软件工程中的测试技术

第一部分

目录

- (1) 软件测试的基本概念
- (2) 软件测试类型及其在软件开发过程中的地位
- (3) 代码检查、走查与评审
- (4) 覆盖率 (白盒) 测试
- (5) 功能 (黑盒) 测试

软件测试的基本概念

- 1.1 软件质量的概念
  - 定义

    软件质量是产品、组织和体系或过程的一组固有特性，应映它们满足顾客和其他相关方面要求的程度。
  - 属性

    满足客户的功能需求、性能需求（处理和响应时间）、可扩展性、灵活性、适应一定程度的需求变化等。

软件测试的基本概念

- 1.1 软件质量的概念
  - 模型

    Boehm 质量模型

    McCall 质量模型

    ISO 的软件质量模型
  - 度量

    外部度量、内部度量、内部度量和外部度量之间的关系、使用质量的度量

软件测试的基本概念

- 1.2 软件测试的概念
  - 定义与目的

    在规定条件下运行系统或构件的过程，在此过程中观察和记录结果，并对系统或构件的某些方面给出评价。

    分析软件项目的过程，检测现有状况和所需状况之间的不同（即 bug ），并评估软件项目的特征。

软件测试的基本概念

- 1.2 软件测试的概念
  - 定义与目的

    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。如果成功地实施了测试，就能发现软件中的错误。

    测试的另一收获是，它能证明软件的功能和性能与需求说明相符合。

软件测试的基本概念

- 1.2 软件测试的概念
  - 原则
    - 应当把“尽早地和不断地进行软件测试”作为软件开发人员的座右铭。
    - 测试用例应由测试输入和与之对应的预期输出结果这两部分组成
    - 程序员应避免测试自己的程序
    - 在设计测试用例时，应当包括合理的输入条件和不合理的输入条件

软件测试的基本概念

- 1.2 软件测试的概念
  - 原则
    - 充分注意测试中的群集现象
    - 严格执行测试计划，排除测试的随意性
    - 应当对每一个测试结果做全面检查
    - 妥善保存测试计划、测试用例、出错统计和最终分析报告

软件测试的基本概念

- 1.3 软件的缺陷与错误
  - 缺陷的定义和类型

    IEEE729-1983 对缺陷的定义：

    (1) 从产品内部看，缺陷是软件产品开发或维护过程中存在的错误、毛病等各种问题。

    (2) 从产品外部看，缺陷是系统所需要实现的某种功能的失效或违背。

软件测试的基本概念

- 1.3 软件的缺陷与错误
  - 缺陷的级别
    - 致命的（ Fatal ）
    - 严重的（ Critical ）
    - 一般的（ Major ）
    - 微小的（ Minor ）

软件测试的基本概念

- 1.3 软件的缺陷与错误
  - 缺陷的产生原因
    - 技术问题
    - 团队工作
    - 软件本身

软件测试的基本概念

- 1.3 软件的缺陷与错误
  - 修复缺陷的代价

    如果在需求阶段修正一个错误的代价是 1 ，那么

    设计阶段： 3~6 倍

    编程阶段： 10 倍

    内部测试： 20~40 倍

    外部测试： 30~70 倍

    产品发布： 40~1000 倍

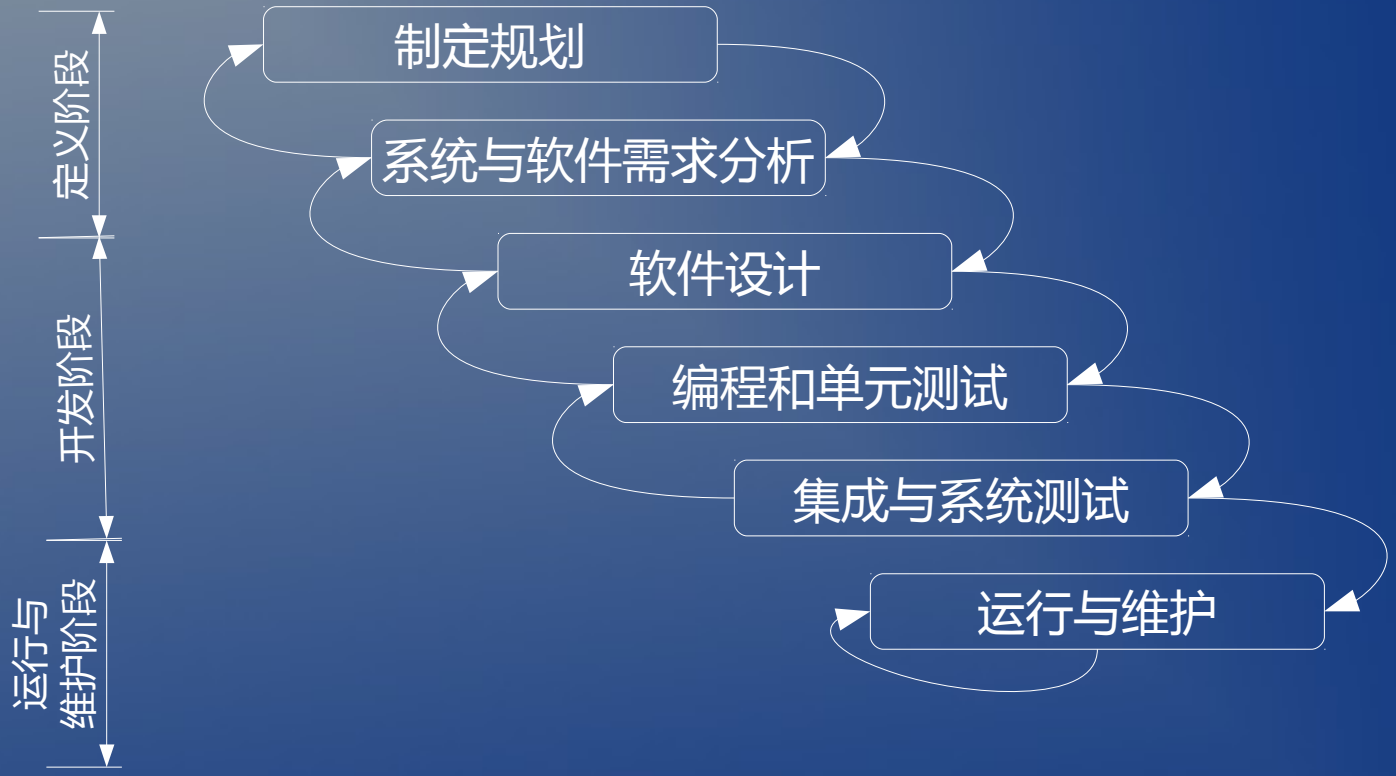
目录

- (1) 软件测试的基本概念
- (2) 软件测试类型及其在软件开发过程中的地位
- (3) 代码检查、走查与评审
- (4) 覆盖率 (白盒) 测试
- (5) 功能 (黑盒) 测试

软件测试类型
及其在软件开发过程中的地位

- 2.1 软件开发阶段
  - 软件生存周期
    - 制定规划
    - 系统与软件需求定义
    - 软件设计
    - 编程和单元测试
    - 集成与系统测试
    - 运行和维护

软件测试类型
及其在软件开发过程中的地位

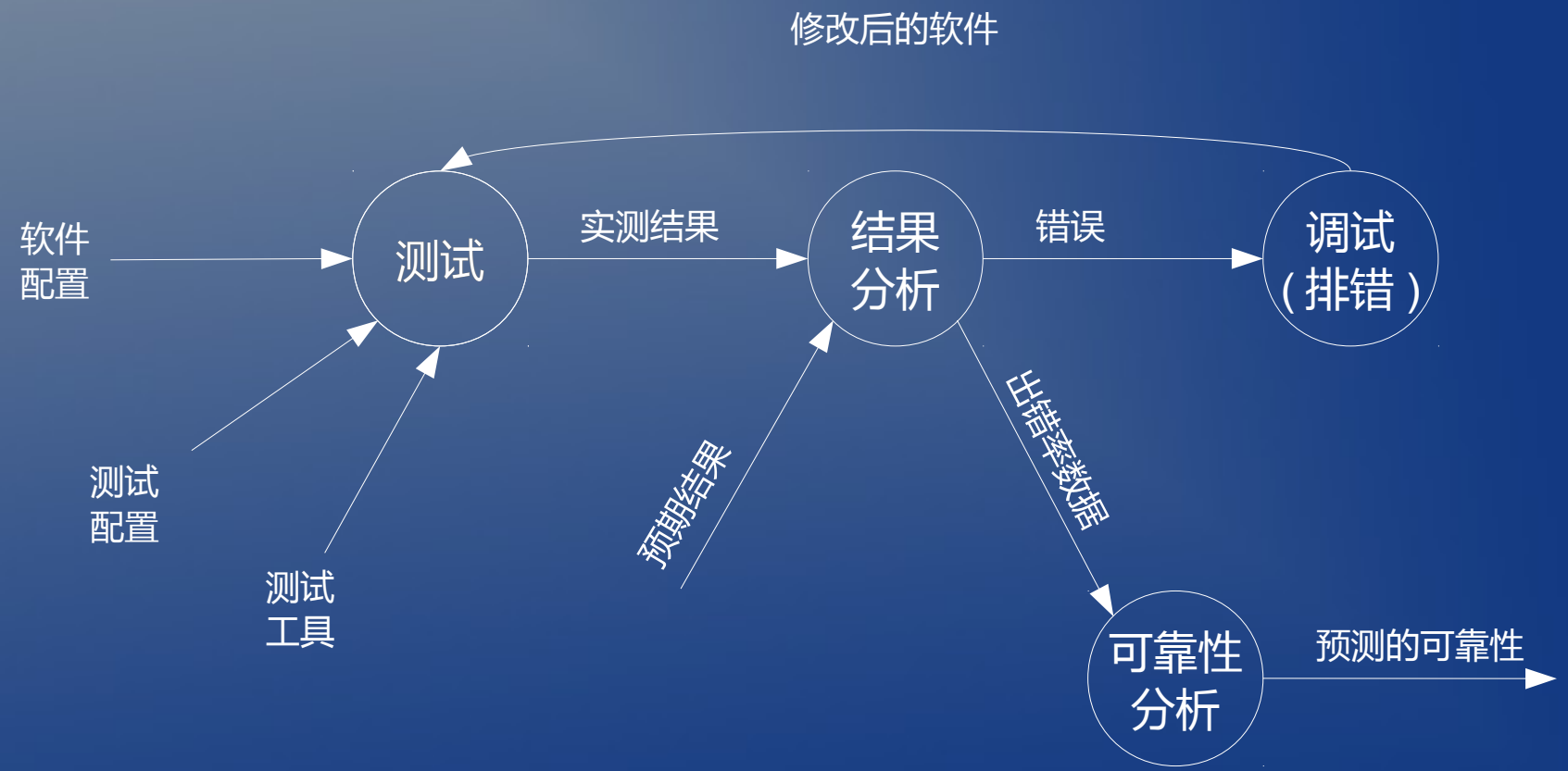


软件测试类型及其在软件开发过程中的地位

- 2.1 软件开发阶段
  - 软件测试的生存周期模型
    - 制定测试计划
    - 测试设计
    - 实施测试
    - 单元测试
    - 集成测试
    - 系统测试
    - 评估测试

软件测试类型及其在软件开发过程中的地位

- 2.1 软件开发阶段
  - 测试信息流



软件测试类型及其在软件开发过程中的地位

- 2.2 规划阶段的测试
  - 规划阶段进行的测试
    - 在规划阶段，测试的对象是规划人员的构想，不是代码。
    - 测试人员（即评审人员）包括营销人员、产品经理、设计人员和人类工程分析师。
    - 应该从六个方面来评价需求规格说明和基于需求的功能定义。

软件测试类型
及其在软件开发过程中的地位

- 2.3 设计阶段的测试
  - 外部设计
    - 外部设计主要是设计用户界面。
    - 《外部设计规格说明》和《用户手册》
    - 基本代码正确无误不能代表设计是良好的。
  - 内部设计
    - 结构设计
    - 数据设计
    - 逻辑设计

软件测试类型及其在软件开发过程中的地位

- 2.3 设计阶段的测试
  - 设计阶段的测试
    - 设计阶段测试的对象来自设计文档。
    - 外部设计（用户界面设计；与其他系统元素的接口设计；系统构件部署设计）的规格说明。
    - 内部设计（功能设计、系统体系结构设计、数据设计）的规格说明。
    - 逻辑设计（模块算法与数据结构设计）的规格说明。

软件测试类型
及其在软件开发过程中的地位

- 2.3 设计阶段的测试
  - 伪代码分析

```
IF ASCII_CODE_OF_ENTERED_CHAR is less than 48
THEN reject it
ELSE IF ASCII_CODE_OF_ENTERED_CHAR is greater than 57
THEN reject it
ELSE it's a digit, so accept it
```

软件测试类型及其在软件开发过程中的地位

- 2.4 编程阶段的测试
  - 白盒测试与黑盒测试
    - 黑盒测试将程序视为一个黑盒子，无法看到里面的东西。测试人员提供输入数据，观察输出数据，并不了解或是佯装不了解程序是如何运行的。
    - 白盒测试中，程序员要运用自己的理解能力，深入到源程序中进行测试。

软件测试类型及其在软件开发过程中的地位

- 2.4 编程阶段的测试
  - 结构测试与功能测试
    - 结构测试属于白盒测试，关注的是在一连串的测试中如何选择合适的程序或子程序路径来执行有效的检查。
    - 功能测试是黑盒测试的一种，对功能测试是通过提供输入数据、检查实际输出的结果来实现的，一般很少考虑内部的程序结果。

软件测试类型
及其在软件开发过程中的地位

- 2.4 编程阶段的测试
  - 路径测试：覆盖准则
    - 所谓路径，可定义为从程序入口到最后出口的一个操作序列。所谓子路径就是一组从程序某处到另一处的语句。
    - 常用的覆盖准备是
      - 语句覆盖
      - 分支覆盖
      - 条件覆盖

软件测试类型及其在软件开发过程中的地位

- 2.4 编程阶段的测试
  - 增量测试与大突击测试
    - 增量测试先对程序的每个程序单元或程序部件单独进行测试，一旦单个程序部件都能运行，就将其中有调用关系的一些程序部件集成在一起。一组一组的模块结合起来进行测试，直到最后，程序的所有模块都被组合起来进行测试。
    - 大突击测试是将所有模块一次性集成为一个完整的系统后进行完全测试。

软件测试类型
及其在软件开发过程中的地位

- 2.4 编程阶段的测试
  - 自顶向下测试与自底向上测试
    - 自顶向下和自底向上的测试策略都是增量式的。
    - 自顶向下的测试无须编写驱动模块。
    - 自底向上的测试无须编写桩模块。
  - 静态测试与动态测试
    - 静态测试不必执行程序。
    - 动态测试需要执行程序。
    - 白盒测试和黑盒测试都是动态测试。
  - 性能测试

软件测试类型及其在软件开发过程中的地位

- 2.5 回归测试
  - 回归测试有两种不同的使用方式，都是建立在复用原有测试的思想之上的：
    - （1）一经发现并改正了程序中的隐藏缺陷，然后再重新执行以前发现这个缺陷的测试，看这个缺陷是否重现。
    - （2）当对发现的缺陷进行修改之后，执行一系列基准测试，以确认程序的修改没有对程序的其他部分产生干扰。

软件测试类型
及其在软件开发过程中的地位

- 2.6 运行和维护阶段的测试
  - 在维护阶段所做的大部分测试与功能测试和确认测试阶段所做的相似。理想情况下，每次软件发生变更后都要执行回归测试。

目录

- (1) 软件测试的基本概念
- (2) 软件测试类型及其在软件开发过程中的地位
- (3) 代码检查、走查与评审
- (4) 覆盖率 (白盒) 测试
- (5) 功能 (黑盒) 测试

代码检查、走查与评审

- 3.1 桌上检查
  - 检查项目
    - 检查变量的交叉引用表，检查标号的交叉引用表，检查子程序、宏、函数，等价性检查，常量检查，标准检查，风格检查，比较控制流，对照程序的规格说明详细阅读源代码，补充文档。
  - 静态错误分析
    - 生成各种引用表
    - 静态错误分析

代码检查、走查与评审

- 3.2 代码检查
  - 代码检查过程
    - 代码检查要求人们组成一个小组来阅读或直观检查特定的程序。
    - 代码检查是对桌上检查过程的改进。
    - 主要进行两项活动
      - 由程序编码人员逐条语句讲述程序的逻辑结构。
      - 对照常见编码错误列表分析程序。

代码检查、走查与评审

- 3.3 走查
  - 什么是“走查”
    - 走查与代码检查很相似，都是以小组为单位进行，是一系列规程和错误检查技术集合。
  - “ 走查”的目标
    - 发现缺陷、遗漏和矛盾的地方；改进产品；考虑可替换的实现方法。

代码检查、走查与评审

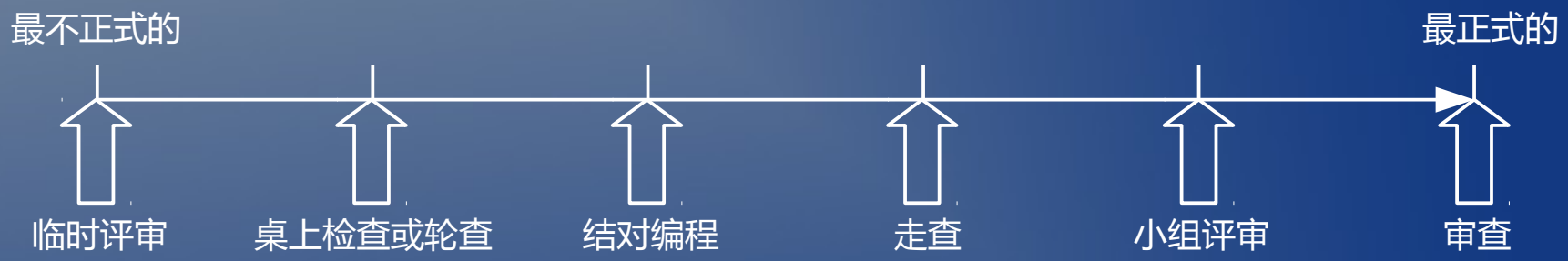
- 3.3 走查
  - 走查的过程
    - 计划走查会议
    - 走查产品
    - 执行走查
    - 解决缺陷
    - 走查记录
    - 产品返工

代码检查、走查与评审

- 3.4 同行评审
  - 为什么需要评审
    - 在评审中发现产品的缺陷，因此在评审上的投入可以减少大量的后期返工。通过评审，还可以将问题记录下来，具有历史可追溯性。
  - 同行评审的内容
    - 管理评审
    - 技术评审
    - 文档评审
    - 过程评审

代码检查、走查与评审

- 3.4 同行评审
  - 评审的方法和技术



目录

- (1) 软件测试的基本概念
- (2) 软件测试类型及其在软件开发过程中的地位
- (3) 代码检查、走查与评审
- (4) 覆盖率 (白盒) 测试
- (5) 功能 (黑盒) 测试

覆盖率（白盒）测试

- 4.1 覆盖率的概念
  - 覆盖率是用于度量测试完整性的一个手段。
  - 覆盖率 = 被执行到的项数 / 项总数 ×100%
  - 通过覆盖率数据，可以知道测试得是否充分，测试的弱点在哪些方面，进而指导我们设计能够增加覆盖率的测试用例。这样就能够有效地提高测试质量，避免设计无效的测试用例。

覆盖率（白盒）测试

- (4.2) 逻辑覆盖
  - 语句覆盖
    - 所谓语句覆盖，就是设计若干个测试用例，运行被测程序，使得每一可执行语句至少执行一次。这里的“若干个”，意味着使用测试用例越少越好。
    - 语句覆盖率 = 被评价到的语句数量 / 可执行的语句总数 ×100%

覆盖率（白盒）测试

- (4.2) 逻辑覆盖
  - 判定覆盖（分支覆盖）
    - 所谓判定覆盖，就是设计若干个测试用例，运行被测程序，使得程序中每个判定的取真分支和取假分支至少评价一次。判定覆盖又称为分支覆盖。
    - 判定覆盖率 = 被评价到的判定分支的个数 / 判定分支的总数 ×100%

覆盖率（白盒）测试

- (4.2) 逻辑覆盖
  - 条件覆盖
    - 所谓条件覆盖，就是设计若干个测试用例，运行被测程序，使得程序中每个判定的每个条件的可能取值至少评价一次。
    - 条件覆盖率 = 被评价到的条件取值的数量 / 条件取值的总数 ×100%

覆盖率（白盒）测试

- (4.2) 逻辑覆盖
  - 条件 / 判定覆盖
    - 所谓条件 / 判定覆盖，就是设计足够的测试用例，使得判定语句中每个条件的所有可能取值至少评价一次，同时每个判定语句本身的所有可能分支也至少评价一次。
    - 条件 / 判定覆盖率 = 被评价到的条件取值和判定分支的数量 /（条件取值总数 + 判定分支总数）

覆盖率（白盒）测试

- (4.2) 逻辑覆盖
  - 条件组合覆盖
    - 所谓条件组合覆盖，就是设计足够的测试用例，运行被测试程序，使得每个判定的所有可能的条件取值组合至少执行一次。
    - 条件组合覆盖率 = 被评价到的条件取值组合的数量 / 条件取值组合的总数

覆盖率（白盒）测试

- (4.2) 逻辑覆盖
  - 路径覆盖
    - 路径覆盖就是设计足够的测试用例，执行程序中所有可能的路径。
    - 路径覆盖率 = 被执行到的路径数 / 程序中总的路径数
  - ESTCA 覆盖
  - LCSAJ 覆盖

覆盖率（白盒）测试

- 4.3 路径覆盖
  - 分支结构的路径测试
    - 嵌套型分支结构
      - 若有 n 个判定语句，则存在 n+1 条不同的路径，需要 n+1 个测试用例覆盖它的每一条路径。
    - 串联型分支结构
      - 若有 n 个判定语句，则有 2^n 条路径，因此需要有 2^n 个测试用例覆盖它的每一条路径。
  - 循环结构的路径测试
  - Z 路径覆盖与基本路径测试

覆盖率（白盒）测试

- 4.4 数据流测试
  - 概念
    - 数据流测试也可以看作是一种路径测试，它主要关注在一条路径上变量在何处定义（赋值），在何处使用（引用）。程序员通过监视变量的定义和使用异常来分析源代码。

覆盖率(白盒)测试

- 4.4 数据流测试
  - 定义/使用测试的几个定义
    - 定义结点
    - 使用结点
    - 谓词使用
    - 定义/使用路径
    - 定义清除路径

覆盖率(白盒)测试

- 4.4 数据流测试
  - 定义/使用路径测试覆盖指标
    - 全定义准则
    - 全使用准则
    - 全谓词使用/部分计算使用准则
    - 全计算使用/部分谓词使用准则
    - 全定义/使用路径准则

目录

- (1) 软件测试的基本概念
- (2) 软件测试类型及其在软件开发过程中的地位
- (3) 代码检查、走查与评审
- (4) 覆盖率 (白盒) 测试
- (5) 功能 (黑盒) 测试

功能（黑盒）测试

- 黑盒测试的概念
  - 软件的黑盒测试就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。

功能（黑盒）测试

- 5.1 等价类测试
  - 概念
    - 首先把数目极多的输入数据（有效的和无效的）划分为若干等价类。并合理地假定：测试某等价类的代表值就等价于对这一类其他值的测试。
  - 类型
    - 弱一般等价类测试
    - 强一般等价类测试
    - 弱健壮等价类测试
    - 强健壮等价类测试

功能（黑盒）测试

- 5.1 等价类测试
  - 原则
    - 如果输入条件规定了取值范围，或值的个数，则可以确立一个有效等价类和两个无效等价类。
    - 如果输入条件规定了输入值的集合，或者规定了“必须如何”的条件，这时可确立一个有效等价类和一个无效等价类。
    - 如果规定了输入数据的一组值，而且程序要对每个输入值分别进行处理，这时可分为每个输入值确立一个有效等价类。此外，针对这组值确立一个无效等价类，它是所有不允许的输入值的集合。

功能（黑盒）测试

- 5.1 等价类测试
  - 原则
    - 如果规定了输入数据必须遵守的规则，则可以确立一个有效等价类（符合规则）和若干个无效等价类（从不同角度违反规则）。
    - 如果确知已划分的等价类中各元素在程序中的处理方式不同，则应将此等价类进一步划分成更小的等价类。

功能（黑盒）测试

- 5.2 边界值分析
  - 概念
    - 边界值分析也是一种黑盒测试方法，是对等价类划分方法的补充。这里所说的边界值是指，对输入等价类和输出等价类而言，位于其边界的值，或稍高于其边界的值及稍低于期 边界的值这样一些特定情况。
  - 原则
    - 如果输入条件规定了值的范围，则应取刚达到这个范围的边界的值，以及刚刚超越这个范围边界的值作为测试输入数据。

功能（黑盒）测试

- 5.2 边界值分析
  - 原则
    - 如果输入条件规定了值的个数，则用最大个数、最小个数、比最大个数多 1 、比最小个数少 1 的数据作为测试数据。
    - 根据规格说明的每个输出条件，使用前面的原则 1 。
    - 根据规格说明的每个输出条件，使用前面的原则 2 。
    - 如果程序的规格说明给出的输入域或输出域是有序集合（如有序表、顺序文件等），则应选取集合的第一个元素和最后一个元素作为测试用例。

功能（黑盒）测试

- 5.2 边界值分析
  - 原则
    - 如果程序中使用了一个内部数据结构，则应当选择这个内部数据结构的边界上的值做为测试用例。
    - 分析规格说明，找出其他可能的边界条件。

功能（黑盒）测试

- 5.3 基于判定表的测试
  - 概念
    - 判定表（ Decision Table ）亦称决策表，最适合描述在多个逻辑条件取值的组合所构成的复杂情况下，分别要执行哪些不同的动作。
    - 判定表由四部分组成
      - 条件茬（ Condition Stub ）
      - 动作茬（ Action Stub ）
      - 条件项（ Condition Entry ）
      - 动作项（ Action Entry ）

功能（黑盒）测试

- 5.3 基于判定表的测试

	规则 1	规则 2	规则 3	规则 4
C1 ：订货单金额 >5000 元？	T	T	F	F
C2 ：拖欠货款时间 >60 天？	T	F	T	F
a1 ： 货拒绝供货备忘录	Y	N	N	N
a2 ：发供货单	N	Y	Y	Y
a3 ：发催款通知单	N	N	Y	N

功能（黑盒）测试

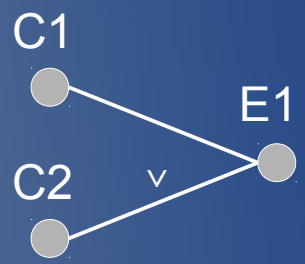
- 5.4 基于因果图的测试
  - 适用范围
    - 前面介绍的等价类测试方法和边界值分析方法，都是着重考虑输入条件，但对于输入条件之间的联系则讨论不多。因此必须考虑使用一种适合于描述对于多种条件的组合，相应产生多个动作的方法来考虑设计测试用例，这就需要利用因果图。

功能（黑盒）测试

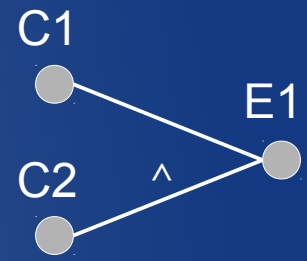
- 5.4 基于因果图的测试
  - 因果图符号

(a) 恒等

(b) 非



(c) 或



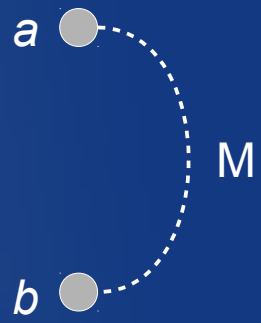
(c) 与

(a)E(互斥)

(a)1(包含)

(a)O(惟一)

(a)R(要求)



(a)M(屏蔽)

功能（黑盒）测试

- 5.5 基于状态图的测试
- 5.6 基于场景的测试

功能（黑盒）测试

- 5.7 其他黑盒测试用例设计技术
  - 规范（规格）导出法
  - 内部边界值测试法
  - 错误猜测法
  - 基于接口的测试
  - 基于故障的测试
  - 基于风险的测试
  - 比较测试

The End