软件测试基础

清华大学软件学院 刘强

教学提纲

1. 软件测试概述
2. 软件测试类型
3. 白盒测试方法
4. 黑盒测试方法

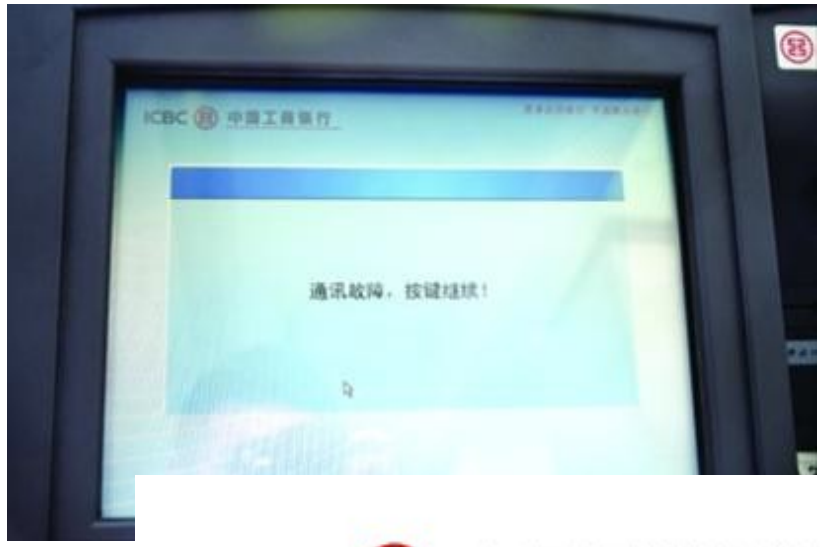
教学提纲

1 □□□□□□

- 软件缺陷术语
- 软件测试概念
- 软件测试基本原则
- 软件测试团队

关于软件质量

The average software product released on the market is not error free.



ICBC 中国工商银行

术语解释

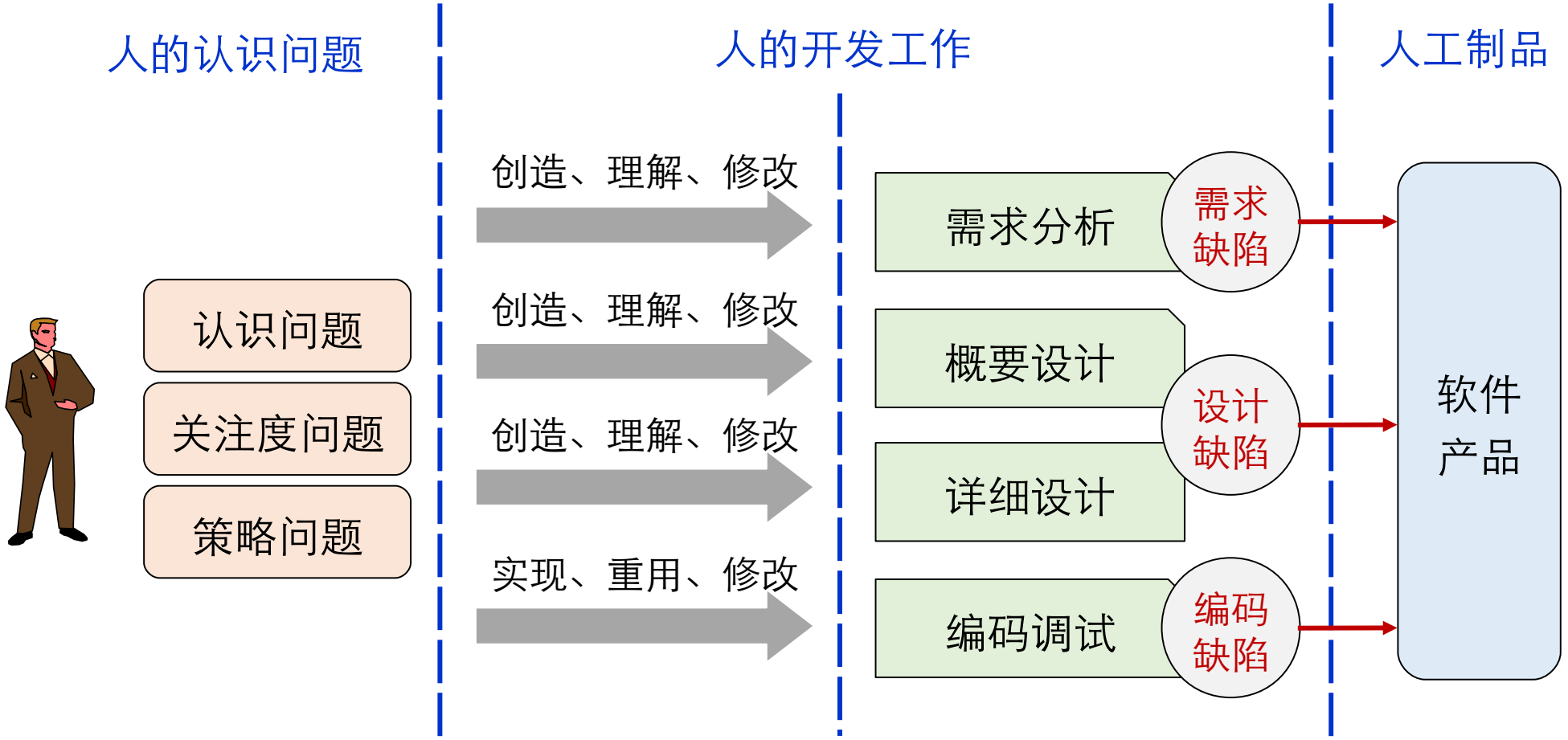
□□□ Error □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□ Defect □□
□□□□□□□□□□□□

□□□ Fault □□□
□□□□□□□□□□□□

□□□ Failure □□□□□□□□□□□□□□□□□□□□□□□□□□□□

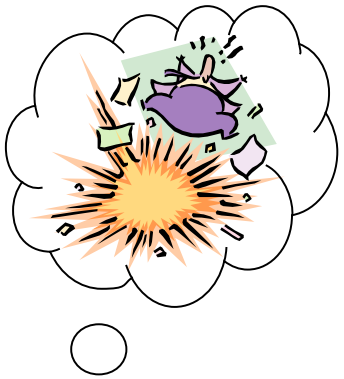
软件缺陷的产生


人的认识问题
人的开发工作
人工制品
认识问题
关注度问题
策略问题
创造、理解、修改
创造、理解、修改
创造、理解、修改
实现、重用、修改
需求分析
概要设计
详细设计
编码调试
需求缺陷
设计缺陷
编码缺陷
软件产品

软件缺陷的演化

□□□□□□□□

1991 □ 2 □□□□□□□□□□
28 □□□□□□□



□□
□□

□□□□□□□□ 1/10 □□□□
□□□□

□□□□□□

□□ □□ □□

□□□□□□□□□□
□ 0.000000095 □ 10

0.000000095×100h×60×60×10 = 0.34s

软件测试

测试 —— 发现问题 —— 修复

什么是软件测试?

软件测试的目的是什么?

软件测试的定义

正向思维：验证软件正常工作

- 评价一个程序或系统的特性或能力并确定是否达到预期的结果。
- 在设计规定的环境下运行软件的所有功能，直至全部通过。

逆向思维：假定软件有缺陷

- 测试是为了发现错误而针对某个程序或系统的执行过程。
- 寻找容易犯错地方和系统薄弱环节，试图破坏系统直至找不出问题。

软件测试的目的

测试的局限性

测试的不彻底性

- 测试只能说明错误的存在，但不能说明错误不存在
- 经过测试后的软件不能保证没有缺陷和错误

测试的不完备性

- 测试无法覆盖到每个应该测试的内容
- 不可能测试到软件的全部输入与响应
- 不可能测试到全部的程序分支的执行路径

测试作用的间接性

- 测试不能直接提高软件质量，软件质量的提高要依靠开发
- 测试通过早期发现缺陷并督促修正缺陷来间接地提高软件质量

软件测试心理学

人类行为总是倾向于具有高度目标性，
确立一个正确的目标具有重要的心理学影响。

- ✓ 如果测试的目的是要证明程序中没有错误，就会在潜意识中倾向于选择使程序出错可能性较小的测试数据。
- ✓ 如果目的是要证明程序中存在错误，就会选择一些易于发现程序所含错误的测试数据。

- ✓ 当人们开始一项工作时，如果已经知道它是不可行或无法实现的，表现就会很糟糕。
- ✓ 要证明软件中不存在错误是不可能的，如果把测试看成是发现错误的过程，那么测试就是可完成的任务。

一定程度的独立测试通常可以更高效地发现软件缺陷，但独立不能替代对软件的熟悉和经验。

软件测试原则

□□ 1 □ [illegible]

□□ 2 □ [illegible]

□□ 3 □ [illegible]

□□ 4 □ [illegible]

□□ 5 □ [illegible]

□□ 6 □ [illegible]“[illegible]”[illegible]“[illegible]” □

□□ 7 □ [illegible]

□□ 8 □ [illegible]

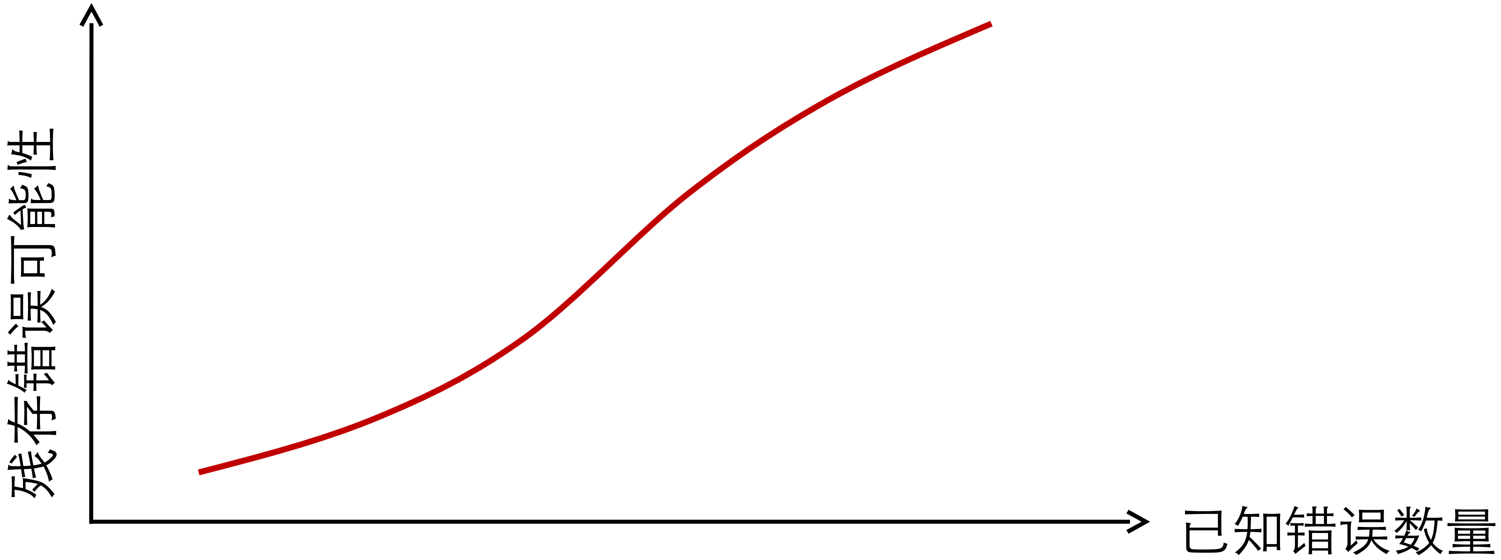
□□ 9 □ [illegible]

□□ 10 □ [illegible]

缺陷的集群性

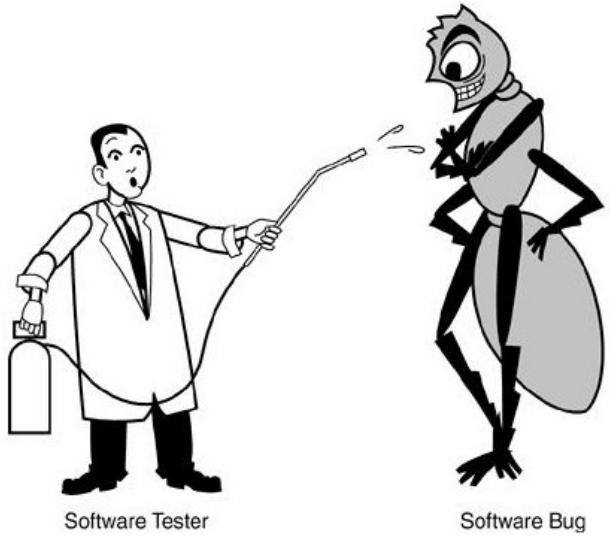
软件错误具有聚集性，对存在错误的部分应重点测试。

- 80/20 □□□ 80% □□□□□□□□ 20% □□□□□
- □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□



杀虫剂悖论

用同样的测试用例多次重复进行测试，最后将不再能够发现新的缺陷。



如同给果树喷撒农药，为了杀灭害虫只打一种杀虫药，虫子就会有抗体而变得适应，于是杀虫剂将不再发挥作用。

测试用例需要定期评审和修改，同时要不断增加新的不同测试用例来测试软件的不同部分，从而发现更多潜在的缺陷。

软件测试团队的任务

软件测试与质量保证合二为一

① 发现软件程序、系统或产品中所有的问题

② 尽早地发现问题

③ 督促开发人员尽快地解决程序中的缺陷

④ 帮助项目管理人员制定合理的开发计划

⑤ 对问题进行分析、分类总结和跟踪

⑥ 帮助改善开发流程，提高产品开发效率

⑦ 提高程序编写的规范性、易读性、可维护性等

测试人员角色

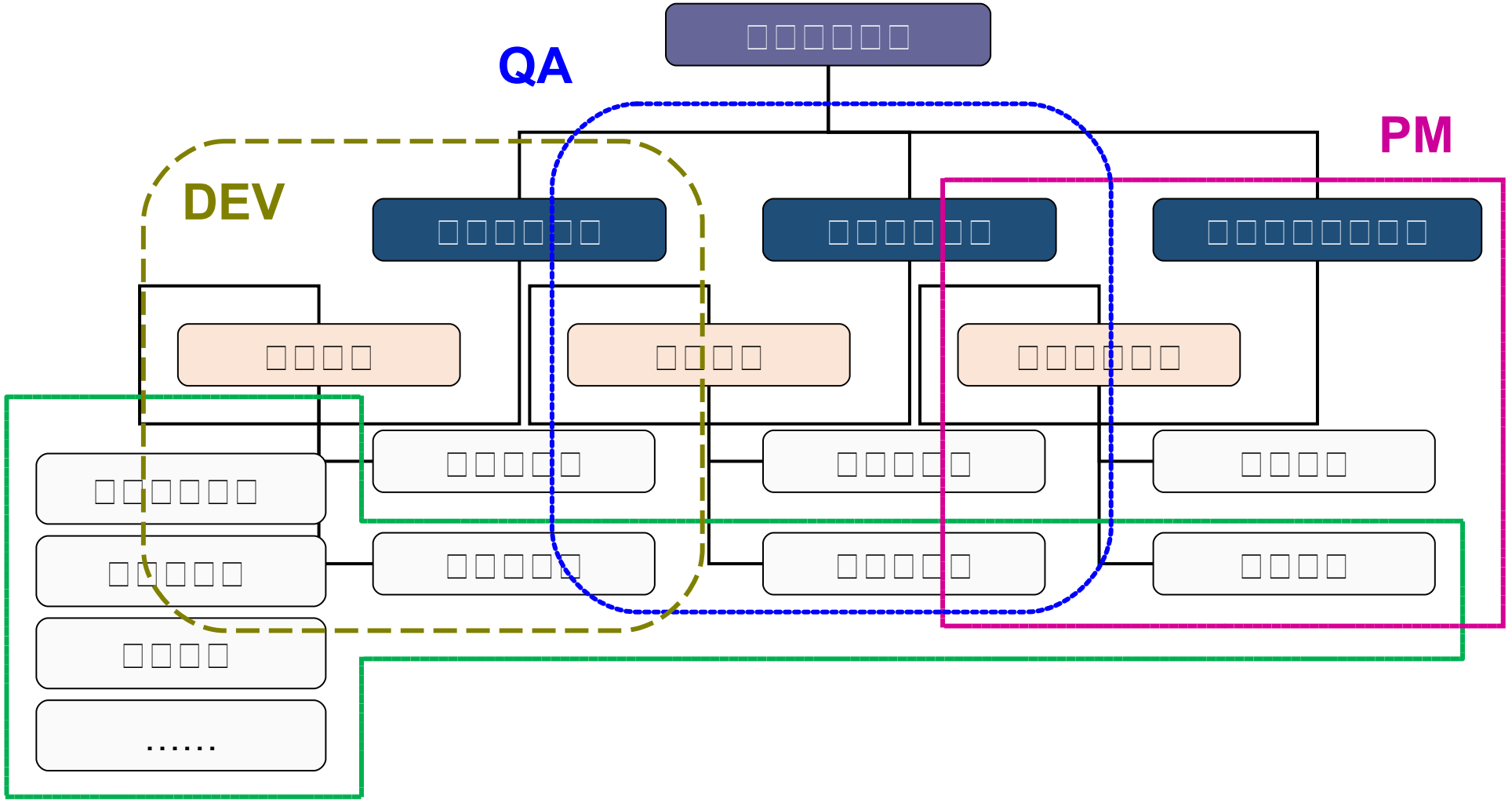
测试经理：建立和完善测试流程以及部门管理体制，审核测试项目并分配资源，监控和协调各项目的测试工作，负责与其他部门的协调和沟通工作。

测试组长：制定测试项目计划（包括人员、进度、软硬件环境和流程等），实施软件测试，跟踪和报告计划执行情况，负责测试用例质量，管理测试小组并提供技术指导。

测试工程师：理解软件产品的要求，对其进行测试以便发现软件中的错误，验证软件是否满足规格说明所描述的需求，编写相应的测试方案和测试用例。

测试工具工程师：编写软件测试工具，并利用所编写的测试工具对软件进行测试，或者为测试工程师开发测试工具。

举例：微软研发团队



微软测试工程师的一天

[illegible]	[illegible] BVT [illegible] Basic Verification Test [illegible]
1.	[illegible] BVT [illegible] Email [illegible] BVT [illegible]
2.	[illegible] Bug [illegible] Bug [illegible]
3.	[illegible] Bug [illegible] Bug [illegible]
4.	[illegible] Bug [illegible]
5.	[illegible]
6.	[illegible]
7.	[illegible]
8.	[illegible]
9.	[illegible]
10.	[illegible]

软件测试人员

核心素质：责任心

测试技术能力

- 掌握理论、方法、工具
- 开发测试工具、自动化测试框架、测试脚本等
- 掌握操作系统、网络、数据库、中间件等知识

非技术能力

- 沟通能力，协作精神
- 自信心、耐心、专心
- 洞察力、记忆力
- 怀疑精神，创造性
- 反向思维与发散思维
- 自我督促、幽默感

专业领域经验

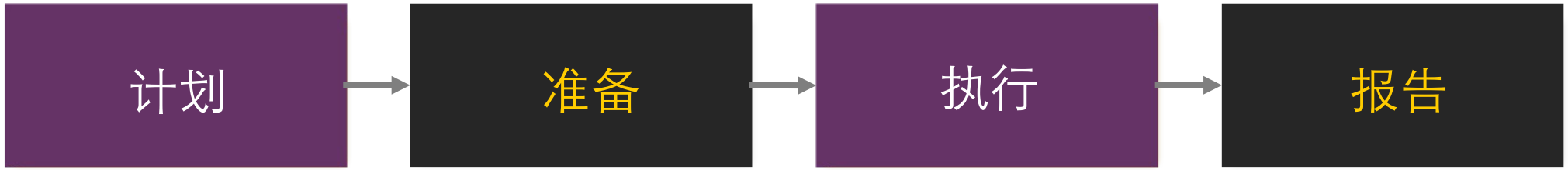
- 了解业务知识
- 积累实践经验

教学提纲

2 □□□□□□

- 软件测试活动
- 软件测试类型
- 软件测试模型
- 测试用例设计

软件测试过程



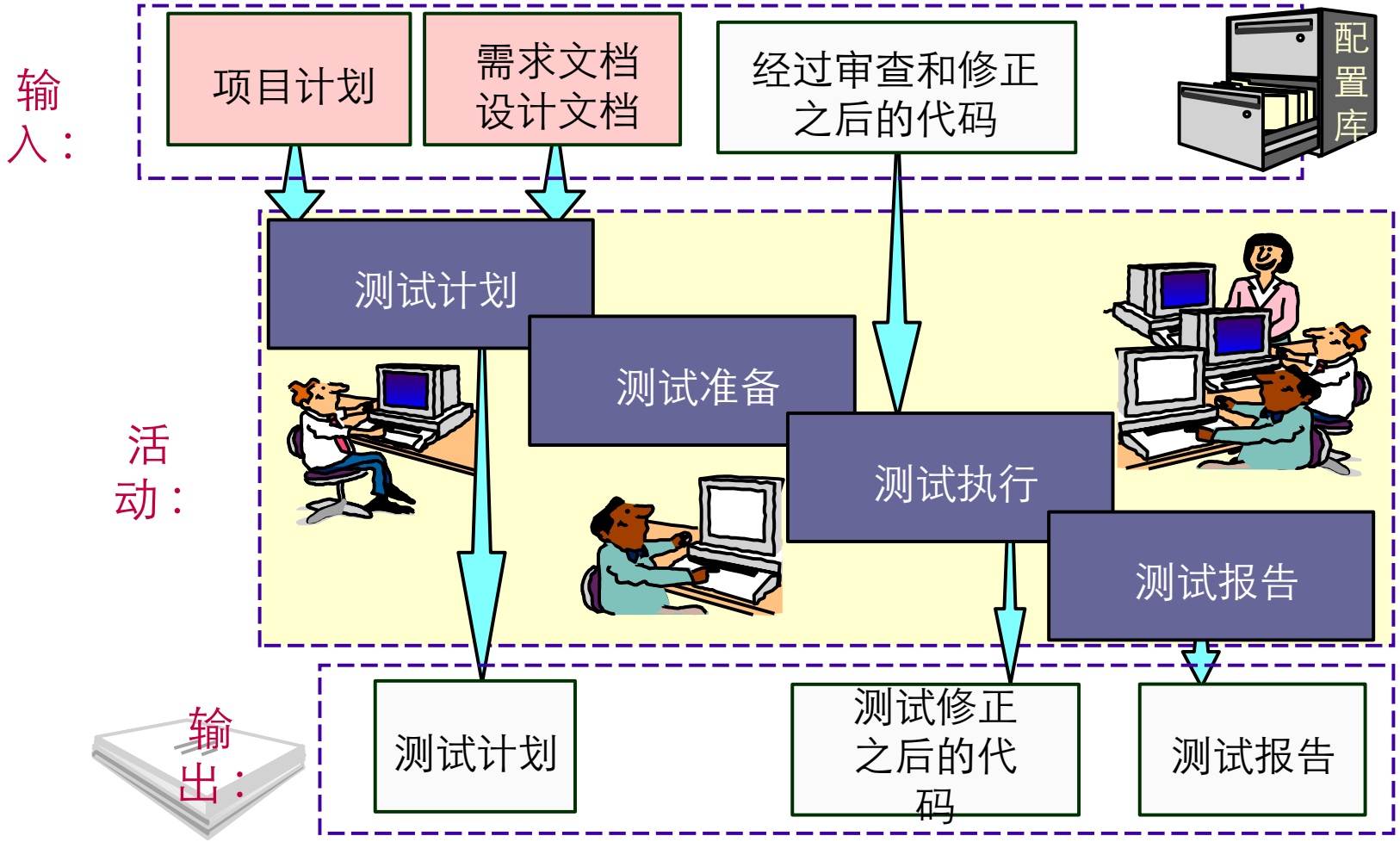
- 识别测试需求
- 分析质量风险
- 拟定测试方案
- 制定测试计划

- 组织测试团队
- 设计测试用例
- 开发工具和脚本
- 准备测试数据

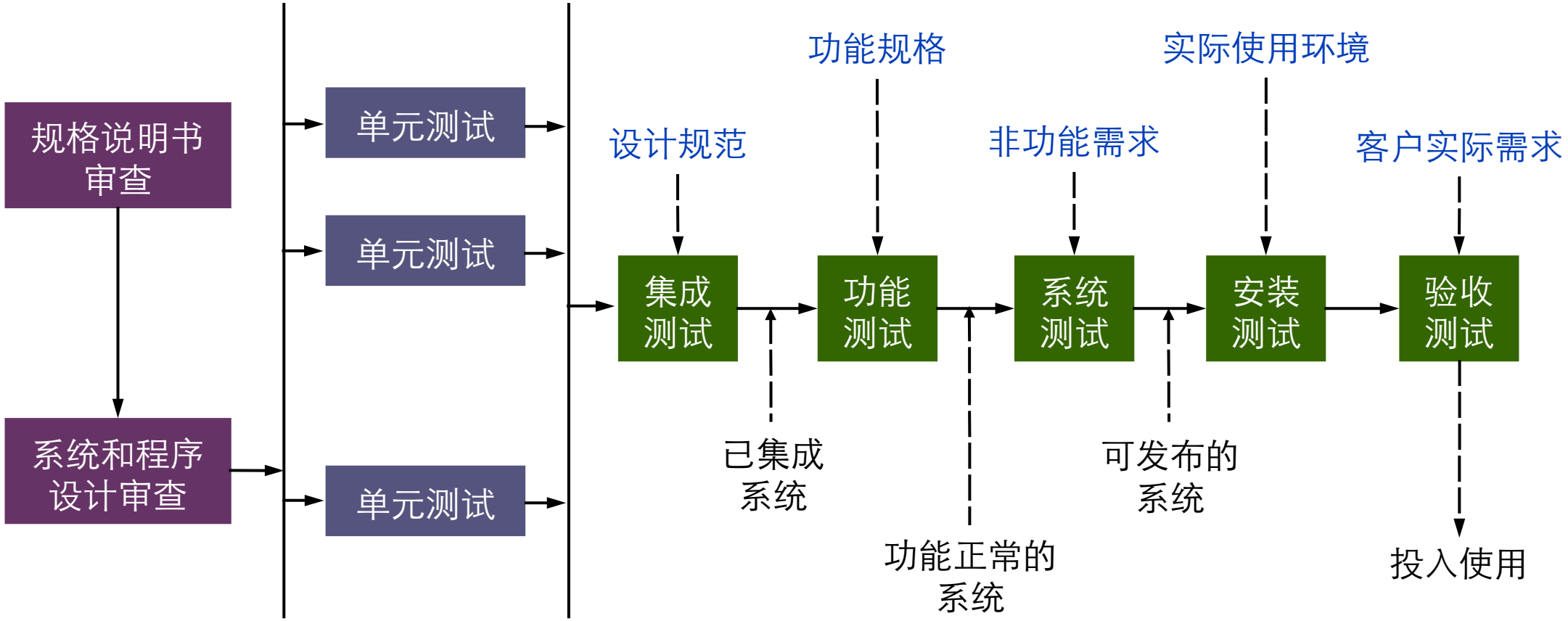
- 获得测试版本
- 执行和实施测试
- 记录测试结果
- 跟踪和管理缺陷

- 分析测试结果
- 评价测试工作
- 提交测试报告

软件测试活动


输入：
项目计划
需求文档
设计文档
经过审查和修正之后的代码
配置库
活动：
测试计划
测试准备
测试执行
测试报告
输出：
测试计划
测试修正之后的代码
测试报告

软件测试阶段


规格说明书
审查
系统和程序
设计审查
单元测试
单元测试
单元测试
设计规范
功能规格
非功能需求
实际使用环境
客户实际需求
集成
测试
功能
测试
系统
测试
安装
测试
验收
测试
已集成
系统
功能正常的
系统
可发布的
系统
投入使用

软件测试类型

测试对象角度	单元测试、集成测试、系统测试、验收测试
测试技术角度	黑盒测试（功能测试）、白盒测试（结构测试）
程序执行角度	静态测试、动态测试
人工干预角度	手工测试、自动化测试

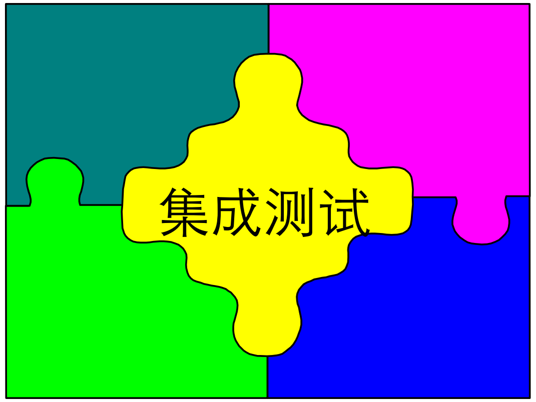
单元测试

□□□□□Unit　Testing□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

集成测试

□□□□□ Integration Testing □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□



- 一次性集成方式：分别测试每个单元，再一次性将所有单元组装在一起进行测试。
- 渐增式集成方式：先对某几个单元进行测试，然后将这些单元逐步组装成较大的系统，在组装过程中边连接边测试。

集成测试的对象是模块间的接口，其目的是找出在模块接口上，包括系统体系结构上的问题。

功能测试

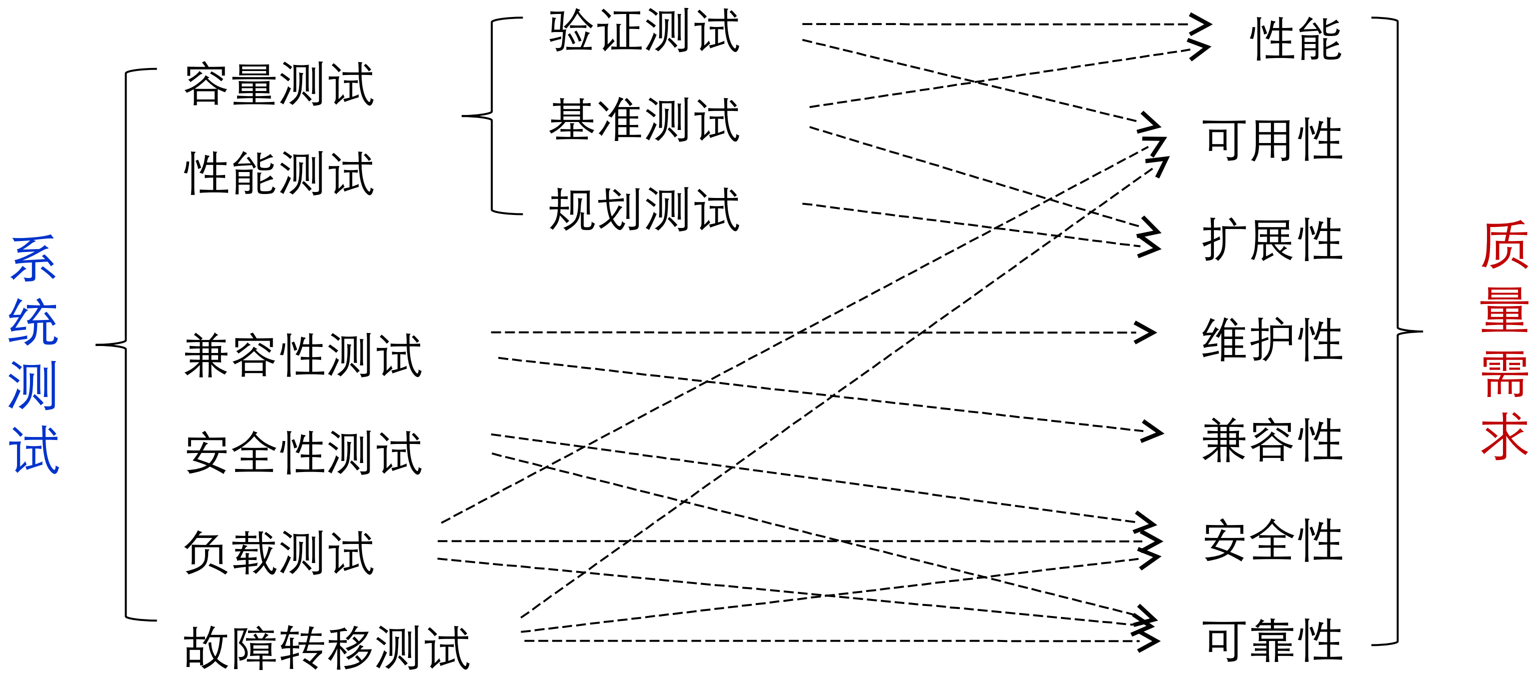
□□□□□ Functional Testing □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□

- 界面
- 数据
- 操作
- 逻辑
- 接口

系统测试

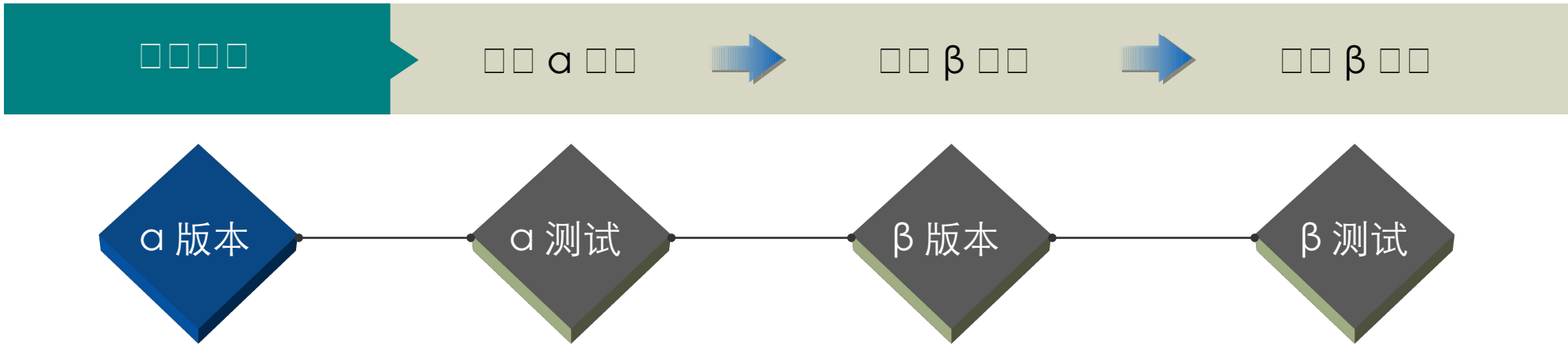
□□□□□ System Testing □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□



验收测试

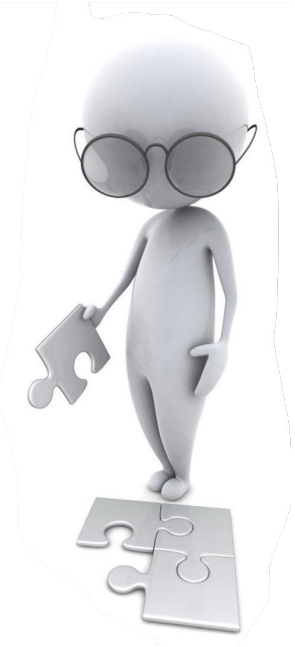
验收测试是在软件产品完成了功能测试和系统测试之后、产品发布之前所进行的软件测试活动，其目的是验证软件的功能和性能是否能够满足用户所期望的要求。



安装测试

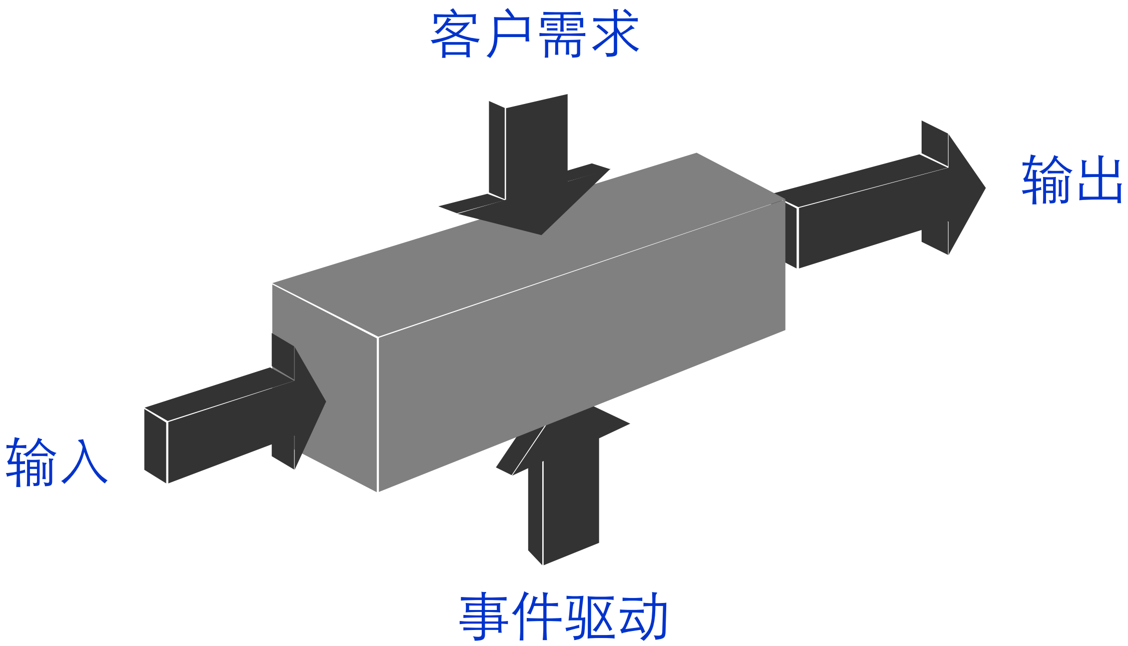
安装测试是系统验收之后，需要在目标环境中进行安装，其目的是保证应用程序能够被成功地安装。

- 应用程序是否可以成功地安装在以前从未安装过的环境中?
- 应用程序是否可以成功地安装在以前已有的环境中?
- 配置信息定义正确吗?
- 考虑到以前的配置信息吗?
- 在线文档安装正确吗?
- 安装应用程序是否会影响其他的应用程序吗?
- 安装程序是否可以检测到资源的情况并做出适当的反应?



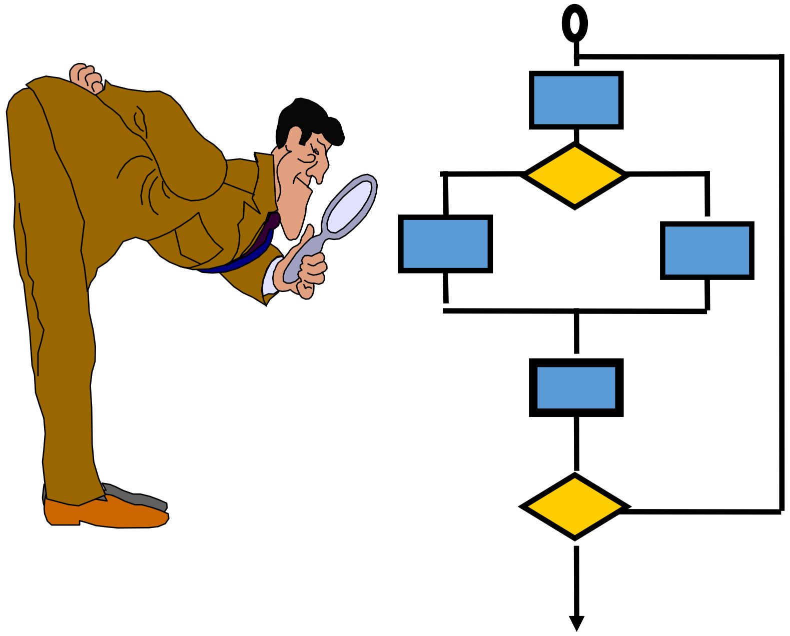
黑盒测试

□□□□□ Black Box Testing □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□



白盒测试

□□□□□ White Box Testing □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□



静态测试与动态测试

静态测试：通过人工分析或程序正确性证明的方式来确认程序正确性。

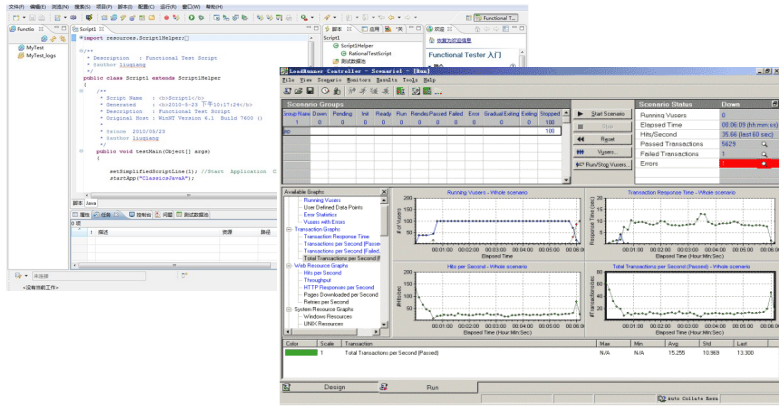
非正式　　正式

互审　　走查　　审查会议

动态测试：通过动态分析和程序测试等方法来检查程序执行状态，以确认程序是否有问题

手工测试与自动化测试

- 手工测试：测试人员根据测试大纲中所描述的测试步骤和方法，手工地输入测试数据并记录测试结果。
- 自动化测试：相对于手工测试而言，主要是通过所开发的软件测试工具、脚本等手段，按照测试工程师的预定计划对软件产品进行的自动测试。

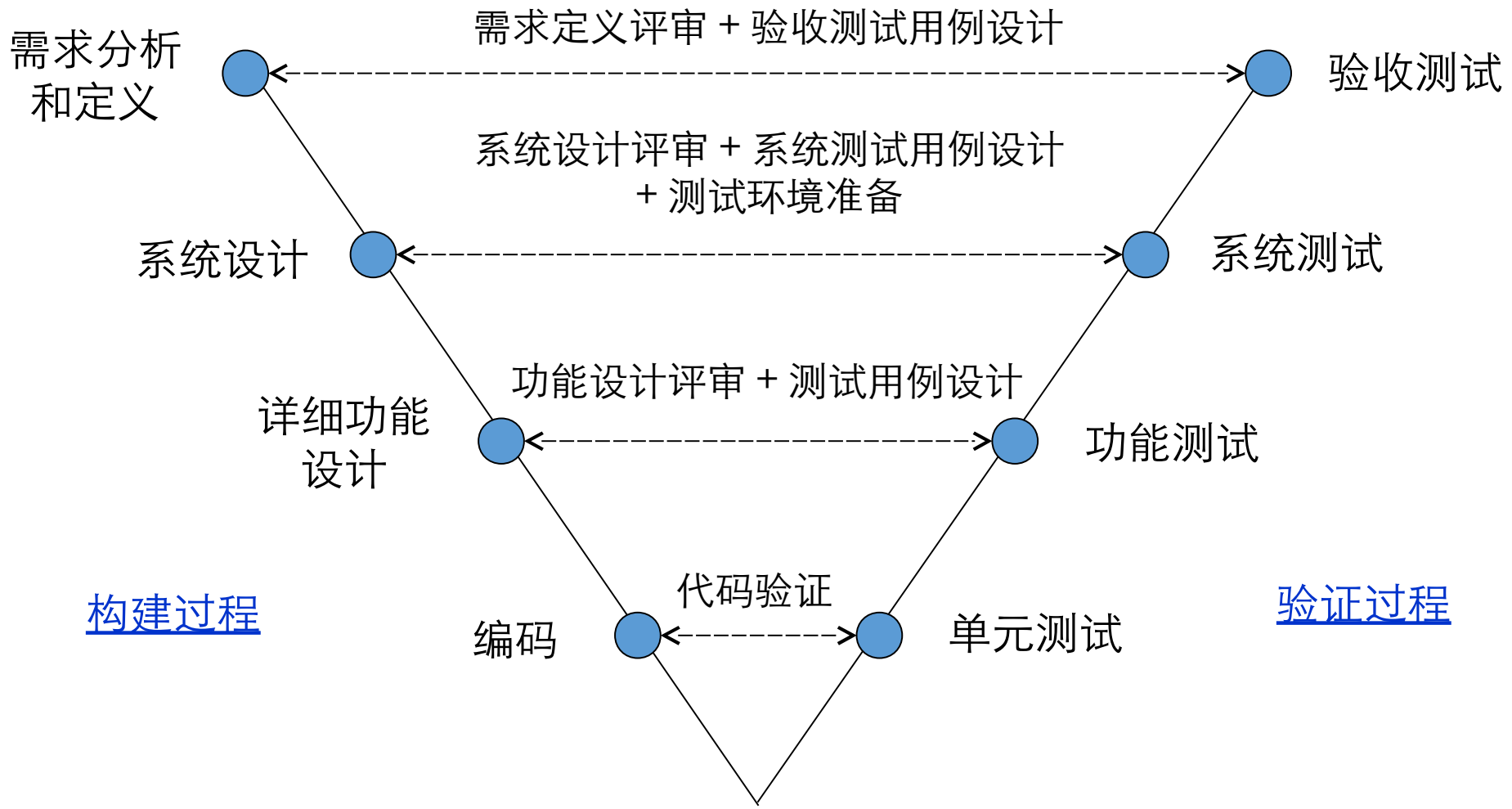


手工测试与自动化测试

自动化测试只是对手工测试的一种补充，但绝不能代替手工测试，二者有各自的特点。

- □□□
- □□
- □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□□□□□□□□□□□15~30%□□□□□□□□□□□□□70~85%□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

V □□


需求分析
和定义
需求定义评审 + 验收测试用例设计
验收测试
系统设计评审 + 系统测试用例设计
+ 测试环境准备
系统设计
系统测试
功能设计评审 + 测试用例设计
详细功能
设计
功能测试
代码验证
编码
单元测试
构建过程
验证过程

测试用例的重要性

降低软件测试成本

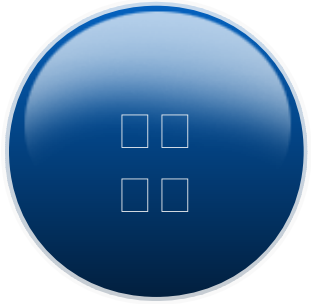
保证测试工作质量

评估和检验测试效果

测试用例的重要性

测试用例的重要性

测试用例



测试集
测试用例值的集合

测试用例值 → 完成被测软件的某个执行所需的输入值

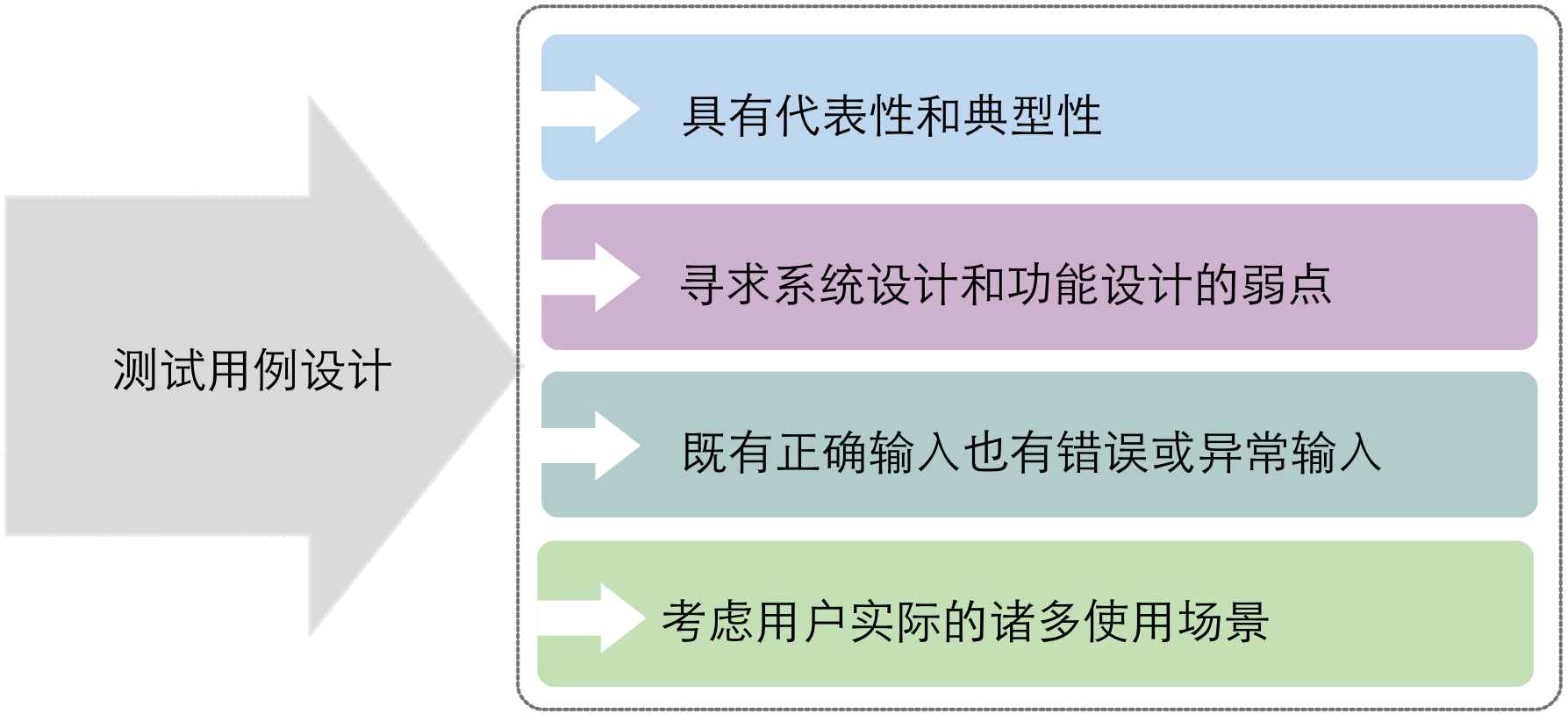
期望结果 → 当且仅当程序满足其期望行为，执行测试时产生的结果

前缀值 → 将软件置于合适的状态来接受测试用例值的任何必要的输入

后缀值 → 测试用例值被发送以后，需要被发送到软件的任何输入

验证值：查看测试用例值结果所要用到的值
结束命令：终止程序或返回到稳定状态所要用到的值

测试用例设计要求


测试用例设计
具有代表性和典型性
寻求系统设计和功能设计的弱点
既有正确输入也有错误或异常输入
考虑用户实际的诸多使用场景

案例讨论：纸杯测试

人们在日常生活中经常使用一次性纸杯，请根据自己的生活常识，提出尽可能多的测试用例，并进一步给出设计建议。

教学提纲

3 □□□□□□

- □□□□□□
- □□□□□□□□□
- □□□□□□□□□□□
- □□□□

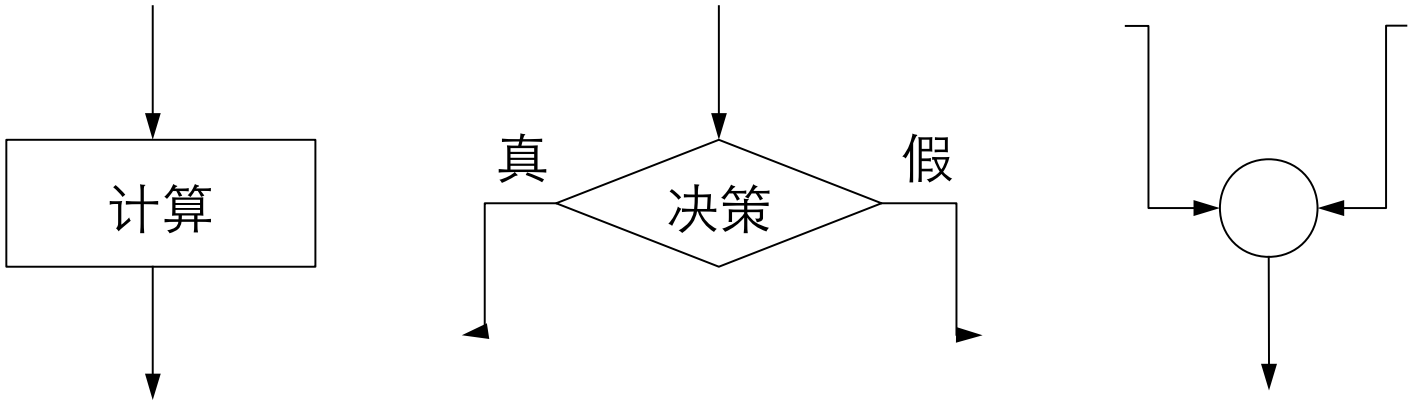
白盒测试方法

控制流图

□□□□□ CFG □ Control Flow Graph □□□□□□□□□□□□□□□□

□□□□□□□□□□

- □□□□□□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□□□□□□□□□□□□□



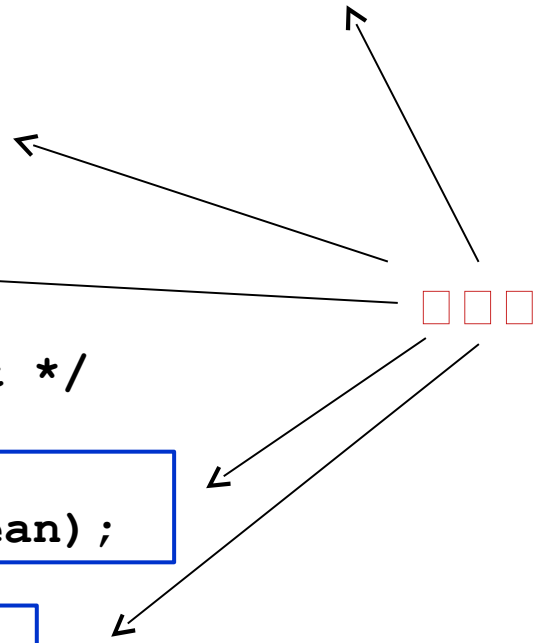
□□□□ 判断节点 合并节点

示例：控制流图

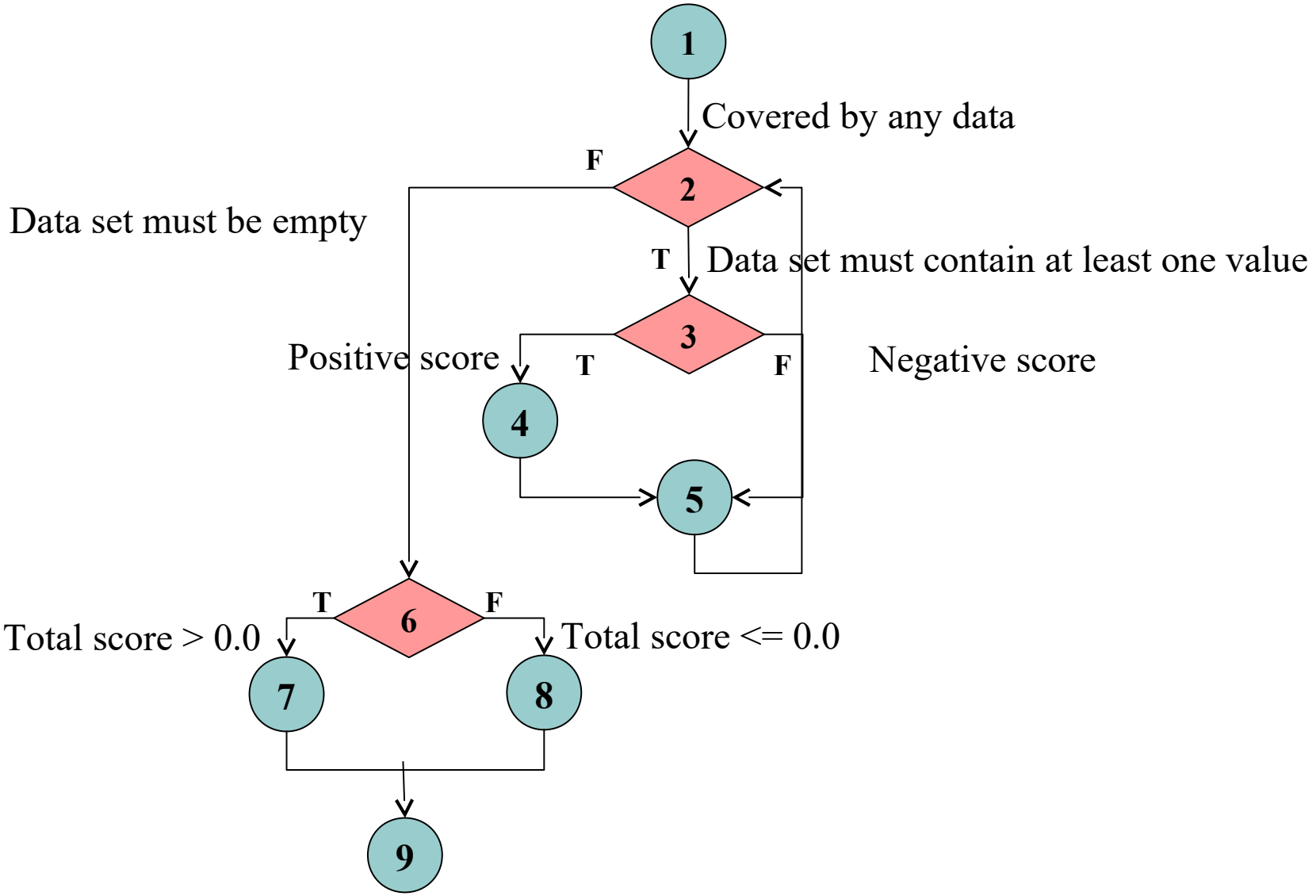
```
FindMean(float *mean, FILE *fp)
{
    float sum = 0.0, score = 0.0;
    int num = 0;

    fscanf(fp, "%f", &score); /* Read and parse into score */
    while (!EOF(fp)) {
        if (score > 0.0) {
            sum += score;
            num++;
        }
        fscanf(fp, "%f", &score);
    }
    /* Compute the mean and print the result */
    if (num > 0) {
        *mean = sum/num;
        printf("The mean score is %f \n", mean);
    } else
        printf("No scores found in file\n");
}
```

示例：控制流图

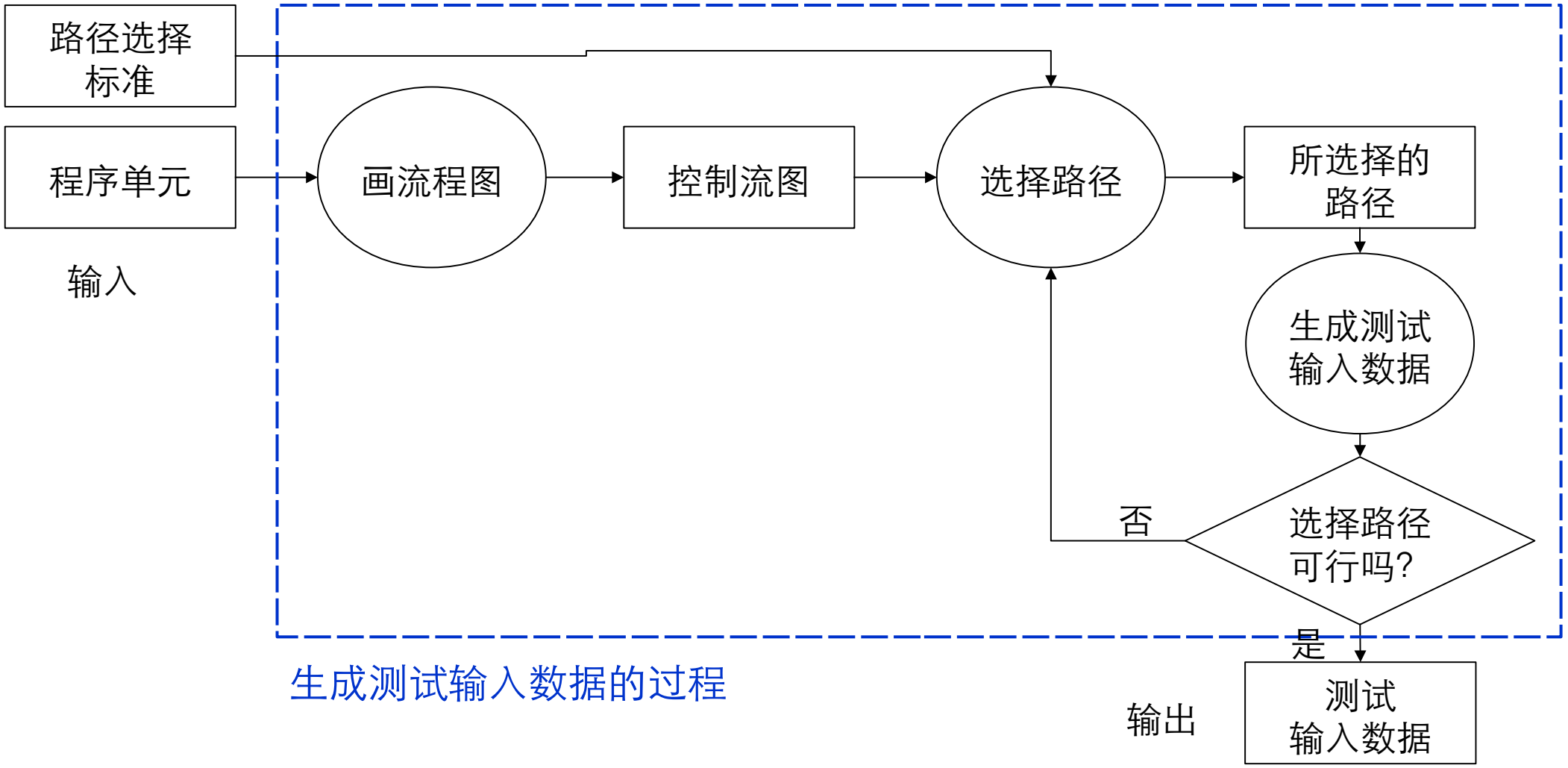
```
FindMean(float *mean, FILE *fp)
{
    float sum = 0.0, score = 0.0;
①   int num = 0;

    fscanf(fp, "%f", &score); /* Read and parse into score */
②   while (!EOF(fp)) {
      ③ if (score > 0.0) {
            ④ sum += score;
              num++;
          }
      ⑤ fscanf(fp, "%f", &score);
    }
    /* Compute the mean and print the result */
⑥  if (num > 0) {
      ⑦ *mean = sum/num;
        printf("The mean score is %f \n", mean);
    } else
      ⑧ printf("No scores found in file\n");
}⑨
```



示例：控制流图

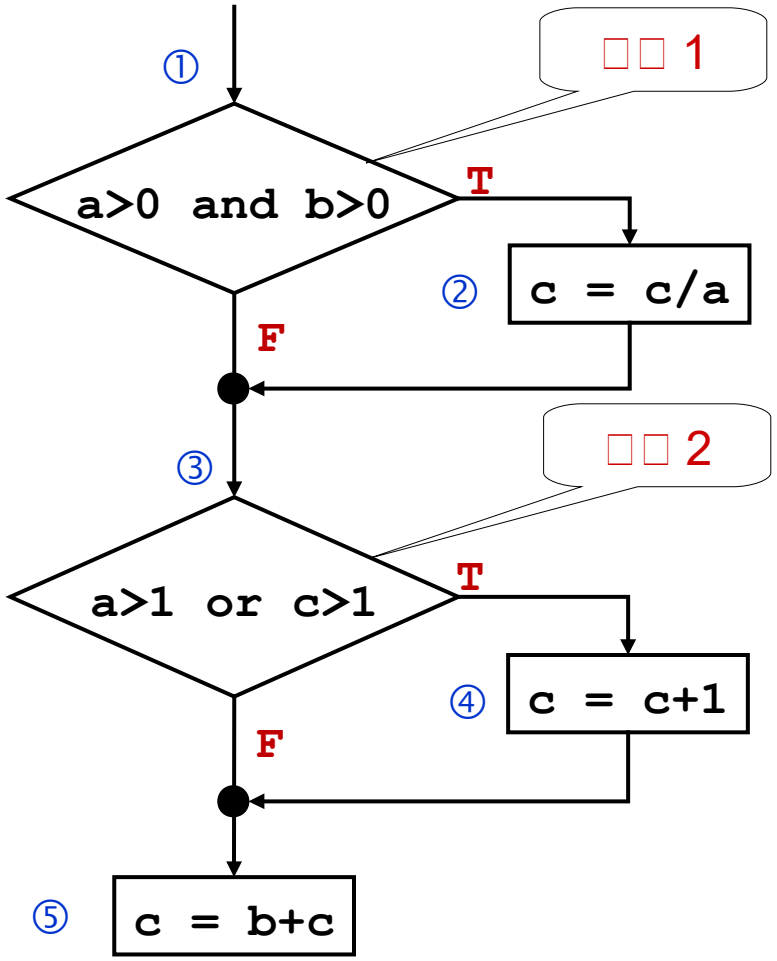

1
Covered by any data
F
2
Data set must be empty
T
Data set must contain at least one value
3
Positive score
T
F
Negative score
4
5
T
6
F
Total score > 0.0
Total score <= 0.0
7
8
9

基于控制流的测试


路径选择
标准
程序单元
输入
画流程图
控制流图
选择路径
所选择的
路径
生成测试
输入数据
选择路径
可行吗?
否
是
测试
输入数据
输出
生成测试输入数据的过程

示例描述

```
double func1(int a, int b, double c)
{
    if (a>0 && b>0) {
        c = c/a;
    }

    if (a>1 || c>1) {
        c = c+1;
    }

    c = b+c;

    return c;
}
```



语句覆盖

基本思想

程序中的每个可执行语句至少被执行一次。

□□□□□

□□□ **a=2 □b=1 □c=6**

□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□

□□“ **&&**” □□□□□□“ **||**” □

□□□□□□□□□□□□□□□

判定覆盖（分支覆盖）

基本思想

程序中每个判断的取真分支和取假分支至少经历一次，即判断真假值均被满足。

□□□□□

□□□ **a=2,b=1,c=6** □□□□□ 1 □ T □□□□□ 2 □ T □□

□□□ **a=-2,b=-1,c=-3** □□□□□ 1 □ F □□□□□ 2 □ F □□

□□□□□

□□□
□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

条件覆盖

基本思想

每个判断中每个条件的可能取值至少满足一次。

□□□□□

□□□ a=2,b=-1,c=-2 □□□ a>0,b<=0,a>1,c<=1 □□

□□□ a=-1,b=2,c=3 □□□ a<=0,b>0,a<=1,c>1 □□

□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ 1 □ T □□□□□ 2 □ F □□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

判定条件覆盖

基本思想

判断条件中的所有条件可能取值至少执行一次，同时所有判断的可能结果至少执行一次。

□□□□□

□□□ a=2,b=1,c=6 □□□ a>0,b>0,a>1,c>1 □□□□□ T

□□□ a=-1,b=-2,c=-3 □□□ a<=0,b<=0,a<=1,c<=1 □□□□□ F

□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□

条件组合覆盖

基本思想

判断中每个条件的所有可能取值组合至少执行一次，并且每个判断本身的结果也至少执行一次。

□□□□	□□□□□□	□□□□□□	□□ - □□□□
1	a>0 □ b>0	□□ 1 □ T	a>0 □ b>0 □□□ 1 □ T
2	a>0 □ b<=0	□□ 1 □ F	a>0 □ b<=0 □□□ 1 □ F
3	a<=0 □ b>0	□□ 1 □ F	a<=0 □ b>0 □□□ 1 □ F
4	a<=0 □ b<=0	□□ 1 □ F	a<=0 □ b<=0 □□□ 1 □ F
5	a>1 □ c>1	□□ 2 □ T	a>1 □ c>1 □□□ 2 □ T
6	a>1 □ c<=1	□□ 2 □ T	a>1 □ c<=1 □□□ 2 □ T
7	a<=1 □ c>1	□□ 2 □ T	a<=1 □ c>1 □□□ 2 □ T
8	a<=1 □ c<=1	□□ 2 □ F	a<=1 □ c<=1 □□□ 2 □ F

条件组合覆盖

□□□□	□□□□	□□□□	□□□□
□□□ a=2 □ b=1 □ c=6	a>0,b>0 a>1,c>1	1-2-4	1 □ 5
□□□ a=2 □ b=-1 □ c=-2	a>0,b<=0 a>1,c<=1	1-3-4	2 □ 6
□□□ a=-1 □ b=2 □ c=3	a<=0,b>0 a<=1,c>1	1-3-4	3 □ 7
□□□ a=-1 □ b=-2 □ c=-3	a<=0,b<=0 a<=1,c<=1	1-3-5	4 □ 8

□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□ 1-2-5 □□□□

路径覆盖

基本思想

覆盖程序中的所有可能的执行路径。

□□□□	□□□□	□□□□	□□□□
□□□ a=2 □ b=1 □ c=6	a>0,b>0 a>1,c>1	1-2-4	1 □ 5
□□□ a=1 □ b=1 □ c=-3	a>0,b>0 a<=1,c<=1	1-2-5	1 □ 8
□□□ a=-1 □ b=2 □ c=3	a<=0,b>0 a<=1,c>1	1-3-4	3 □ 7
□□□ a=-1 □ b=-2 □ c=-3	a<=0,b<=0 a<=1,c<=1	1-3-5	4 □ 8

路径覆盖

□□□□□前面的测试用例完全覆盖所有路径，但没有覆盖所有条件组合。

下面结合条件组合和路径覆盖两种方法重新设计测试用例：

□□□□	□□□□	□□□□	□□□□
□□□ a=2 □ b=1 □ c=6	a>0,b>0 a>1,c>1	1-2-4	1 □ 5
□□□ a=1 □ b=1 □ c=-3	a>0,b>0 a<=1,c<=1	1-2-5	1 □ 8
□□□ a=2 □ b=-1 □ c=-2	a>0,b<=0 a>1,c<=1	1-3-4	2 □ 6
□□□ a=-1 □ b=2 □ c=3	a<=0,b>0 a<=1,c>1	1-3-4	3 □ 7
□□□ a=-1 □ b=-2 □ c=-3	a<=0,b<=0 a<=1,c<=1	1-3-5	4 □ 8

基本路径测试

基本路径测试是在程序控制流图的基础上，通过分析控制构造的环路复杂性，导出基本可执行路径集合，从而设计测试用例的方法。

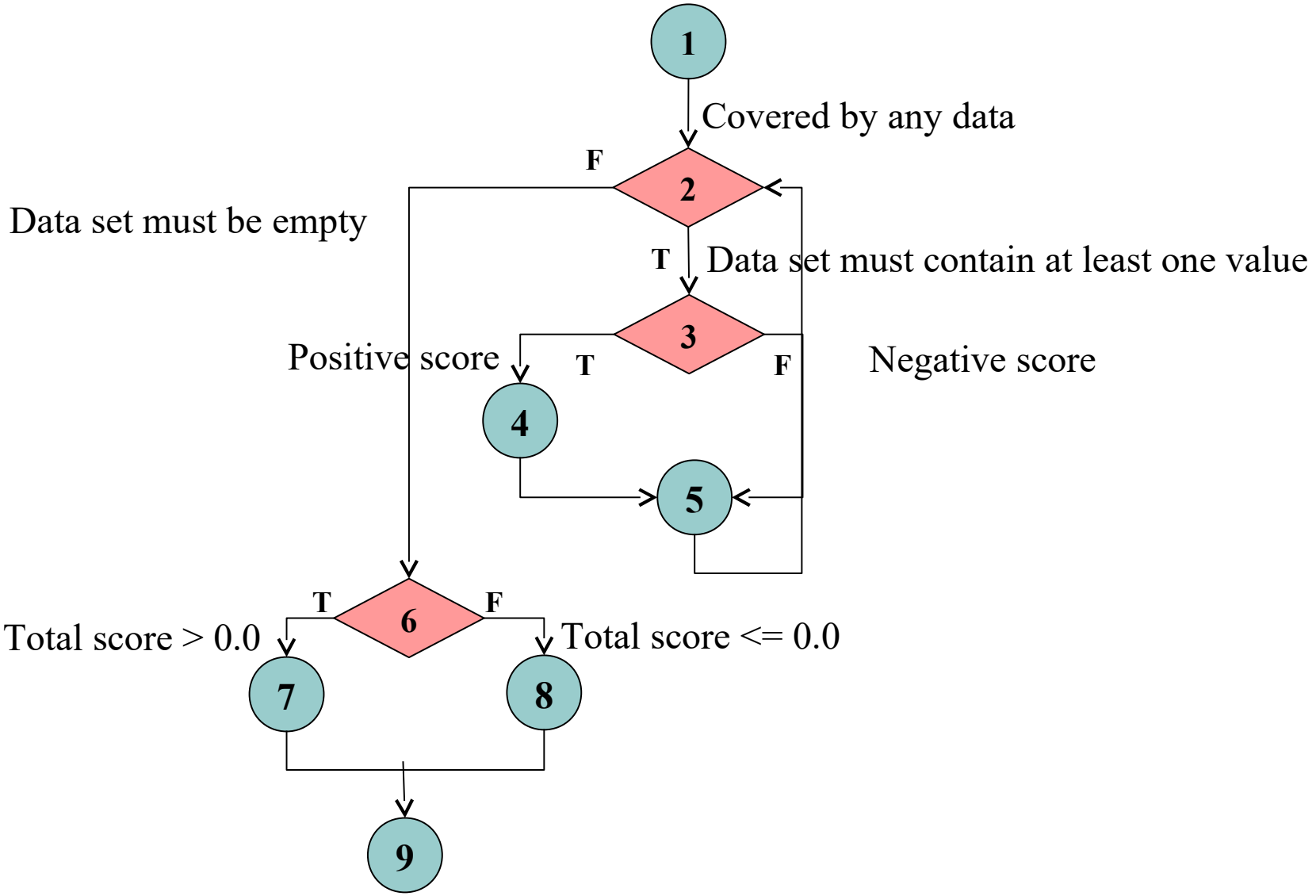
具体测试步骤如下：

- 绘制程序控制流图：描述程序控制流的一种图示方法；
- 计算程序环路复杂度：环路复杂度是一种为程序逻辑复杂性提供的软件度量，可用于计算程序的基本独立路径数。
- 确定基本路径：通过程序的控制流图导出基本路径集。
- 设计测试用例：确保基本路径集中的每一条路径被执行一次。

示例：基本路径测试

```
FindMean(float *mean, FILE *fp)
{
    float sum = 0.0, score = 0.0;
    int num = 0;

    fscanf(fp, "%f", &score); /* Read and parse into score */
    while (!EOF(fp)) {
        if (score > 0.0) {
            sum += score;
            num++;
        }
        fscanf(fp, "%f", &score);
    }
    /* Compute the mean and print the result */
    if (num > 0) {
        *mean = sum/num;
        printf("The mean score is %f \n", mean);
    } else
        printf("No scores found in file\n");
}
```

示例：基本路径测试


1
Covered by any data
F
2
Data set must be empty
T
Data set must contain at least one value
3
Positive score
T
F
Negative score
4
5
T
6
F
Total score > 0.0
Total score <= 0.0
7
8
9

示例：基本路径测试

环路复杂性：

V(G) □ □□□□

V(G) □ □□ □ □□□ □ 2

V(G) □ □□□□□ □ 1

示例的结果为 4，代表基本路径数

基本路径集：

P1 □ 1 □ 2 □ 3 □ 4 □ 5 □ 2 □•••

P2 □ 1 □ 2 □ 3 □ 5 □ 2 □•••

P3 □ 1 □ 2 □ 6 □ 7 □ 9

P4 □ 1 □ 2 □ 6 □ 8 □ 9

示例：基本路径测试

设计测试用例：

① □□□ fp≠NULL □□□□□□ {1.0} □□□□□ P1

□□□□□□□“The mean score is 1.0”

① □□□ fp≠NULL □□□□□□ {-1.0} □□□□□ P2

□□□□□□□“No scores found in file”

② □□□ fp≠NULL □□□□□□□□□□□□ P3

□□□□□□□□□

① □□□ fp≠NULL □□□□□□□□□□□□ P4

□□□□□□□“No scores found in file”

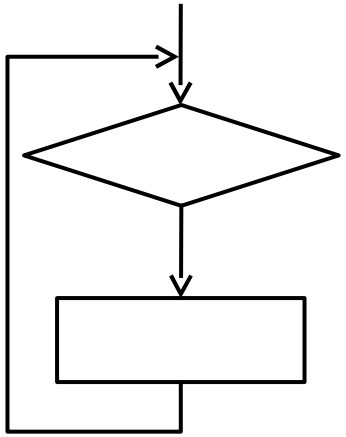
循环测试

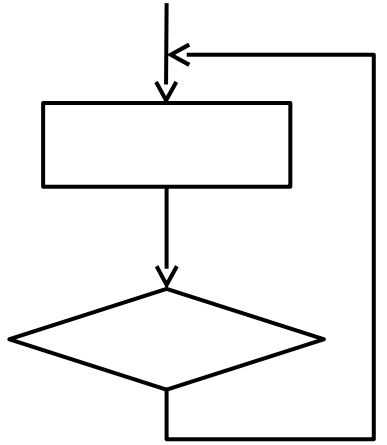
□□□□

- □□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□ n □

- □□□□□□
- □□□ 1 □
- □□□ 2 □
- □□ m □ m < n □□
- □□□□ n-1, n, n+1 □



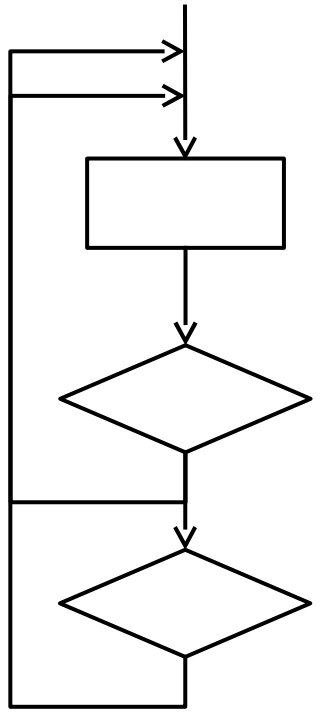


循环测试

嵌套循环

•从最内层循环开始，所有外层循环次数设为最小值；

- 对最内层循环按照简单循环方法进行测试；
- 由内向外进行下一个循环的测试，本层循环的所有外层循环仍取最小值，而由本层循环嵌套的循环取某些“典型”值；
- 重复上一步的过程，直到测试完所有循环。



串接循环

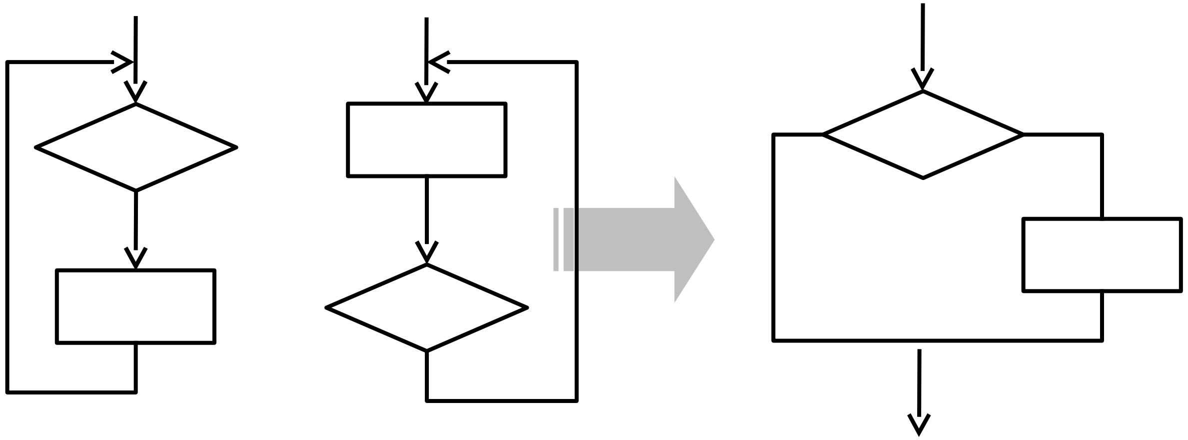
- 独立循环：分别采用简单循环的测试方法；
- 依赖性循环：采用嵌套循环的测试方法。

循环测试

Z □□□□□□□□□□

•□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

- □□
  □□□□□□□□□□□□□□□□



□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□
□□、□□□□□

教学提纲

4 □□□□□□

- □□□□□□□
- □□□□□□□
- □□□
- □□□□□

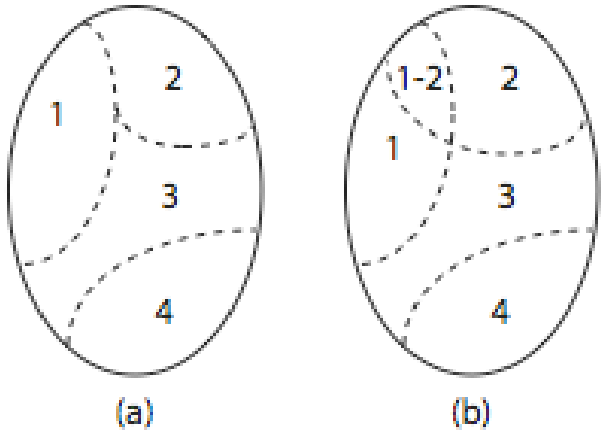
黑盒测试方法

黑盒测试并不是白盒测试的替代品，而是用于辅助白盒测试发现其他类型错误。
通常由独立测试人员根据用户需求文档来进行，但不一定要求用户参与。

等价类划分

等价类划分是将输入域划分成尽可能少的若干子域，在划分中要求每个子域两两互不相交，每个子域称为一个等价类。

等价类划分的原则是用同一等价类中的任意输入对软件进行测试，软件都输出相同的结果。因此，只需从每个等价类中选取一个输入作为测试用例，从而构成软件的完整测试用例集。

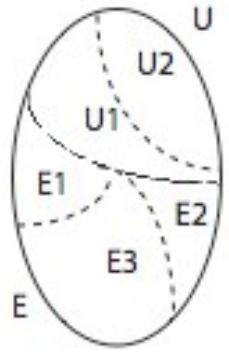


- (a) 图等价类之间互不相交，构成输入域的一个划分；(b) 图不构成一个划分。
- 同一输入域的等价类划分可能不唯一。
- 对于相同的等价类划分，不同测试人员选取的测试用例集可能是不同的，测试用例集的故障检测效率取决于人员经验。

等价类类型

有效等价类是对规格说明有意义、合理的输入数据构成的集合。利用有效等价类，能够检验程序是否实现了规格说明中预先规定的功能和性能。

无效等价类是对规格说明无意义、不合理的输入数据构成的集合。利用无效等价类，可以发现程序异常处理的情况，检查被测对象的功能和性能的实现是否有不符合规格说明要求的地方。

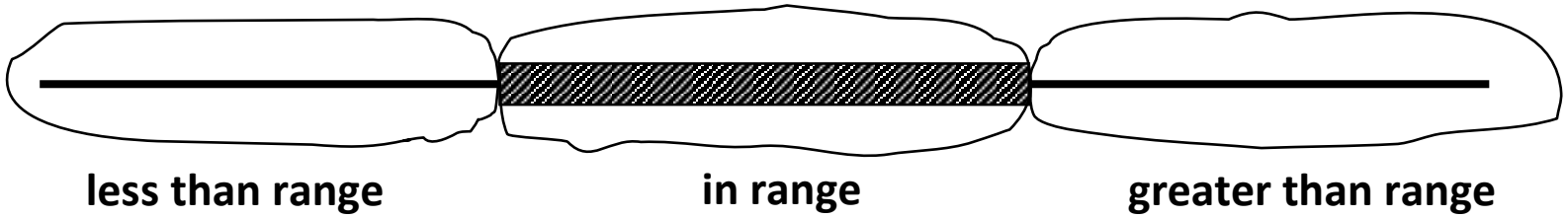


- E □□□□□□□□□□□□
- U □□□□□□□□□□□□

设计测试用例时要同时考虑这两种等价类，因为软件不仅要能接收合理的数据，也要能经受意外的考验。

变量的等价类

取值范围：在输入条件规定了取值范围的情况下，可以确定一个有效等价类和两个无效等价类。



□□□□□□□□□ x □□□ 100 □□ 10 □□□□

1 □□□□□□□ 10 □ x □ 100

2 □□□□□□□ x≤10 □ x≥100

变量的等价类

字符串：在规定了输入数据必须遵守的规则情况下，可确定一个有效等价类（符合规则）和若干个无效等价类（从不同角度违反规则）。

□□□□□□□□□□ 20 □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

1 □□□□□□□□□□□□□□□□□□□□

3 □□□□□□□□

- □□□□
- □□□□ 20 □□□□
- □□□□□□□□□□□□□□□□

变量的等价类

[illegible] N [illegible] N [illegible]

[illegible]

4 [illegible]

1 [illegible]

[illegible]

变量的等价类

数组：数组是一组具有相同类型的元素的集合，数组长度及其类型都可以作为等价类划分的依据。

举例：假设某程序的输入是一个整数数组 **`int oper[3]`**

1 个有效等价类：所有合法值的数组，如 **`{-10,20}`**

2 个无效等价类：

- 空数组
- 所有大于期望长度的数组，如 **`{-9,0,12,15}`**

如果对数组元素有其他附加约束，例如数组 **oper** □□□□□□□□ **[-3,3]** □□□□□□□□□□□□□□

变量的等价类

复合数据类型：复合数据类型是包含两个或两个以上相互独立的属性的输入数据，在进行等价类划分时需要考虑输入数据的每个属性的合法和非法取值。

举例：

```
struct student {
string name;
string course[100];
int grade[100];
}
```

□□□

等价类组合

测试用例生成：测试对象通常有多个输入参数，如何对这些参数等价类进行组合测试，来保证等价类的覆盖率，是测试用例设计首先需要考虑的问题。

所有有效等价类的代表值都集成到测试用例中，即覆盖有效等价类的所有组合。任何一个组合都将设计成一个有效的测试用例，也称正面测试用例。

无效等价类的代表值只能和其他有效等价类的代表值（随意）进行组合。因此，每个无效等价类将产生一个额外的无效测试用例，也称负面测试用例。

等价类的组合将产生数以百计的测试用例，
如何有效地减少测试用例的数目？

选取等价类的测试用例

- 由所有代表值组合而成的测试用例按使用频率（典型的使用特征）进行排序，并按照这个序列设置优先级，这样就能仅对相关的测试用例（典型的组合）进行测试。
- 优先考虑包含边界值或者边界值组合的测试用例。
- 将一个等价类的每个代表值和其他等价类的每个代表值进行组合来设计测试用例（即双向组合代替完全组合）。
- 保证满足最小原则：一个等价类的每个代表值至少在一个测试用例中出现。
- 无效等价类的代表值不与其他无效等价类代表值进行组合。

选取等价类的测试用例

[illegible]

- [illegible]
- [illegible]

[illegible] GUI [illegible]

举例：判断三角形类型

□□□□□□a□b□c□□

□□□□□□□□□□□a□b□c□□□□□□□□□

□□1□1≤a≤100

□□2□1≤b≤100

□□3□1≤c≤100

□□4□a<b+c

□□5□b<a+c

□□6□c<a+b

□□□□□□□□□□□□□□□□□□□□□□

举例：判断三角形类型

□□□□□□

①□□□□□□□

{0 □ >0 □ <0} □ {0 □ 1 □ >1 □ <0}

②□□□□□□□□□□

{□□□□□□□□□□□□□□□□□□□□□□□□□□□}

□□□

□□□□□□□□

R1 □ { <a,b,c> □□ a □ b □ c □□□□□□□ }
R2 □ { <a,b,c> □□ a □ b □ c □□□□□□□ }
R3 □ { <a,b,c> □□ a □ b □ c □□□□□□□ }
R4 □ { <a,b,c> □□ a □ b □ c □□□□□□□ }

举例：判断三角形类型

标准等价类测试用例

□□	□□□□□□	□□□□			□□□□
		a	b	c	
1	a>0 □ b>0 □ c>0 a+b>c □ b+c>a □ a+c>b a=b=c	10	10	10	□□□□□
2	a>0 □ b>0 □ c>0 a+b>c □ b+c>a □ a+c>b a=b≠c □ b=c≠a □ a=c≠b	10	10	5	□□□□□
3	a>0 □ b>0 □ c>0 a+b>c □ b+c>a □ a+c>b a≠b≠c	3	4	5	□□□□□
4	a>0 □ b>0 □ c>0 a+b≤c □ b+c≤a □ a+c≤b	4	1	2	□□□□

举例：判断三角形类型

健壮等价类测试用例

□□	□□□□□□	□□□□			□□□□
		a	b	c	
1-4	□□□□□□□□□□□□				
5	a<0 □ b>0 □ c>0	-1	5	5	a □□□
6	a>0 □ b<0 □ c>0	5	-1	5	b □□□
7	a>0 □ b>0 □ c<0	5	5	-1	c □□□
8	a>100 □ b>0 □ c>0	101	5	5	a □□□
9	a>0 □ b>100 □ c>0	5	101	5	b □□□
10	a>0 □ b>0 □ c>100	5	5	101	c □□□

边界值分析

边界值分析是对输入或输出的边界值进行测试的一种方法，它通常作为等价类划分法的补充，这种情况下的测试用例来自等价类的边界。

•首先确定边界情况。通常输入或输出等价类的边界就是应该着重测试的边界情况。

•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，而不是选取等价类中的典型值或任意值。

实践表明：大多数故障往往发生在输入定义域或输出值域的边界上，而不是在其内部。因此，针对各种边界情况设计测试用例，通常会取得很好的测试效果。

边界值分析

□□□	□□□	□□□□□□□□□
□□	□□□ 1 □□□ □□□ 1 □□□	□□□□□□□□□□□□□ 1 □ 255 □□□□□□ 1 □□ 255 □□□□□□□□□□□ □□ 0 □□ 256 □□□□□□□ □□□□□□□□□□□□□□□□□□□
□□	□□□□ 1 □□□□ 1	□□□□□□□□□ 5 □□□□□□□□□□□□ 10000 □□ □□□□ 99999 □□□□□□□□□□□□□□ 5 □□□□ 5 □□□□□□□□□□
□□	□□□□□□□□ □□□□□□□	□□□□ U □□□□□□□□□□□□□□□□□□□□ KB □□□□□□□□□□

边界值分析

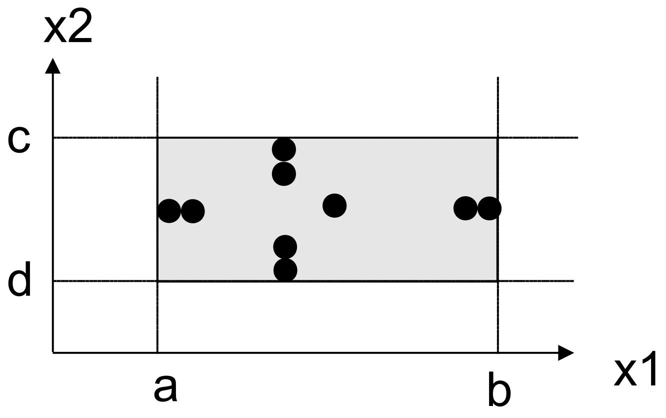
□□□□□□□□□□□□□□□□□□□□□□□□

•□□□□□□□□□□□□□□□□□“□□□”□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

•□□□

min □ min+ □ nom □ max- □ max □

□□□□□ F □□□□□ x1 □ a≤x1≤b □□ x2 □ c≤x2≤d □



$<x1_{nom}, x2_{min}>$ $<x1_{nom}, x2_{min+}>$ $<x1_{nom}, x2_{nom}>$

$<x1_{nom}, x2_{max}>$ $<x1_{nom}, x2_{max-}>$ $<x1_{min}, x2_{nom}>$

$<x1_{min+}, x2_{nom}>$ $<x1_{max}, x2_{nom}>$ $<x1_{max-}, x2_{nom}>$

举例：判断三角形类型

□□	□□□□□□	□□□□			□□□□
		a	b	c	
1	a>0 □ b>0 □ c>0 a+b>c □ b+c>a □ a+c>b a=b=c	60	60	60	□□□□□
2	a>0 □ b>0 □ c>0 a+b>c □ b+c>a □ a+c>b a=b≠c □ b=c≠a □ a=c≠b	60	60	1	□□□□□
3		60	60	2	
4		50	50	99	
5		60	1	60	
6		60	2	60	

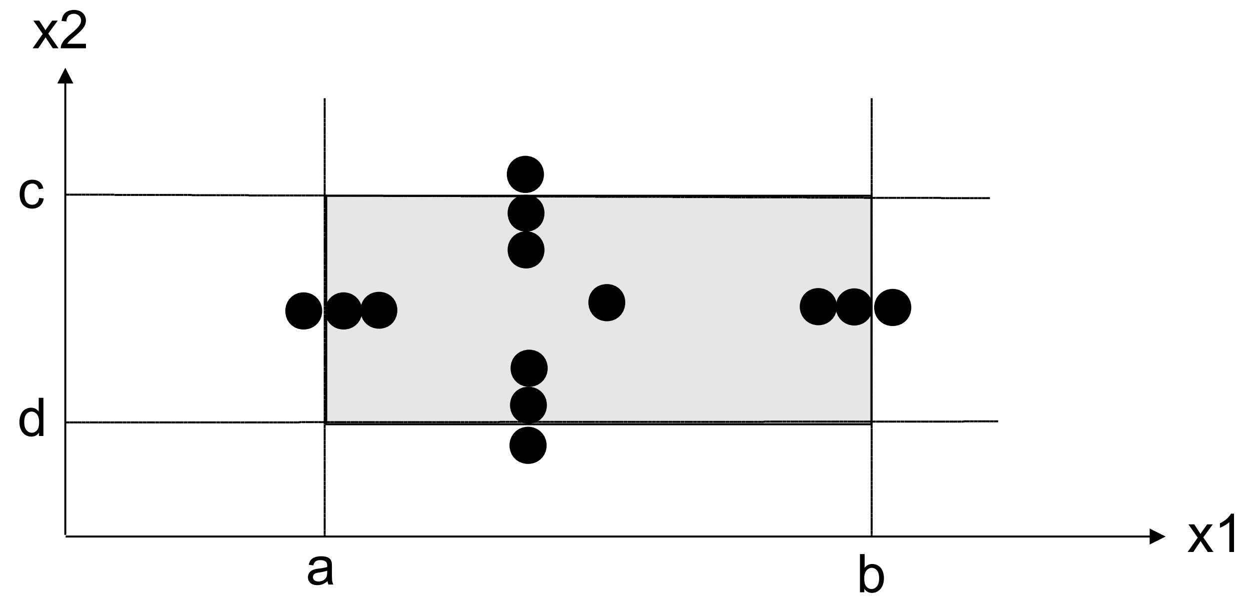
举例：判断三角形类型

□□	□□□□□□	□□□□			□□□□
		a	b	c	
7	a>0 □ b>0 □ c>0 a+b>c □ b+c>a □ a+c>b a=b≠c □ b=c≠a □ a=c≠b	50	99	50	□□□□□
8		1	60	60	
9		2	60	60	
10		99	50	50	
11	a>0 □ b>0 □ c>0 a+b≤c □ b+c≤a □ a+c≤b	50	50	100	□□□□
12		50	100	50	
13		100	50	50	

健壮性测试

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□5 □□□□□□□□□□□□□□□□□□□□□□

□□ max+ □□□□□□□□□□ min- □□□□□□□□□□□□□□□□□□□



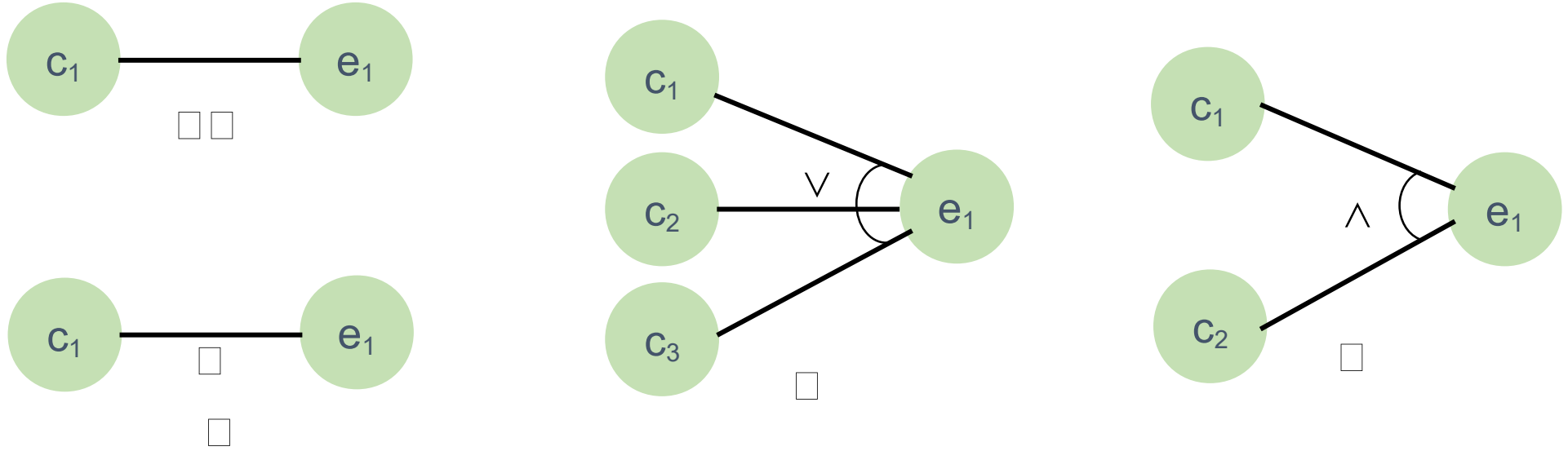
因果图

因果图是一种利用图解法分析输入的各种组合情况，从而设计测试用例的方法，适合于检查程序输入条件的各种组合情况。

- 等价类划分和边界值分析方法都是着重考虑输入条件，但没有考虑输入条件的各种组合、输入条件之间的相互制约关系。这样虽然各种输入条件可能出错的情况已经测试到了，但多个输入条件组合起来可能出错的情况却被忽视了。
- 如果在测试时必须考虑输入条件的各种组合，则可能的组合数目将是天文数字，因此必须考虑采用一种适合于描述多种条件的组合、相应产生多个动作的形式来进行测试用例的设计，这就需要利用因果图。

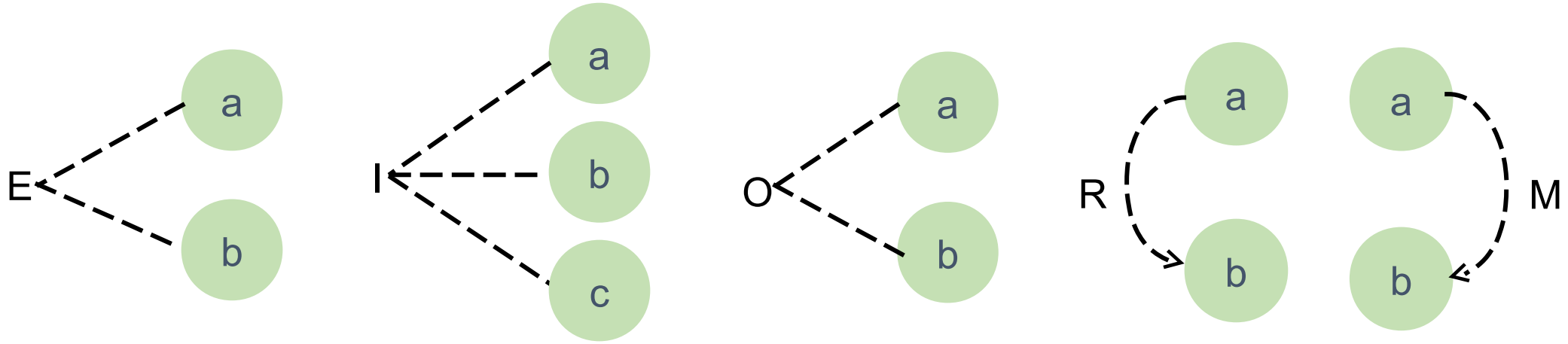
因果图

基本符号：用于表示四种因果关系



因果图

约束符号：用于表示约束关系



E □□□□□□ a □ b □□□□□□□□□ 1 □□ a □ b □□□□□ 1 □

I □□□□□□ a □ b □ c □□□□□□□□□ 1 □□ a □ b □ c □□□□□ 0 □

O □□□□□□□ a □ b □□□□□□□□□□□ 1 □

R □□□□□□□ a □ 1 □□ b □□□ 1 □□ a □ 1 □□ b □□□ 0 □

M □□□□□□□□□□ a □ 1 □□□□ b □□□ 0 □

因果图

①

②

③

-
-
-

因果图

[illegible] # [illegible] * [illegible]
[illegible] # [illegible] * [illegible] N [illegible] M [illegible]

因果图

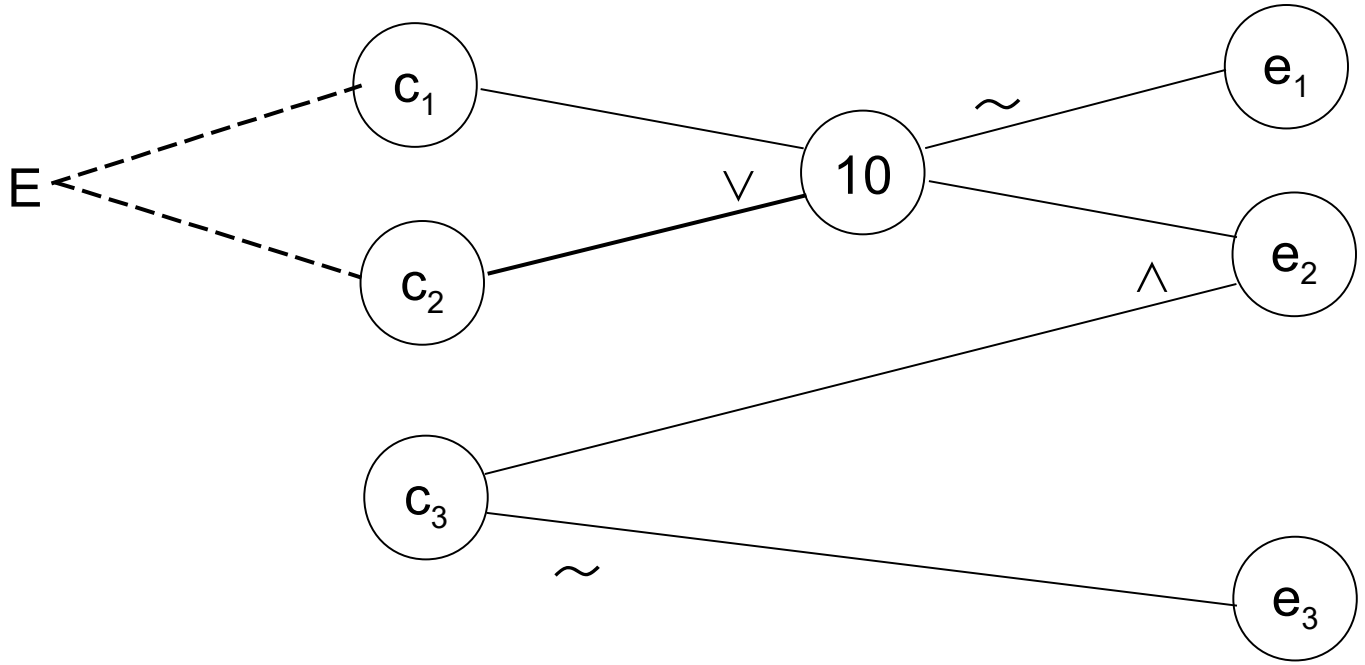
举例：某程序输入的第一个字符必须是 # 或 *，第二个字符必须是一个数字，此情况下进行文件的修改；如果第一个字符不是 # 或 *，则给出信息 N，如果第二个字符不是数字，则给出信息 M 。

□□□□□□□□□□□□□□□□□□□□

□□	□□
c1 □□□□□□□ #	e1 □□□□□ N
c2 □□□□□□□ *	e2 □□□□□
c3 □□□□□□□□□□□	e3 □□□□□ M

因果图

□□□□□□□□□



□□□□□□ 10 □□□□□□□□□□□□□□□□□□□

因果图

□□□□□□□□□□□□□□□□□□□□

		1	2	3	4	5	6	7	8
□□	c1	1	1	1	1	0	0	0	0
	c2	1	1	0	0	1	1	0	0
	c3	1	0	1	0	1	0	1	0
	10			1	1	1	1	0	0
□□	e1							√	√
	e2			√		√			
	e3				√		√		√
	□□□	√	√						
□□□□				#3	#A	*6	*B	A1	GT

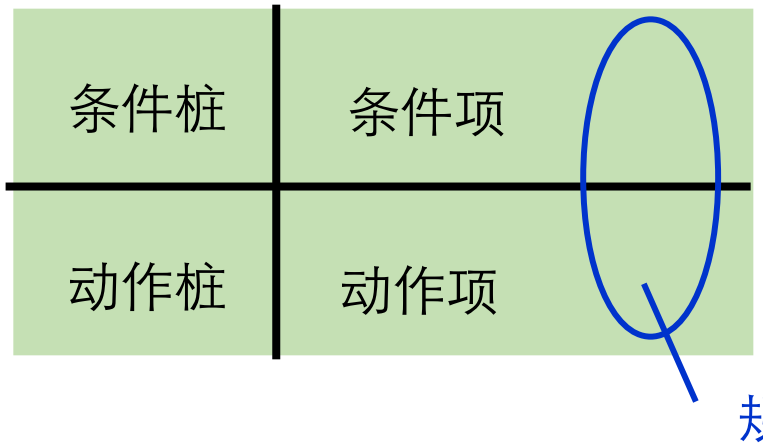
因果图

第四步：根据决策表中的每一列设计测试用例

□□□□□□	□□□□	□□□□
1	#3	□□□□
2	#A	□□□□ M
3	*6	□□□□
4	*B	□□□□ M
5	A1	□□□□ N
6	GT	□□□□ N □□□ M

决策表

适用情况：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。



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：列出问题规定的可能采取的操作
动作项：指出在条件项的各组取值情况下应采取的动作

将任何一个条件组合的特定取值及相应要执行的动作称为一条规则。
在决策表中贯穿条件项和动作项的一列就是一条规则。

决策表

□□□□□□□

- □□□□□□□□□□
- □□□□□□□□□□□ n □□□□□□□□□□□□□□□ 0 □ 1 □□□□ 2^n □□□□
- □□□□□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□

规则		规则 1	规则 2	.....	规则 p
条件桩	条件项 1				
	条件项 2				
	.....				
	条件项 m				
动作桩	动作项 1				
	动作项 2				
	.....				
	动作项 n				

举例：判断三角形类型

□□□□□□a□b□c□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□a□b□c□□□□□□□□□

□□1□1≤a≤100　　□□4□a<b+c

□□2□1≤b≤100　　□□5□b<a+c

□□3□1≤c≤100　　□□6□c<a+b

□□□□□□□□□□□□□□□□□□□□

举例：判断三角形类型

□□		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
□□□	C1 □ a □ b □ c □□□□□□																
	C2 □ a=b?																
	C3 □ a=c?																
	C4 □ b=c?																
□□□	A1 □□□□□																
	A2 □□□□□□□																
	A3 □□□□□□																
	A4 □□□□□□																
	A5 □□□□□□																

举例：判断三角形类型

□□		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
□□□	C1 □ a □ b □ c □□□□□□	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N
	C2 □ a=b?	Y	Y	Y	Y	N	N	N	N	Y	Y	Y	Y	N	N	N	N
	C3 □ a=c?	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N
	C4 □ b=c?	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N
□□□	A1 □□□□□									Y	Y	Y	Y	Y	Y	Y	Y
	A2 □□□□□□□								Y								
	A3 □□□□□□				Y		Y	Y									
	A4 □□□□□□	Y															
	A5 □□□□□□		Y	Y		Y											

举例：判断三角形类型

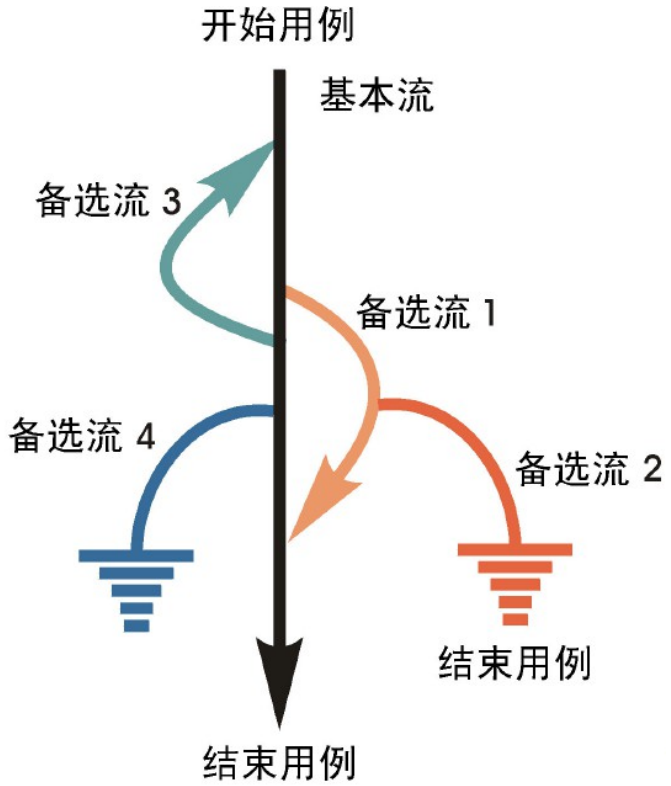
□□		1	2	3	4	5	6
□□□	C1 □ a □ b □ c □□□□□□	Y	Y	Y	Y	Y	N
	C2 □ a=b?	Y	Y	N	N	N	-
	C3 □ a=c?	Y	N	Y	N	N	-
	C4 □ b=c?	Y	N	N	Y	N	-
□□□	A1 □□□□□						Y
	A2 □□□□□□□					Y	
	A3 □□□□□□		Y	Y	Y		
	A4 □□□□□□	Y					

举例：判断三角形类型

□□	□□□□□□	□□□□
1	C1 □ C2 □ C3 □ C4 □□□□	□□□□□
2	C1 □ C2 □□□ C3 □ C4 □□□	□□□□□
3	C1 □ C3 □□□ C2 □ C4 □□□	□□□□□
4	C1 □ C4 □□□ C2 □ C3 □□□	□□□□□
5	C1 □□□ C2 □ C3 □ C4 □□□	□□□□□□
6	C1 □□□	□□□□

场景法

场景法通过运用场景来对系统的功能点或业务流程的描述，从而提高测试效果的一种方法。



□□ 1	□□□
□□ 2	□□□□□□□ 1
□□ 3	□□□□□□□ 1 □□□□ 2
□□ 4	□□□□□□□ 3
□□ 5	□□□□□□□ 3 □□□□ 1
□□ 6	□□□□□□□ 3 □□□□ 1 □□□□ 2
□□ 7	□□□□□□□ 4
□□ 8	□□□□□□□ 3 □□□□ 4

□□□ ATM □□

用例描述

□□□	□□□□
□□ 2	ATM □□□□□
□□ 3	ATM □□□□□
□□ 4	PIN □□□□□□□□□□
□□ 5	PIN □□□□□□□□□□
□□ 6	□□□□□ / □□□□□□
□□ 7	□□□□□□

□□□ ATM □□

场景设计

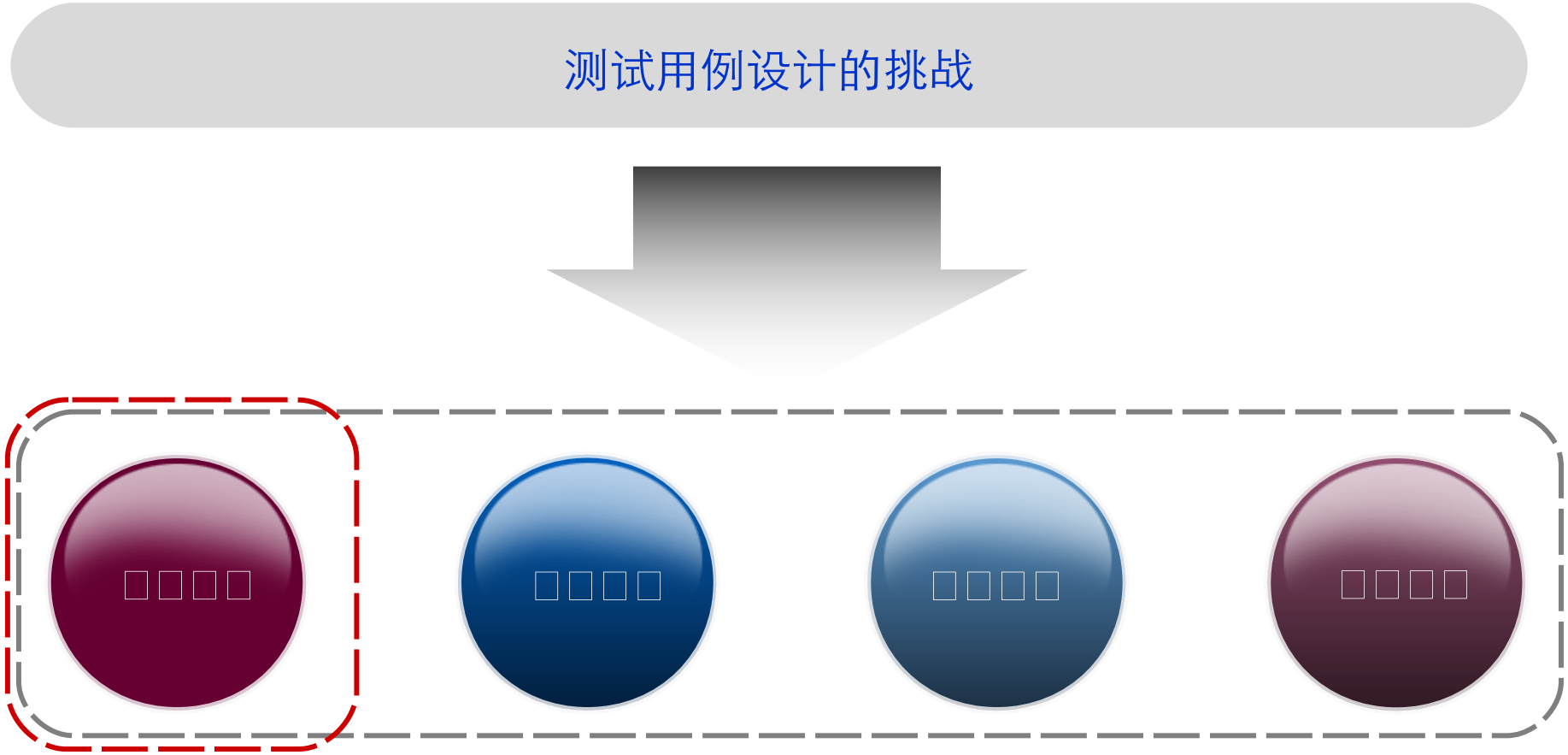
□□ 1 □□□□□	□□□
□□ 2 □ ATM □□□□□	□□□□□□□ 2
□□ 3 □ ATM □□□□□	□□□□□□□ 3
□□ 4 □ PIN □□□□□□□□□□	□□□□□□□ 4
□□ 5 □ PIN □□□□□□□□□□	□□□□□□□ 5
□□ 6 □□□□□□ / □□□□□□	□□□□□□□ 6
□□ 7 □□□□□□□	□□□□□□□ 7

□□□ ATM □□

测试用例设计

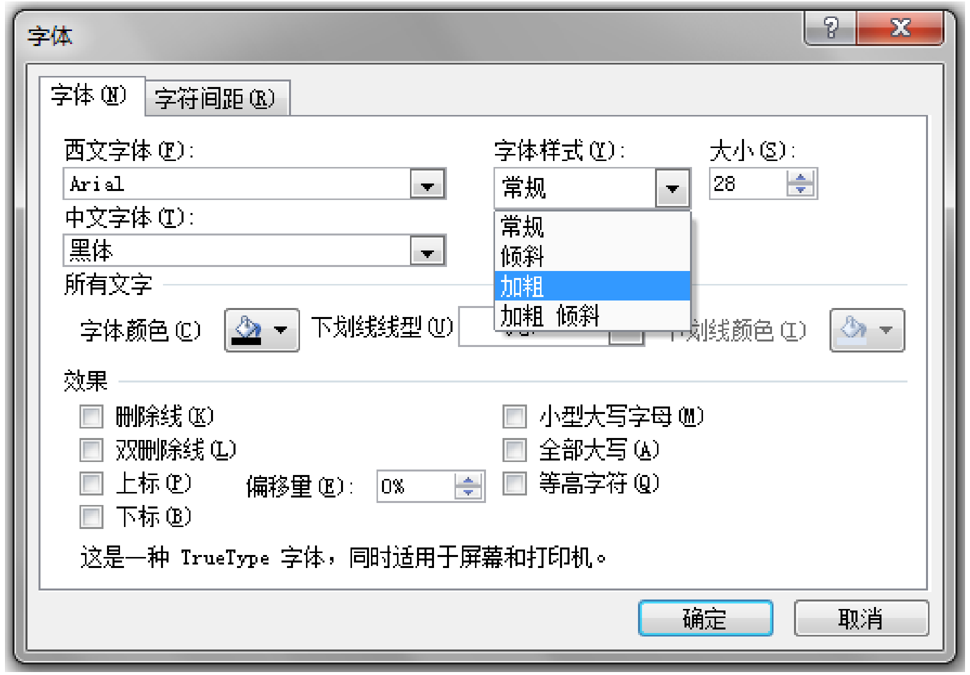
□□	□□	PIN	□□	□□□□	□□□□	ATM □□	□□□□
1	□□ 1 □□□□□	V	V	V	V	V	□□□□
2	□□ 2 □ ATM □□□□□	V	V	V	V	X	□□□□□□□□□□□
3	□□ 3 □ ATM □□□□□	V	V	V	V	X	□□□□□□□□□□□□□□□□□□□□
4	□□ 4 □ PIN □□ □□□□□□□□□□	X	V	n/a	V	V	□□□□□□□□□□□□□□□□□□□□ □□□ PIN
5	□□ 4 □ PIN □□ □□□□□□□□	X	V	n/a	V	V	□□□□□□□□□□□□□□□□□□□□ □□□ PIN
6	□□ 5 □ PIN □□ □□□□□□□□□	X	V	n/a	V	V	□□□□□□□□□□□□□□□
	.....						

测试用例设计的挑战



输入参数组合

□□
□□□□□□□□□



① □□□□

② □□□□

③ □□□□

④ □□

⑤ □□□□

⑥ □□

⑦ ……

平台参数组合

[illegible]

[illegible]

- [illegible] IE6, IE7, IE8, IE9, Firefox14, Opera11, Safari5, Chrome18
- [illegible] RealPlayer, MediaPlayer, None
- [illegible] Win2000, WinXP, Win2003, Win7, Win8, MacOS
- Web [illegible] IIS, Apache, Weblogic
- [illegible] WinNT, Win2003, Linux

[illegible] 8×3×6×3×3=1296 [illegible]

组合测试的策略

组合太多：测试输入、平台、路径等所有组合的规模太大

测试人员可以采取的常用策略：

- 尝试测试所有输入的组合，延期项目，可能是失去产品市场
- 选择一些容易设计和执行的测试用例，难以发现质量问题
- 罗列所有的组合，再随机选择其中的子集进行测试
- 采取合适的测试技术，选择能发现大部分缺陷的子集进行测试

建议：选用组合设计技术，从参数组合的完全集中选择一个较小的子集，其目的是发现因参数组合而引起的故障。

组合测试设计技术

- 模型是由一组参数及其对应的值组成，输入空间或环境建模并不互斥，可根据被测程序要求同时对二者或其中之一建模。
- 组合设计由一个 $N\times k$ 的矩阵表示成一个组合对象，其中 N □□□□□□□□□□□□□□□□□□ k □□□□□□□□□□□□□
- □□□

常用的正交矩阵

□□□□□□□□□□□□□□□□□□□□

$L_4(2^3)$ $L_8(2^7)$ $L_9(3^4)$ $L_{16}(4^5)$

□□□□□□□□□□□□□□□□□□□□

$L_8(4\times2^4)$ $L_{12}(3\times2^3)$ $L_{12}(3\times2^4)$ $L_{16}(2^6\times4^3)$

http://www.research.att.com/~njas/oadir

常用的正交矩阵

$L_4(2^3)$

	1	2	3
1	1	1	1
2	1	2	2
3	2	1	2
4	2	2	1

$L_8(2^7)$

	1	2	3	4	5	6	7
1	1	1	1	1	1	1	1
2	1	1	1	2	2	2	2
3	1	2	2	1	1	2	2
4	1	2	2	2	2	1	1
5	2	1	2	1	2	1	2
6	2	1	2	2	1	2	1
7	2	2	1	1	2	2	1
8	2	2	1	2	1	1	2

常用的正交矩阵

$L_9(3^4)$

	1	2	3	4
1	1	1	1	1
2	1	2	2	2
3	1	3	3	3
4	2	1	2	3
5	2	2	3	1
6	2	3	1	2
7	3	1	3	2
8	3	2	1	3
9	3	3	2	1

常用的正交矩阵

$L_{16}(4^5)$

	1	2	3	4	5
1	1	1	1	1	1
2	1	2	2	2	2
3	1	3	3	3	3
4	1	4	4	4	4
5	2	1	2	3	4
6	2	2	1	4	3
7	2	3	4	1	2
8	2	4	3	2	1
9	3	1	3	4	2
10	3	2	4	3	1
11	3	3	1	2	4
12	3	4	2	1	3
13	4	1	4	2	3
14	4	2	3	1	4
15	4	3	2	4	1
16	4	4	1	3	2

常用的正交矩阵

$L_8(4\times2^4)$

	1	2	3	4	5
1	1	1	1	1	1
2	1	2	2	2	2
3	2	1	1	2	2
4	2	2	2	1	1
5	3	1	2	1	2
6	3	2	1	2	1
7	4	1	2	2	1
8	4	2	1	1	2

常用的正交矩阵

$L_{12}(3\times2^3)$

	1	2	3	4
1	1	1	1	1
2	1	1	2	1
3	1	2	1	2
4	1	2	2	2
5	2	1	1	2
6	2	1	2	2
7	2	2	1	1
8	2	2	2	1
9	3	1	1	2
10	3	1	2	1
11	3	2	1	1
12	3	2	2	2

常用的正交矩阵

$L_{12}(3\times2^4)$

	1	2	3	4	5
1	1	1	1	1	1
2	1	1	1	2	2
3	1	2	2	1	2
4	1	2	2	2	1
5	2	1	2	1	1
6	2	1	2	2	2
7	2	2	1	2	2
8	2	2	1	2	2
9	3	1	2	1	2
10	3	1	1	2	1
11	3	2	1	1	2
12	3	2	2	2	1

常用的正交矩阵

$L_{16}(2^6 \times 4^3)$

	1	2	3	4	5	6	7	8	9
1	1	1	1	1	1	1	1	1	1
2	2	2	1	2	1	2	1	3	3
3	1	2	2	2	2	1	3	1	3
4	2	1	2	1	2	2	3	3	1
5	1	1	2	2	2	2	1	4	4
6	2	2	2	1	2	1	1	2	2
7	1	2	1	1	1	2	3	4	2
8	2	1	1	2	1	1	3	2	4
9	2	2	1	1	2	2	4	1	4
10	1	1	1	2	2	1	4	3	2
11	2	1	2	2	1	2	2	1	2
12	1	2	2	1	1	1	2	3	4
13	2	2	2	2	1	1	4	4	1
14	1	1	2	1	1	2	4	2	3
15	2	1	1	1	2	1	2	4	3
16	1	2	1	2	2	2	2	2	1

基于正交矩阵的测试用例设计

①

②

③

④

⑤

□□□ Powerpoint □□□□

Powerpoint □□□□□□□□□□

□□□□□	□□□□□□□□□□□□□
□□□□□	□□□□□□□□□□□□□□□
□□□□/□□□	□□□□□□□□
□□□□□	□□□□□□□□□□□□

□□□

- □□□□□□□□□□□ 3×4×3×2=72 □□□□□□□□□□□□□□□□□□□□□□□□ □
- □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□ Powerpoint □□□□

□□ 1 □□□□□□□□□□□□□□□□□□□□□□□□

□□□ □□	A □□□□	B □□□□	C □□□□ / □□	D □□□□
1	□□□ A1 □	□□□□ B1 □	□□□ C1 □	□□□□□□ D1 □
2	□□□□□□ A2 □	□□□ B2 □	□□□ C2 □	□□□□□□□ D2 □
3	□□□□□ A3 □	□□□□ B3 □	□□□ C3 □	
4		□□□□□ B4 □		

□□ 2 □□□□□□□□□□□□□

- □□□□□□□ 4 □□□□□□ 4 □□□□□□□ ≥ 2
- □□□□□□□□□□□□□□□ 2×(3-1)+(4-1)+(2-1)+1 □ 9
- □□□□□□□□□□ $L_{16}(4^5)$

$L_{16}(4^5)$

	1	2	3	4	5
1	1	1	1	1	1
2	1	2	2	2	2
3	1	3	3	3	3
4	1	4	4	4	4
5	2	1	2	3	4
6	2	2	1	4	3
7	2	3	4	1	2
8	2	4	3	2	1
9	3	1	3	4	2
10	3	2	4	3	1
11	3	3	1	2	4
12	3	4	2	1	3
13	4	1	4	2	3
14	4	2	3	1	4
15	4	3	2	4	1
16	4	4	1	3	2

□□ 3 □□□□□□□□□□□□□□□□

	1	2	3	4	5
1	A1	B1	C1	D1	1
2	A1	B2	C2	D2	2
3	A1	B3	C3	3	3
4	A1	B4	4	4	4
5	A2	B1	C2	3	4
6	A2	B2	C1	4	3
7	A2	B3	4	D1	2
8	A2	B4	C3	D2	1
9	A3	B1	C3	4	2
10	A3	B2	4	3	1
11	A3	B3	C1	D2	4
12	A3	B4	C2	D1	3
13	4	B1	4	D2	2
14	4	B2	C3	D1	4
15	4	B3	C2	4	1
16	4	B4	C1	3	2

	1	2	3	4	5
1	A1	B1	C1	D1	1
2	A1	B2	C2	D2	2
3	A1	B3	C3	D1	3
4	A1	B4	C1	D2	4
5	A2	B1	C2	D1	4
6	A2	B2	C1	D2	3
7	A2	B3	C2	D1	2
8	A2	B4	C3	D2	1
9	A3	B1	C3	D2	2
10	A3	B2	C3	D1	1
11	A3	B3	C1	D2	4
12	A3	B4	C2	D1	3

去掉

去掉

□□□ Powerpoint □□□□

□□ 4 □□□□□□□

□□□□	PPT-ST-FUNCTION-PRINT-001
□□□□	□□ Powerpoint □□□□
□□□□	□□ Powerpoint □□ A □□□□□□□□□□□□□
□□□□	□
□□□□	Powerpoint □□ A □□□□□□□□□□□□□□□□□
□□	□□ A □ D:\test.ppt
□□□□	1. □□□□□□□ 2. □□□□□□“□□”□ 3. □□□□□□“□□□”□ 4. □□ / □□□□“□□”□ 5. □□“□□□□□”□□□□ 6. □□“□□”□
□□□□	□□□□□□□□□□□□□□□□□

□□□ Powerpoint □□□□

□□□□	PPT-ST-FUNCTION-PRINT-002
□□□□	□□ Powerpoint □□□□
□□□□	□□ Powerpoint □□ A □□□□□□□□□□□□□□□□□
□□□□	□
□□□□	Powerpoint □□ A □□□□□□□□□□□□□□□□□
□□	□□ A □ D:\test.ppt
□□□□	1. □□□□□□□ 2. □□□□□□“□□”□ 3. □□□□□□“□□”□ 4. □□ / □□□□“□□”□ 5. □□“□□”□
□□□□	□□□□□□□□□□□□□□□□

□□□ Powerpoint □□□□

□□□□	PPT-ST-FUNCTION-PRINT-003
□□□□	□□ Powerpoint □□□□
□□□□	□□ Powerpoint □□ A □□□□□□□□□□□□
□□□□	□
□□□□	Powerpoint □□ A □□□□□□□□□□□□□□□□□
□□	□□ A □ D:\test.ppt
□□□□	1. □□□□□□□ 2. □□□□□□“□□”□ 3. □□□□□□“□□□”□ 4. □□ / □□□□“□□”□ 5. □□“□□□□□”□□□□ 6. □□“□□”□
□□□□	□□□□□□□□□□□□□□□□

□□□ Powerpoint □□□□

□□□□	PPT-ST-FUNCTION-PRINT-004
□□□□	□□ Powerpoint □□□□
□□□□	□□ Powerpoint □□ A □□□□□□□□□□□
□□□□	□
□□□□	Powerpoint □□ A □□□□□□□□□□□□□□□□□□
□□	□□ A □ D:\test.ppt
□□□□	1. □□□□□□□ 2. □□□□□□“□□”□ 3. □□□□□□“□□□□”□ 4. □□ / □□□□“□□”□ 5. □□“□□”□
□□□□	□□□□□□□□□□□□□□□□□□

□□□ Powerpoint □□□□

□□□□	PPT-ST-FUNCTION-PRINT-005
□□□□	□□ Powerpoint □□□□
□□□□	□□ Powerpoint □□ A □□□□□□□□□□□
□□□□	□
□□□□	Powerpoint □□ A □□□□□□□□□□□□□□□□□□
□□	□□ A □ D:\test.ppt
□□□□	1. □□□□□□□ 2. □□□□□□“□□□□□”□ 3. □□□□□□“□□□”□ 4. □□ / □□□□“□□”□ 5. □□“□□□□□”□□□□ 6. □□“□□”□
□□□□	□□□□□□□□□□□□□□□□□

□□□ Powerpoint □□□□

□□□□	PPT-ST-FUNCTION-PRINT-006
□□□□	□□ Powerpoint □□□□
□□□□	□□ Powerpoint □□ A □□□□□□□□□□□□
□□□□	□
□□□□	Powerpoint □□ A □□□□□□□□□□□□□□□□□
□□	□□ A □ D:\test.ppt
□□□□	1. □□□□□□□ 2. □□□□□□“□□□□□”□ 3. □□□□□□“□□”□ 4. □□ / □□□□“□□”□ 5. □□“□□”□
□□□□	□□□□□□□□□□□□□□□□□□□

□□□ Powerpoint □□□□

□□□□	PPT-ST-FUNCTION-PRINT-007
□□□□	□□ Powerpoint □□□□
□□□□	□□ Powerpoint □□ A □□□□□□□□□□□□□□□□
□□□□	□
□□□□	Powerpoint □□ A □□□□□□□□□□□□□□□□□□
□□	□□ A □ D:\test.ppt
□□□□	1. □□□□□□□ 2. □□□□□□“□□□□□”□ 3. □□□□□□“□□□”□ 4. □□ / □□□□“□□”□ 5. □□“□□□□□”□□□□ 6. □□“□□”□
□□□□	□□□□□□□□□□□□□□□□□□□□

□□□ Powerpoint □□□□

□□□□	PPT-ST-FUNCTION-PRINT-008
□□□□	□□ Powerpoint □□□□
□□□□	□□ Powerpoint □□ A □□□□□□□□□□□□□
□□□□	□
□□□□	Powerpoint □□ A □□□□□□□□□□□□□□□□□
□□	□□ A □ D:\test.ppt
□□□□	1. □□□□□□□ 2. □□□□□□“□□□□□”□ 3. □□□□□□“□□□□”□ 4. □□ / □□□□“□□”□ 5. □□“□□”□
□□□□	□□□□□□□□□□□□□□□□□□□□□

□□□ Powerpoint □□□□

□□□□	PPT-ST-FUNCTION-PRINT-009
□□□□	□□ Powerpoint □□□□
□□□□	□□ Powerpoint □□ A □□□□□□□□□□□□□□□□
□□□□	□
□□□□	Powerpoint □□ A □□□□□□□□□□□□□□□□□□
□□	□□ A □ D:\test.ppt
□□□□	1. □□□□□□□ 2. □□□□□□“□□□”□ 3. □□□□□□“□□□”□ 4. □□ / □□□□“□□”□ 5. □□“□□”□
□□□□	□□□□□□□□□□□□□□□□□□□

□□□ Powerpoint □□□□

□□□□	PPT-ST-FUNCTION-PRINT-010
□□□□	□□ Powerpoint □□□□
□□□□	□□ Powerpoint □□ A □□□□□□□□□□□□□□□□□□
□□□□	□
□□□□	Powerpoint □□ A □□□□□□□□□□□□□□□□□□
□□	□□ A □ D:\test.ppt
□□□□	1. □□□□□□□ 2. □□□□□□“□□□”□ 3. □□□□□□“□□”□ 4. □□ / □□□□“□□”□ 5. □□“□□□□□”□□□□ 6. □□“□□”□
□□□□	□□□□□□□□□□□□□□□□□□□□□

□□□ Powerpoint □□□□

□□□□	PPT-ST-FUNCTION-PRINT-011
□□□□	□□ Powerpoint □□□□
□□□□	□□ Powerpoint □□ A □□□□□□□□□□□□□□□□
□□□□	□
□□□□	Powerpoint □□ A □□□□□□□□□□□□□□□□□□
□□	□□ A □ D:\test.ppt
□□□□	1. □□□□□□□ 2. □□□□□□“□□□”□ 3. □□□□□□“□□□”□ 4. □□ / □□□□“□□”□ 5. □□“□□”□
□□□□	□□□□□□□□□□□□□□□□□□□□□□□□

□□□ Powerpoint □□□□

□□□□	PPT-ST-FUNCTION-PRINT-012
□□□□	□□ Powerpoint □□□□
□□□□	□□ Powerpoint □□ A □□□□□□□□□□□□□□□□□□□□
□□□□	□
□□□□	Powerpoint □□ A □□□□□□□□□□□□□□□□□□□□
□□	□□ A □ D:\test.ppt
□□□□	1. □□□□□□□□ 2. □□□□□□□“□□□”□ 3. □□□□□□□“□□□□”□ 4. □□ / □□□□“□□”□ 5. □□“□□□□□”□□□□ 6. □□“□□”□
□□□□	□□□□□□□□□□□□□□□□□□□□□□□□□□□□

谢谢大家！

THANKS