软件工程

第18章 测试传统的应用系统

徐本柱 软件学院

2018-10

主要内容

- 软件测试基础
- 测试的内部视角和外部视角
- 白盒测试
- 基本路径测试
- 控制结构测试
- 黑盒测试

要点浏览

- 测试目标是设计一系列极可能发现错误的测试用例
- 软件测试技术为测试设计提供系统化的指导：
  - (1)执行每个软件构件的内部逻辑和接口;
  - (2)测试程序的输入和输出域以发现功能、行为和性能方面的错误。
- 人员：软件工程师→测试专家
- 步骤：两个不同角度
  - (1)利用“白盒”测试用例设计技术执行程序内部逻辑;
  - (2)利用“黑盒”测试用例设计技术确认软件需求
- 工作产品：一组测试用例、形成文档、记录结果
- 质量保证措施：改变视角，努力去“破坏”软件
  - 规范化设计测试用例并进行周密评审
  - 评估测试覆盖率并追踪错误检测活动

18.1 软件测试基础

- 测试目标是发现错误，好的测试发现错误的可能性较大
  - 设计与实现基于计算机的系统或产品时，应该考虑可测试性
  - 测试本身必须展示一系列特征，目标最小工作量发现最多错误
- 可测试性就是能够被测试的容易程度。特征如下：
  - 可操作性“运行得越好，越能有效地测试”
  - 可观察性“你所看见的就是你所测的”
  - 可控制性“对软件控制得越好，测试越能被自动执行或优化”
  - 可分解性“控制测试范围，更快孤立问题，完成更灵巧的再测试”
  - 简单性“需要测试的内容越少，测试的速度越快”
  - 稳定性“变更越少，对测试的破坏越小”
  - 易理解性“得到的信息越多，进行的测试越灵巧。”

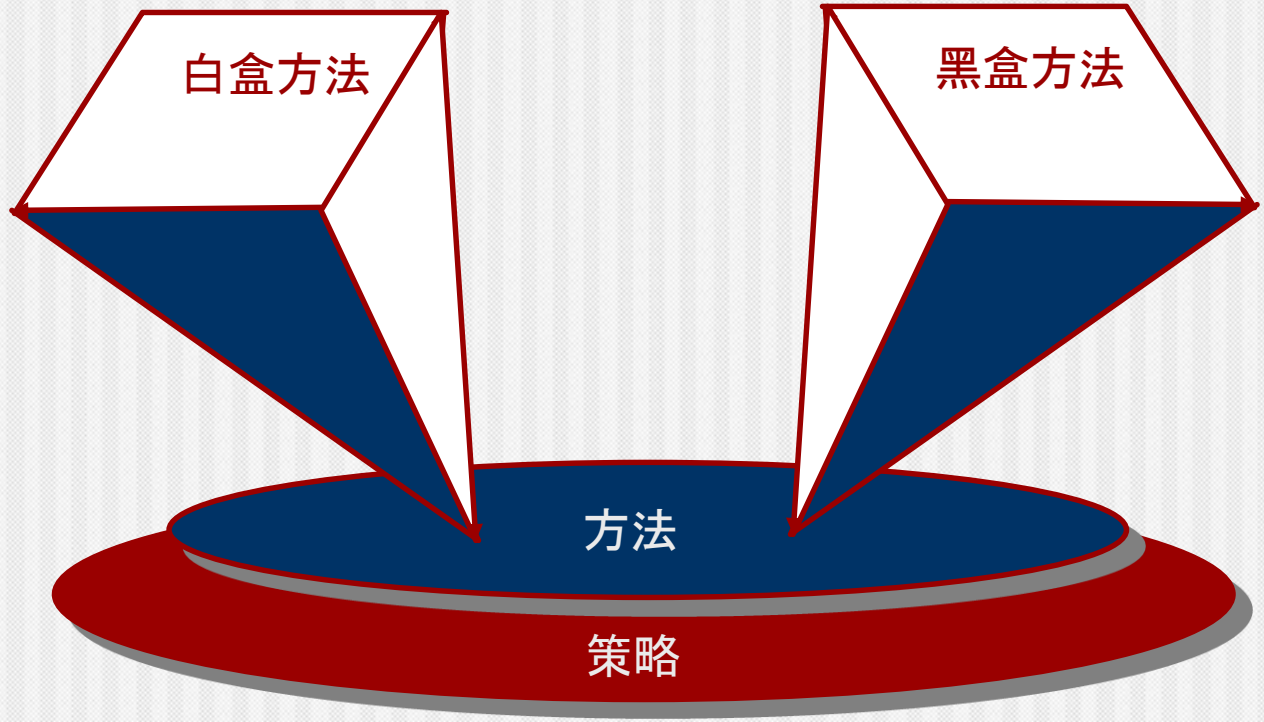
测试特征

- “好”的测试具有以下特性：
  - 具有较高的发现错误的可能性
  - 不冗余
  - 应该是“最佳品种”
  - 应该既不太简单也不太复杂

18.2 测试的内部视角和外部视角

- 工程化的产品都可以采用两种方式之一进行测试：
  - (1)了解已设计的产品所完成的指定功能，可以执行测试以显示每个功能是可操作的，同时查找在每个功能中的错误；
  - (2)了解产品的内部运行情况，可以执行测试以确保内部操作依据规格说明执行，而且对所有的内部构件已进行了充分测试。
- 第一种测试方法称为黑盒测试，第二种方法称为白盒测试。
  - 黑盒测试暗指在软件接口处执行测试。
  - 黑盒测试检查系统的功能方面，很少关心软件的内部结构。
  - 白盒测试是基于过程细节的封闭检查。
  - 通过提供检查特定条件集和(或)循环的测试用例，
  - 测试贯穿软件的逻辑路径和构件间的协作。

软件测试



18.3 白盒测试

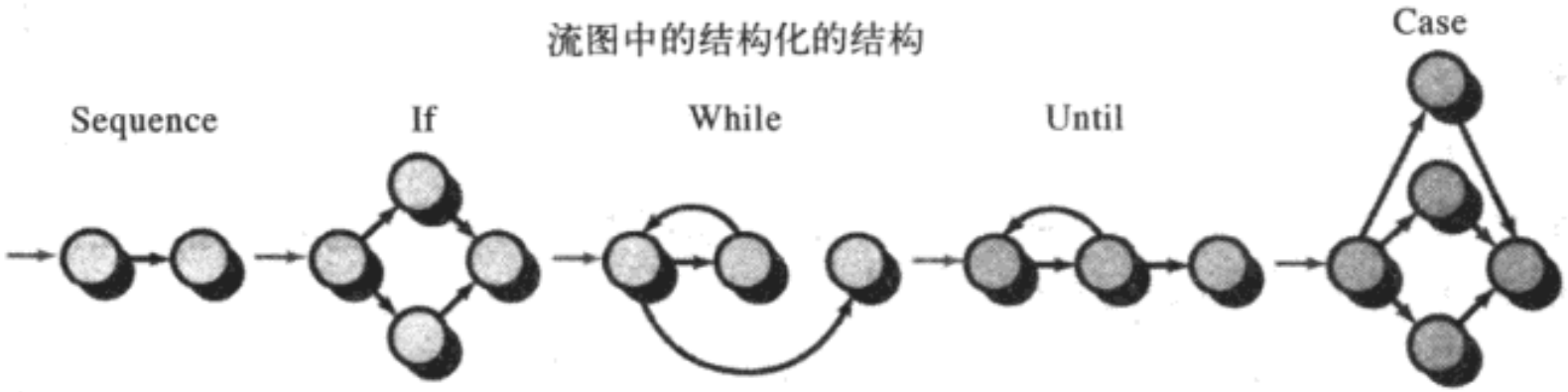
- 是一种测试用例设计方法，
  - 利用构件层设计描述的控制结构来生成测试用例。
  - 利用白盒测试方法导出的测试用例可以：
    - (1)保证一个模块中的所有独立路径至少被执行一次；
    - (2)对所有的逻辑值均需测试真和假；
    - (3)在上下边界及可操作的范围内执行所有的循环；
    - (4)检验内部数据结构以确保其有效性。

18.4 基本路径测试

- Tom McCabe首先提出的一种白盒测试技术
- 基本路径测试方法使测试用例设计者产生
  - 一种过程设计的逻辑复杂性测量，
  - 为执行路径的基本集的定义提供指导。
  - 该基本集所生成的测试用例保证程序中每一条语句至少执行一次

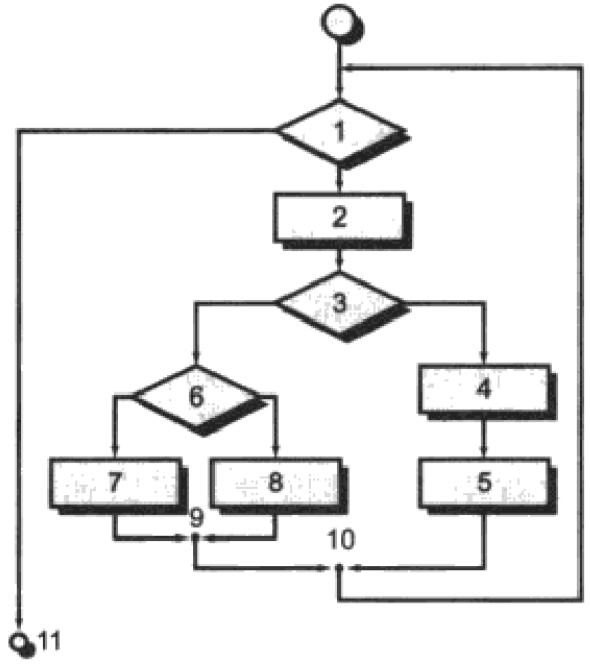
18.4.1 流图表示

- 流图利用下图所示的表示描述逻辑控制流
- 每种结构化构造都有相应的流图符号。
  - 圆称为流图结点，表示一个或多个过程语句。
  - 处理框序列和一个菱形判定框可以映射为单个结点
  - 流图中的箭头称为边或连接，表示控制流
  - 由边和结点限定的区间称为域。图形的外部作为一个域

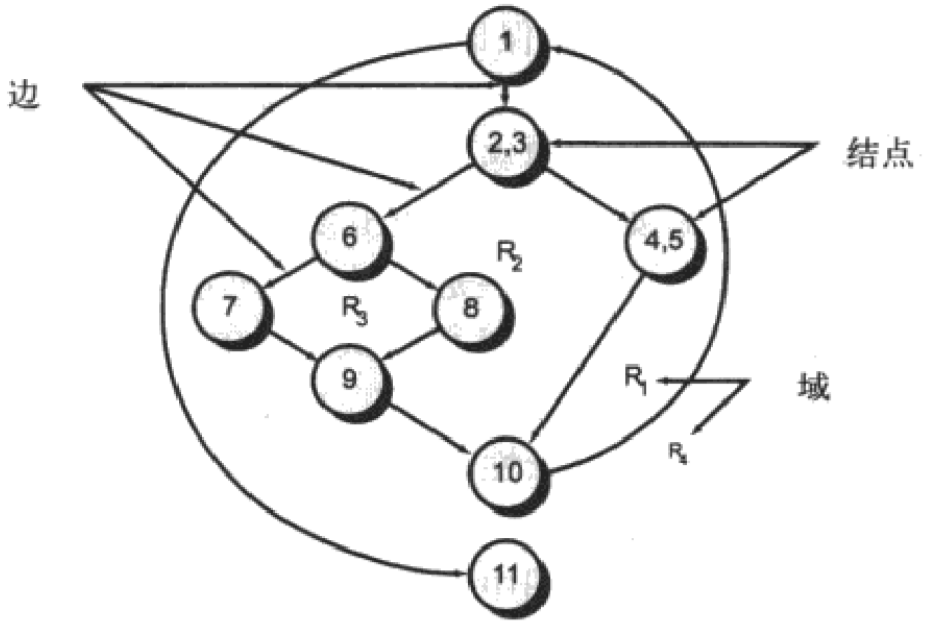


图中，每个圆圈表示一个或多个
无分支的PDL语句或源代码语句

流图转换



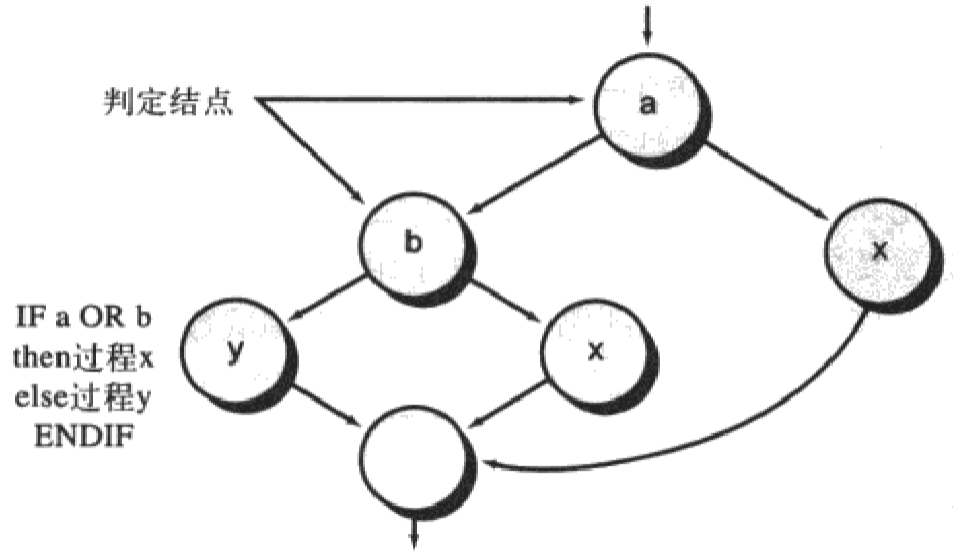
a) 流程图



b) 流图

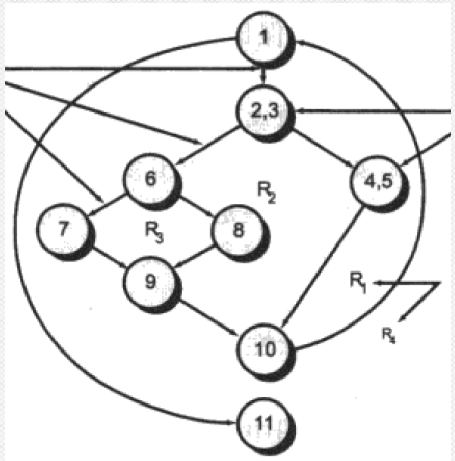
复合条件处理

- 为条件语句”IF a OR b”的每个条件(a或b)创建不同的结点
  - 包含条件的结点称为判定结点，
  - 用两条或多条由它发射出的边来描述。



18.4.2 独立程序路径

- 独立路径是
  - 任何贯穿程序的、
  - 至少引入一组新的处理语句或一个新的条件的路径。
- 按照流图进行描述，
  - 独立路径必须沿着至少一条边移动
  - 这条边在定义该路径之前未被遍历
  - 图18-2b所示的流图的一组独立路径如下：



路径1：1-11
路径2: 1-2-3-4-5-10-1-11
路径3: 1-2-3-6-8-9-10-1-11
路径4: 1-2-3-6-7-9-10-1-11

- 路径1、2、3和4构成所示流图的基本集合。
  - 设计测试执行这些路径，可保证每条语句至少执行一次
  - 且执行每个条件值为真和假。
  - 基本集合不是唯一的。

环复杂性计算

- 一共多少条独立路径？环复杂性的计算提供了答案。
- 环复杂性
  - 是一种软件度量，为程序的逻辑复杂度提供一个量化的测度。
  - 提供了保证所有语句被执行一次所需测试数量的上限。
- 计算方法
  - 1.域的数量与环复杂性相对应
  - 2.对流图G，环复杂性V(G)定义如下：

    V(G)=E-N+2

    其中E为流图的边数，N为流图的结点数。
  - 3.对流图G，环复杂性V(G)也可以定义如下：

    V(G)=P+1

    其中P为包含在流图G中的判定结点数

18.4.3 导出测试用例

- 基本路径测试方法可以应用于过程设计或源代码
- 基本测试用例集的生成
  - 1、以设计或源代码为基础，画出相应的流图
  - 2、确定所得流图的环复杂性。
  - 3、确定线性独立路径的基本集合。
  - 4、准备测试用例，强制执行基本集合中每条路径。
- 某些独立路径不能单独进行测试
  - 遍历路径所需的数据组合不能形成程序的正常流
  - 这些路径作为另一个路径的**一部分**进行测试。

18.5 控制结构测试

- 1 条件测试
  - 通过检查程序模块中包含的逻辑条件进行测试用例设计
  - 侧重于测试程序中的每个条件以确保其不包含错误。
- 2数据流测试
  - 根据变量的定义和使用位置来选择程序测试路径的测试方法
- 3循环测试
  - 侧重于循环构成元素的有效性
  - 简单循环、嵌套循环、串接循环和非结构化循环

18.6 黑盒测试

- 黑盒测试
  - 也称为行为测试，侧重于软件的功能需求。
  - 设计出将测试程序所有功能需求的输入条件集。
  - 不是白盒测试的替代品，是发现其他类型错误的辅助方法。
- 黑盒测试试图发现以下类型的错误：
  - 1）功能不正确或遗漏；
  - 2）接口错误；
  - 3）数据结构或外部数据库访问错误；
  - 4）行为或性能错误；
  - 5）初始化和终止错误。
- 与白盒测试不同
  - 白盒测试在测试过程的早期执行，而黑盒测试在测试的后期阶段
  - 黑盒测试故意不考虑控制结构，而是侧重于信息域。

18.6.1 等价划分

- 等价划分是一种黑盒测试方法，
  - 将程序的输入划分为若干个数据类，从中生成测试用例。
  - 理想的测试用例是可以单独发现一类错误。
- 测试用例设计是基于对输入条件的等价类进行评估。
  - 等价类表示输入条件的一组有效或无效的状态。
- 指导原则（输入条件）
  - 1指定一个范围→定义一个有效和两个无效的等价类；
  - 2需要特定的值→定义一个有效的和两个无效的等价类；
  - 3指定集合的某个元素→定义一个有效和一个无效的等价类；
  - 4为布尔值→定义一个有效和一个无效的等价类。

18.6.2 边界值分析

- 大量错误发生在输入域的边界处，而不是在其“中间”
  - 将边界值分析(BVA)作为一种测试技术的原因。
  - 边界值分析选择一组测试用例检查边界值。
- 是对“等价划分”测试用例设计技术的一种补充
  - 不是选择等价类元素，而是在其“边缘”上选择测试用例。
  - BVA不是仅仅侧重于的任何输入条件，也从输出域中生成测试。
- BVA的指导原则：（输入条件、输出条件）
  - 1、以a和b为边界→测试用例应该包括a、b、略大于a和略小于b;
  - 2、为一组值→最大值和最小值，略大于最小值和略小于最大值;
  - 3、指导原则1和2也适用于输出条件;
  - 4、内部数据结构有预定义的边界值→在其边界处测试数据结构。

18.7 基于模型的测试

- 分析软件已有的行为模型（或创建一个行为模型）
  - 回顾一下，一个行为模型表明软件将如何应对外部事件或刺激。
- 遍历行为模型，并指定输入，使得软件从状态到状态的转换。
  - 输入将触发事件，使得转换发生。
- 回顾行为模型，并注意到预期的输出，因为软件使状态到状态的转换发生。
- 运行测试用例。
- 比较实际和预期的结果，并根据需要采取纠正措施。

谢谢！