预的恢复系统，还需估测平均修复时间，确定其是否在可接受的范围内。

26.强度测试

27.正确性测试

28.Web 测试

什么是web测试

web测试就是针对于B/S架构的系统，一般指浏览器访问服务器，比如打开淘宝购买东西就是web测试。

29.条件组合覆盖

条件组合覆盖

一、定义：

> 判定中条件的各种组合都至少被执行一次

二、特点：

> 1、满足条件组合覆盖的用例一定满足语句覆盖
> 2、满足条件组合覆盖的用例一定满足条件覆盖
> 3、满足条件组合覆盖的用例一定满足判定覆盖
> 4、满足条件组合覆盖的用例一定满足条件判定覆盖
> 5、条件组合覆盖没有考虑各判定结果（真或假）组合情况，不满足路径覆盖
> 6、条件组合数量大，设计测试用例的时间花费较多

30.软件可靠性

软件可靠性是软件产品在规定的条件下和规定的时间区间完成规定功能的能力。规定的条件是指直接与软件运行相关的使用该软件的计算机系统的状态和软件的输入条件，或统称为软件运行时的外部输入条件；规定的时间区间是指软件的实际运行时间区间；规定功能是指为提供给定的服务，软件产品所必须具备的功能。

31.软件缺陷

软件缺陷定义:

- 软件未达到产品说明书中标明的功能。
- 软件出现了产品说明书中指明的不会出现的功能。
- 软件功能超出了产品说明书中指明的范围。
- 软件未达到产品说明书中指明应达到的目标。
- 软件测试人员认为软件难以理解和使用、运行速度慢，或最终用户认为不好。

符合以上任意一种情况，即为软件缺陷。

软件缺陷属性

发现缺陷后，需要提交缺陷单，通常情况下，缺陷单需要包含以下的内容：

- ID
- 标题(Title)
- 测试环境(Environment)
- 严重等级(Severity)
- 优先级(Priority)
- 类别(Category)
- 状态(Status)
- 描述信息(Description)
- 重现步骤(Reproduce)
- 附件(Attachment)
- 测试人员(Created by)
- 处理人员（Assign to）
- …

How – 如何进行缺陷管理

不同成熟度的软件组织采用不同的缺陷管理方式。

- 低成熟度的软件组织会记录缺陷，并跟踪缺陷纠正过程。
- 高成熟度的软件组织，还会充分利用缺陷提供的信息，建立组织过程能力基线，实现量化过程管理，并可以此为基础，通过缺陷预防实现过程的持续性优化。

缺陷管理一般要借助于缺陷管理工具。

32. 测试用例

测试用例，英文为TestCase，缩写为TC，指的是在测试执行之前设计的一套详细的测试方案，包括测试环境、测试步骤、测试数据和预期结果。

测试用例设计的好坏直接决定了测试的效果和结果。所以说在软件测试活动中最关键的步骤就是设计有效的测试用例。

测试用例可以针对黑盒测试设计用例，也可以针对白盒测试设计用例.

根据什么写测试用例呢？

我们编写测试用例的唯一标准就是**用户需求**，具体的参考资料是《需求规格说明书》，但需要说明的是，用户需求不是一成不变的，而是在一直变化的，这就需要我们根据不断调整变化的需求，来修改和维护我们已写好的测试用例，这个工作量也很大。

33. 变异测试

变异测试定义

- 变异测试也称为“变异分析”，是一种对测试数据集的有效性、充分性进行评估的技术，能为研发人员开展需求设计、单元测试、集成测试提供有效的帮助
- 变异测试通过对比源程序与变异程序在执行同一测试用例时差异来评价测试用例集的错误检测能力
- 在变异测试过程中，一般利用与源程序差异极小的简单变异体来模拟程序中可能存在的各种缺陷

变异测试应用场景

- 在软件测试时，若当前测试用例未能检测到软件缺陷，则存在两种情形：
  ① 软件已满足预设的需求，软件质量较高；
  ② 当前测试用例设计不够充分，不能有效检测到软件中的缺陷。
- 逻辑测试和路径测试方法，分别从程序实体覆盖和路径覆盖的角度来评估软件测试的充分性。然而，这些方法并不能直观地反映测试用例的缺陷检测能力
- 变异测试方法可用于度量测试用例缺陷检测能力

变异测试过程

- 在变异测试过程中，程序与变异程序的执行差异主要表现为以下两个情形：
  - (1) 执行同一测试用例时，源程序和变异程序产生了不同的运行时状态
  - (2) 执行同一测试用例时，源程序和变异程序产生了不同的执行结果
- 根据满足执行差异要求的不同，可将变异测试分为弱变异测试 (Weak Mutation Testing) 和强变异测试 (Strong Mutation Testing)
- 在弱变异测试过程中，当情形 (1) 出现时就可认为变异程序被杀死，而在强变异测试过程中，只有情形 (1) 和 (2) 同时满足才可认为变异程序被杀死

34. 回归测试

回归测试(regression testing)

是指软件被修改后重新进行的测试，如重复执行上一个版本测试时的用例，是为了保证对软件所做的修改没有引入新的错误而重复进行的测试。

35. 兼容性测试

兼容性测试

兼容性测试是指测试软件在特定的硬件平台上、不同的应用软件之间、不同的操作系统平台上、不同的网络等环境中是否能够很友好的运行的测试。

36. 第三方测试

概念：第三方测试也称独立测试，意思为独立于软件开发方与用户方。

> 第三方测试是指介于软件开发方和用户方之间的测试组织的测试，第三方测试也称为独立测试，它有独立的验证和确认活动。在模拟用户真实应用环境下，进行软件确认测试。

37. 冒烟测试

冒烟测试（smoke testing）

是指在对一个新版本进行系统大规模的测试之前，先验证一下软件的基本功能是否实现，是否具备可测试性。

38. 确认测试

确认测试也称为验收测试，它的**目标是验证软件的有效性**。
通常，验证指的是**保证软件正确地实现了某个特定要求的一系列活动**；**确认指的是为了保证软件确实满足了用户需求而进行的一系列活动。**
软件有效性的一个简单定义是：如果软件的功能和性能如同用户所合理期待的那样，软件就是有效的。
需求分析阶段产生的**软件需求规格说明书**，准确地描述了用户对软件的合理期望，因此是**软件有效性的标准，也是进行确认测试的基础。**

39. 性能测试

二、性能测试概念

- 是指通过**模拟**生产运行的业务压力或用户使用场景来测试系统的性能**是否满足生产性能**的要求。
- 性能测试是一种**“正常”测试**，主要测试使用时系统是否满足要求，同时可能为了保留系统的扩展空间而进行的一些**稍稍超过“正常”范围**的测试（比如：当前系统使用用户100人，可能未来人数会增多到300人，所以要让系统能够在300人情况下正常运行）

40. 压力测试

四、压力测试

- 逐步增加系统负载，测试系统性能的变化，并最终确定在什么负载下系统性能**处于失效状态**，并以此来获得系统能提供的**最大服务级别的测试**

41. 负载测试

三、负载测试

- 是通过逐步增加系统负载，测试系统性能的变化，并在**满足最终确定性能指标的情况下**，系统所能承受的**最大负载量**的测试
  性能指标：是系统应该满足的，比如请求响应时间等
  负载测试是正常范围的测试

五、压力测试与负载测试两者区别

相同点：都是性能测试
负载测试强调系统**正常工作**情况下的性能指标
压力测试的目的是发现在什么条件下系统的性能变得**不可接受**，发现应用程序性能下降的拐点。

举例：工人建桥，**桥身上表明**，该桥的最大负重量为**60吨。**—负载测试
该桥的**内部建筑资料**中，表明该桥的最大载重量为**70吨。**这个数据是给内部建桥工程师掌握的。—压力测试

42. 安全测试

- 在应用投产前，应由独立的安全团队对应用的安全性进行综合评估
  - 功能性安全测试
  - 对抗性安全测试
- 传统测试方法
  - 白盒测试
  - 黑盒测试
  - 灰盒测试
- 特定的安全测试手段
  - 模糊测试
  - 渗透测试

为什么要软件安全测试？

- 软件测试
  - 按照特定规程，发现软件错误的过程。
  - 检查软件是否满足规定的要求，或是清楚地了解预期结果与实际结果之间的差异
  - 其目的在于发现软件中的错误。
- 软件安全测试
  - 有关验证软件安全等级和识别潜在安全缺陷的过程
  - 查找软件自身程序设计中存在的安全隐患，并检查应用程序对非法侵入的防范能力
  - 传统测试仅考虑软件出错时的处理，没有考虑对软件的故意攻击

43. 自动化测试

4.1测试自动化的内涵

4.1.1 为什么要软件测试自动化

- 自动化测试（automated test）是相对手工测试（manual test）而存在的一个概念，由手工逐个地运行测试用例的操作过程被测试工具自动执行的过程所代替。

- 测试工具的使用是自动化测试的主要特征。

4.1测试自动化的内涵

4.1.1 为什么要软件测试自动化

自动化测试 vs 测试自动化

❖**自动化测试**焦点集中在测试执行，主要是由测试工具自动地完成测试。

❖**测试自动化**指“一切可以由计算机系统自动完成的测试任务都已经由计算机系统或软件工具、程序来承担并自动执行”

44. 软件质量保证

软件质量保证

软件质量保证是**贯穿软件项目整个生命周期**的有计划和有系统的活动，经常针对整个项目质量计划执行情况进行评估，检查和改进，向管理者、顾客或其他方提供信任，确保项目质量与计划保持一致。

确保软件项目的过程**遵循了对应的标准及规范要求且产生了合适的文档和精确反映项目情况的报告**，其目的是通过评价项目质量建立项目达到质量要求的信心。软件质量保证活动主要包括评审项目过程、审计软件产品，就软件项目是否真正遵循已经制定的计划、标准和规程等，给管理者提供可视性项目和产品可视化的管理报告。

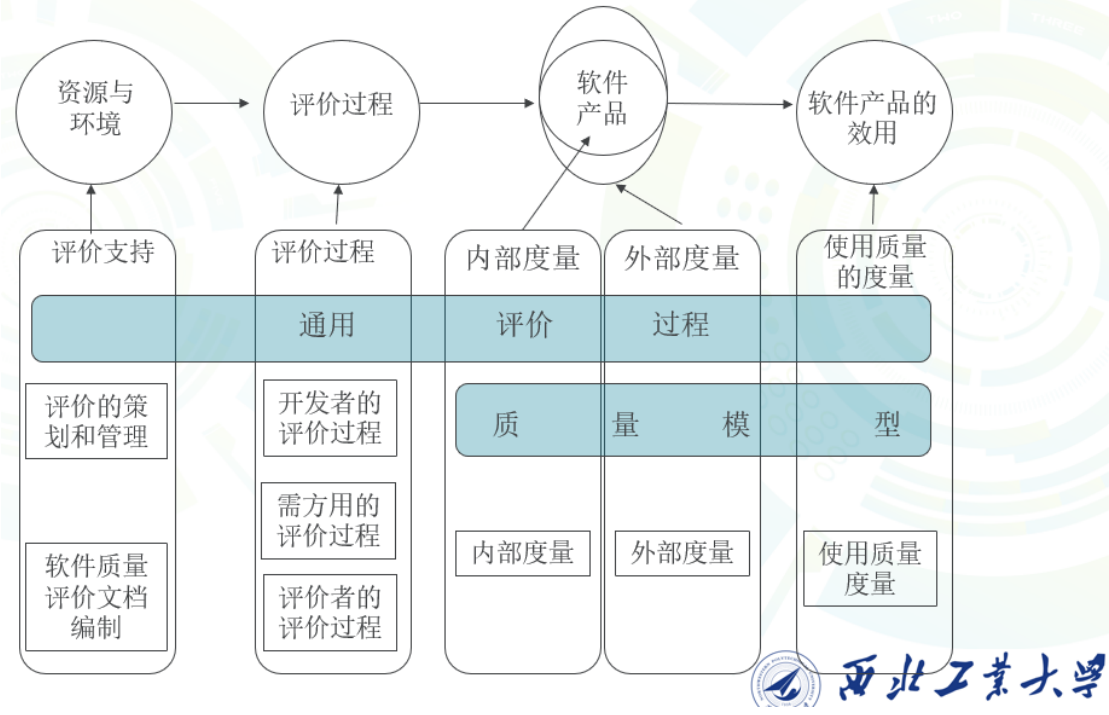
软件质量保证与软件测试

- 评价、度量和测试在技术内容上有着非常重要的关系。**软件测试是获取度量值的一种重要手段**。软件度量在GJB 5236主要规定软件质量模型和内部质量度量、外部质量度量以及使用质量的度量，可用于在确定软件需求时规定软件质量需求或其他用途。

- 软件质量评价在GJB 2434A则针对开发者、需求方和评价者提出了3种不同的评价过程框架。在执行软件产品评价时，**确立评价需求的质量模型就需要采用GJB 5236给出的内部度量、外部度量、使用质量的度量**等。

- 这两个系列标准的关系如下页图所示，从图中可以看出GJB 2434A和GJB 5236的联系是非常密切的，需要有机地结合起来才能有效完成软件产品的度量和评价工作。其中，**度量值的获取主要来自软件测试**。可以说评价依据度量，而度量依据测试。也可以说评价指导度量，度量指导测试。

软件质量保证与软件测试

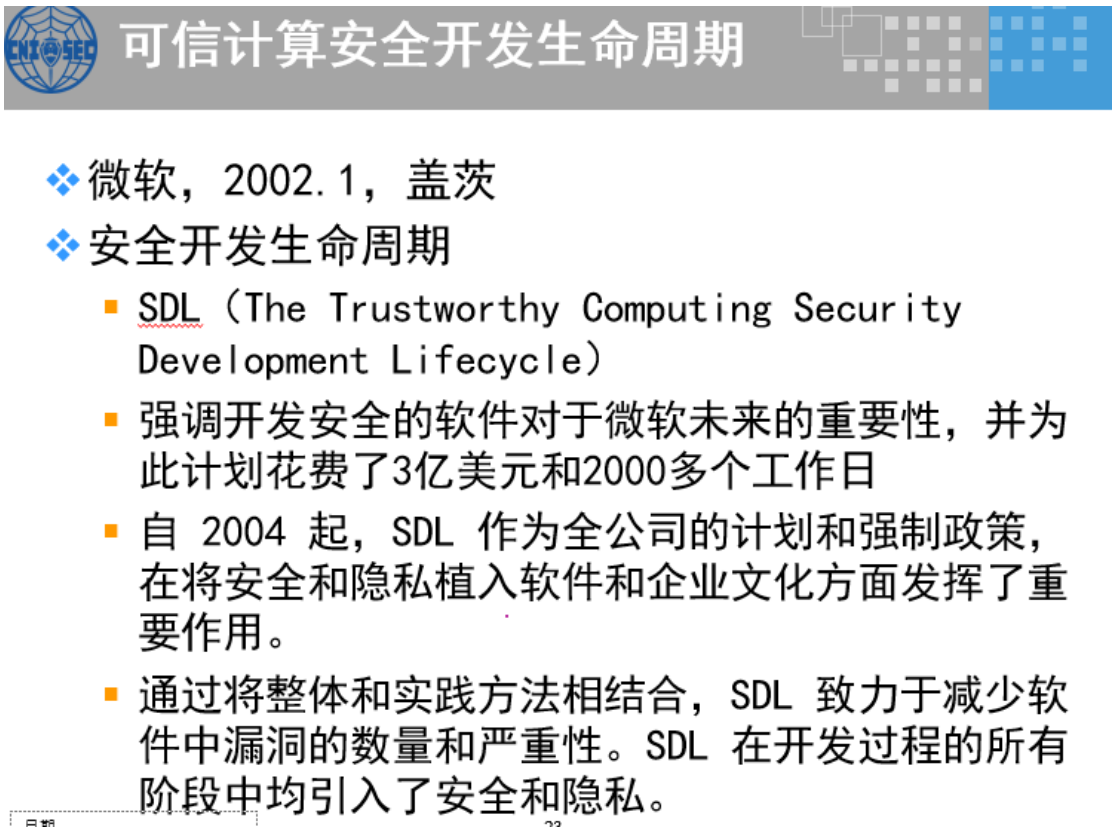


45. 逻辑覆盖

- 根据覆盖目标的不同，逻辑覆盖又可分为语句覆盖、判定覆盖、条件覆盖、判定/条件覆盖、组合覆盖和路径覆盖。

- 语句覆盖：选择足够多的测试用例，使得程序中的每个可执行语句至少执行一次。
- 判定覆盖：通过执行足够的测试用例，使得程序中的每个判定至少都获得一次“真”值和“假”值， 也就是使程序中的每个取“真”分支和取“假”分支至少均经历一次，也称为“分支覆盖”。
- 条件覆盖：设计足够多的测试用例，使得程序中每个判定包含的每个条件的可能取值（真/假）都至少满足一次。

46. SDL



SDL使用

- SDL 是一个安全保证过程，其在开发过程的所有阶段中引入了安全和隐私原则。
- Microsoft 将 SDL 与软件行业和客户开发组织自由共享，供他们用来开发更为安全的软件。
- 如何使用SDL？
  - 为了实现所需安全和隐私目标，项目团队或安全顾问可以自行决定添加可选的安全活动
  - 开发团队应以SDL指南为指导，实施SDL的时候结合考虑组织的时间、资源和业务运营方式
  - Cisco、EMC、Symantec等安全公司均借鉴微软SDL中的做法——CSDL

SDL的阶段和安全活动

- 软件安全开发生命阶段
  - 5+2个阶段
  - 16项必需的安全活动

48. 模糊测试

- 模糊测试（Fuzz测试）
  - Baron Miller、Lars Fredriksen、Bryan So首次提出
  - 是一种通过提供非预期的输入并监视异常结果来发现软件故障的方法
- 属于黑盒测试
  - 不关心被测试目标的内部实现
  - 设计输入，检测结果，发现安全漏洞
- 微软SDL包含了Fuzz测试

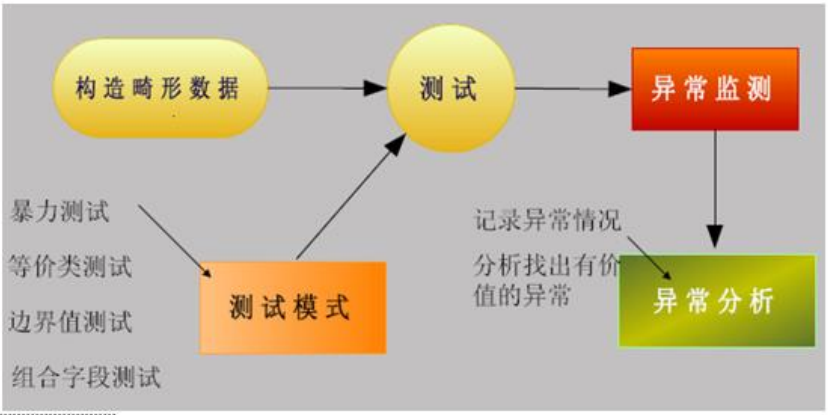
非常有效的漏洞挖掘技术，已知漏洞大部分都是通过这种技术发现的。

Fuzz测试

- 强制软件程序使用恶意/破坏性的数据并进行观察结果的一种测试方法
  - 不够强壮的程序会崩溃
  - 编码良好的程序正常运行
- 特性
  - 方法学
  - 随机值
  - 大量测试用例
  - 查找漏洞或可靠性错误

Fuzz测试步骤

- 生成大量的畸形数据作为测试用例
- 将这些测试用例作为输入应用于被测对象
- 监测和记录由输入导致的任何崩溃或异常现象
- 查看测试日志，深入分析产生崩溃或异常的原因



49. 渗透测试

❖渗透测试的概念

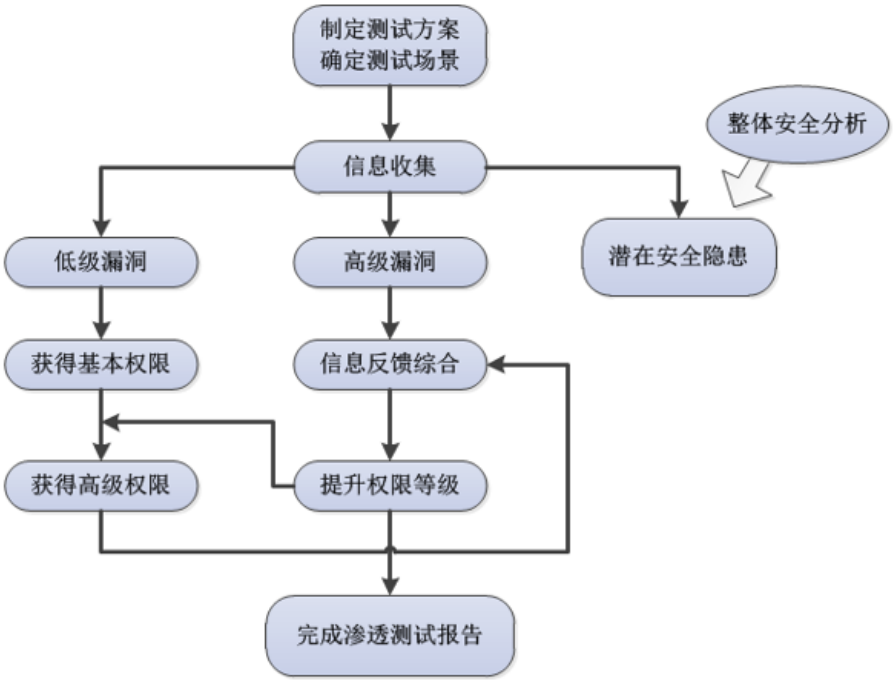
- 渗透测试是通过模拟攻击方法，来评估对象（系统或产品）安全的一种方法。

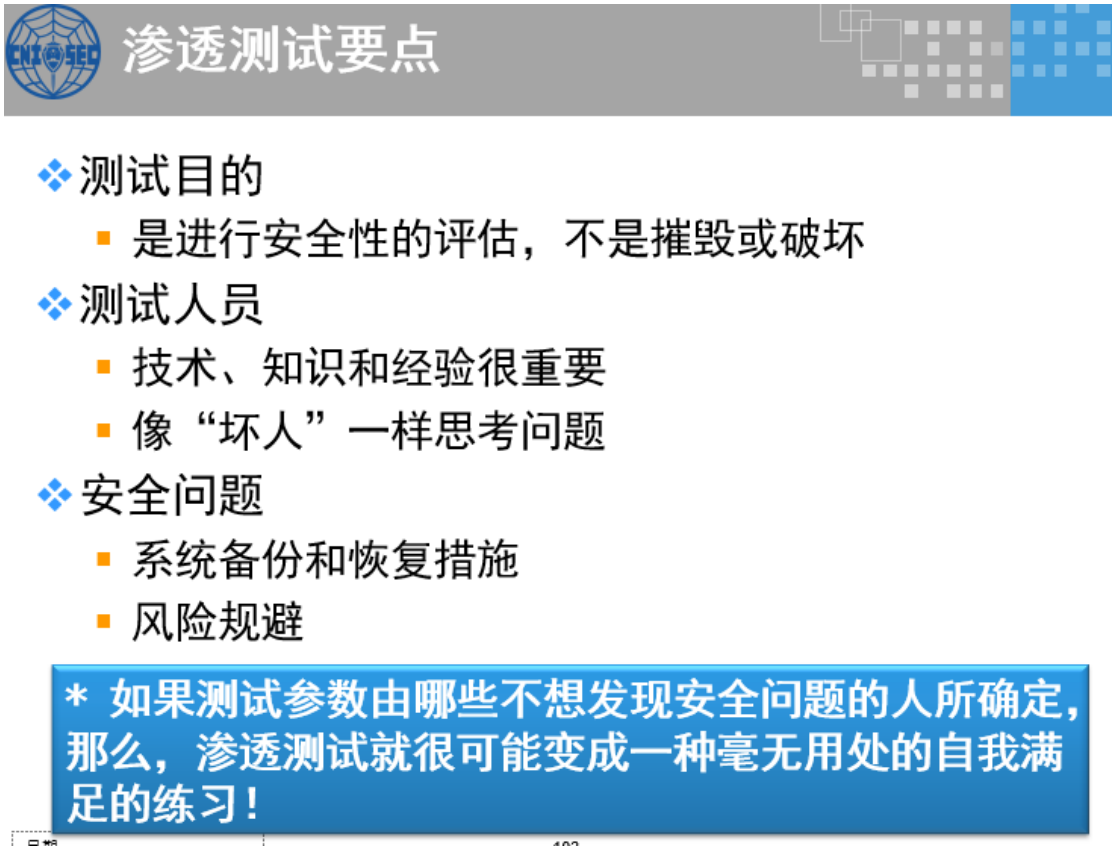
❖渗透测试的优势

- 测试是基于软件运行的最后环境，因此，除了可以发现软件本身的安全问题外，更重要的是还可以发现一些关于环境和配置的安全问题。

- 渗透测试
  - 通过模拟恶意黑客的攻击方法，来评估系统安全的一种评估方法
  - 从攻击的角度测试软件系统是否安全
  - 使用自动化工具或者人工的方法模拟黑客的输入，找出运行时刻目标系统所存在的安全漏洞
- 优点
  - 找出来的问题都是真实的，也是较为严重的
- 缺点
  - 只能到达有限的测试点，覆盖率较低

渗透测试流程





50. SBSE

在SBSE（Search-Based Software Engineering）中，术语“搜索”是指使用的基于元启发式搜索的优化(SBO)技术

SBSE试图将SE问题重新表述为SBO问题(简称“搜索问题”)。

一个搜索问题是在候选解决方案的搜索空间中寻找最优或接近最优的解决方案，由适应度函数指导。