# 软件测试概述

## 软件测试背景

### 为什么会出现软件缺陷

一、**产品说明书(需求) (主要原因)** 随意、易变、沟通不足

二、**设计 (次要原因)** 随意、易变、沟通不足

三、**编码** 软件复杂度，进度压⼒力，低级错误

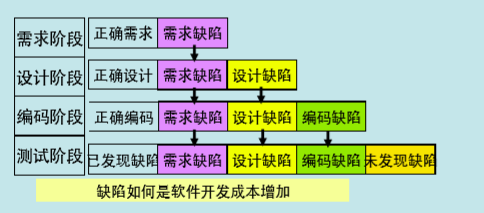
四、**其它**  理解错误，测试错误

### 软件危机

危机背景：落后的软件生产方式无法满足日趋复杂大型软件系统的开发需求（20世纪60年代）

危机表现：项目延期、经费超支、产品无法维护……

危机原因：缺乏规范化工程约束 -> 缺陷的不断积累与放大效应



缺陷修复代价——随开发时间快速增加

### 测试观点

**错误**：

1957-1978面向可运行的时代：

软件工程思想开始深入人心，软件测试定义：“**测试就是建立一种信心，认为程序能够按预期设想运行**”，核心思想：**测试是试图验证软件是工作的**

**正确：**

1979-1982面向缺陷发现的时代

Glenford J. Myers: **测试是尽可能多地发现软件错误**

Myers的软件测试定义：测试是为发现错误而执行一个程序或系统的过程

Myers的**三个重要观点**： 1. 测试是为了**证明程序有错**，而不是证明程序无错误；

2.一个**好的测试用例**是在于它能**发现至今未发现的错误**；

3.一个**成功的测试**是**发现了至今未发现的错误的测试**；

## 软件测试基本概念

### 测试和调试区别

测试，是test，就是看看程序中有什么问题（称为bug）；[软件测试](https://www.baidu.com/s?wd=%E8%BD%AF%E4%BB%B6%E6%B5%8B%E8%AF%95&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9mW7BPjF9ujb3PAc3ryR30AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBt1nHTvPjcsP1D" \t "_blank)是[软件测试](https://www.baidu.com/s?wd=%E8%BD%AF%E4%BB%B6%E6%B5%8B%E8%AF%95&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9mW7BPjF9ujb3PAc3ryR30AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBt1nHTvPjcsP1D)人员和程序员都参与的一项工作，是贯穿整个生命周期的，  
调试(debug)，就是通过种种手段，将程序中的bug给定位出来，然后解决（特别考验人解决问题的能力）主要是程序员自己参与，主要是在开发阶段。

### 软件测试目的

1.确保软件质量

2.确保软件开发过程方向的正确性

### 软件测试原理

1用户需求至上 2.测试是有计划的活动 3.缺陷出现的集群性 4.测试应从“小规模”走向“大规模” 5.穷尽测试不可能 6.有效的测试应由第三方独立进行 7.测试无法解释所有缺陷 8.测试的杀虫剂悖论 9.测试是有风险的行为

### 测试用例定义（三要素）

一个文档，详细说明**输入、期望输出**，和为一测试项所准备一组的**执行条件**。

### 软件测试类型



**视角1：测试技术**（代码是否可见的通用划分）

1.1白盒测试:基于软件代码的内部逻辑来设计测试用例。测试推出条件是代码覆盖率。一般由开发人员完成。

1.2黑盒测试：基于需求和功能性，不需要了解内部的设计和代码，通过软件外部表现来发现缺陷和错误。

1.3灰盒测试：既关注软件外部表现，又注意程序内部逻辑结构。

**视角2：开发阶段**

2.1单元测试：对最小的软件设计单元——模块的验证工作

2.2集成测试：验证模块间接口是否正确，多个模块是否能协调一致地正确实现需求和功能。

2.3系统测试：测试整个系统的行为和错误属性

2.4性能测试：评价系统相应时间、吞吐率等系统属性，并与不同版本或竞争产品进行比较

2.5确认测试：验证软件是否可以按照用户合理的期望方式工作

2.6验收测试：保证客户对所有的需求都满意(α测试和β测试)

2.7回归测试：保证增强型或改正型修改使软件正常运行，不影响已有功能

**视角3：执行状态**

3.1静态测试：不运行程序，而通过人工对程序和文档进行分析与检查

3.2动态测试：通过人工或利用工具运行程序进行检查，分析程序执行转台和外部表现

**视角4：执行主体**

4.1开发方测试：（验证测试/α测试）开发方通过检测和提供证据，正式软件商和ifoumanzu规定的需求。

4.2用户测试（β测试）用户通过运行和使用软件，检测与核实软件是信啊是否符合自己预期要求

4.3第三方测试：介于软件开发方和用户之间你的测试组织进行的测试活动

**视角5：特殊测试**

5.1国际化测试：保证全球化软件产品符合不同国家的语言和使用习惯

5.2即兴测试、随机测试、探索性测试：通过直觉和经验，不采用任何形式化的测试。针对有经验的工程师

5.3兼容性测试：确保软件在不同基础设施下都能够一致地发挥作用

5.4安全性测试：测试软件是否存在安全漏洞和隐患

5.5可用性与已获得性测试：确认产品的易用性、美感以及是否方便行动不便的用户

5.6面向对象系统测试：针对采用面向对象技术开发的软件所使用的测试技术

5.7Web测试：发现存在于web应用中的内容、功能、性能、安全性等方面的错误的测试活动集

### 软件测试过程W模型

在V模型的基础上，增加与开发阶段的同步测试，形成W模型；**测试与开发同步进行**，有利用尽早的发现问题

## 软件测试现状和趋势

软件测试逐渐受到重视;存在不同程度的测试人才缺口;高素质的测试人才紧缺.

开发：测试=1：3

# 白盒测试

## 白盒测试基本概念

**白盒测试定义**:一种基于**源程序**或**代码结构与逻辑**，生成测试用例以尽可能多地发现并修改源程序错误的测试方法。白盒测试分为**动态**和**静态**两种类型。

**意义**：主要的单元测试方法，保证软件质量的基础

**实施者**：单元测试阶段：一般由开发人员进行。

集成测试阶段：一般由测试人员和开发人员共同完成

**步骤：动态：**1.程序逻辑分析2.生成测试用例3.执行测试4.分析覆盖标准5.判定测试结果

**静态：测试方法：**桌面检查；代码走查；代码审查

## 静态白盒测试

**静态白盒测试**：在**不执行软件**的条件下有条理地仔细**审查软件设计、体系结构和代码，**从而找出软件缺陷的过程，有时称为**结构化分析。**

**测试条件：**只要求提供软件源代码，不要求提供可执行程序，即**不用在计算机上执行程序**，而**由人阅读代码**。

**静态白盒测试方法：1.桌面检查2.代码检查/走查3.代码评审** (规范化程度，弱🡪强)

**代码审查小组**：1.主持人2.作者3.评论员4.记录员

**代码审查步骤**：1**.计划**（主持人做计划）2.**概述**（描述技术背景）3.**准备**（评论员审查代码 **核对表**）4.**审查会议**（阅读代码讨论提问，记录错误：类型、严重级别，讨论速度，不讨论解决方案）5.**审查报告**（缺陷列表：类型、严重级别）6.**返工**（分配缺陷并修复）7.**跟进**（监督，审查修复部分）

## 动态白盒测试

### 基于控制流的测试方法

#### 1.基于控制流的6种基本测试方法：

**1.语句覆盖:保证程序中每条语句都执行一遍**，但可能会遗漏边。

一、100%的语句覆盖很困难

1)处理错误的代码片段2）小概率事件3)不可达代码（逻辑短路）

二、脆弱的100%语句覆盖

无法发现某些严重问题，原因：遗漏某些不存在语句的边

**2.判定覆盖（分支覆盖）：保证每个判断取真假值至少一次（判定的每个分支至少经过一次）**

**（判定覆盖包含语句覆盖）**

优点：简单，避免了语句覆盖的问题

缺点：忽略了表达式内的条件，不能发现每个条件的错误

**3.条件覆盖：保证每个判断中的每个条件去真假值至少一次**（很弱的覆盖准则）

局限：1.条件覆盖不稳定，无法知道对分支到底覆盖了多少

2.不能保证程序所有分支都被执行，即同时可能遗漏语句和分支

**4.判定条件覆盖：保证每个判断和每个条件取真假值至少一次**（较强）

错误屏蔽现象：100%覆盖，但不能检测出测试用例错误

**5.条件组合覆盖：保证每个判断中的所有条件的取值组合至少出现一次**

缺陷：1.可能会遗漏路径

2.代价昂贵：2n组合数目，n为一个判定中的原子条件数

3.某些条件组合式不可能的

条件组合测试用例的**约简**：利用**短路效应**寻找最小测试用例集，先做短路分析，再做测试用例

**6.路径覆盖：保证每条可能执行到的路径都至少经过一次（如果有环路，则每条环路至少经过一次）**

本质是判定的组合

优点：较为彻底的一种测试

局限：1)路径分支指数级增加2x,x为相互独立的分指数

2）不可达路径存在

3）不能替代涉及**条件测试**的方法

路径覆盖是一种比较强的覆盖标准，但**不能**替代条件覆盖、判定条件覆盖和条件组合覆盖

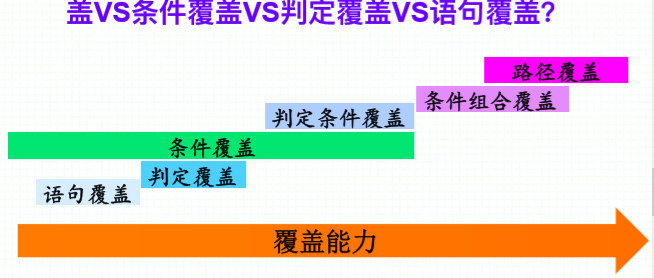
**测试真实的程序的总结：**

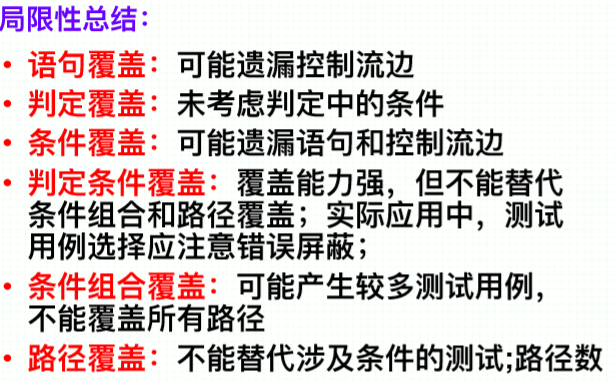
1注意**逻辑短路**

2.**没有被执行**的判断没有被测试，所以**没有取值**

3.没有被执行的条件没有被测试，所以没有取值

4.利用**逻辑短路减少条件组合测试用例数目**

****

****

**MC/DC覆盖（修订的判定条件覆盖）**

**MC/DC覆盖条件**：1.每个判定都取真假至少一次

2.每个条件都取真假至少一次

**3.判定中的每个条件对判断取值是独立的（**条件C对判断D取值独立：存在两个测试用例使得判断D分别取得T\F,除了条件C外，所有条件的取值都未发生改变）

**100%MC/DC覆盖所需测试用例：**设n为条件数，最少：n+1,最大：2n(n+1)

#### 2.基本路径测试方法

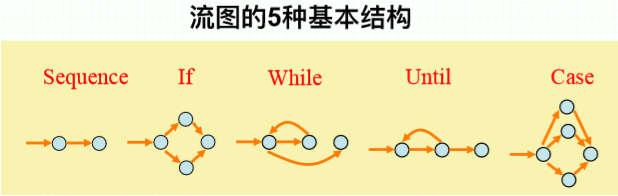
优势：1.更细的控制流描述——程序流图

同时刻画判定和条件对程序控制流的影响；判定条件覆盖

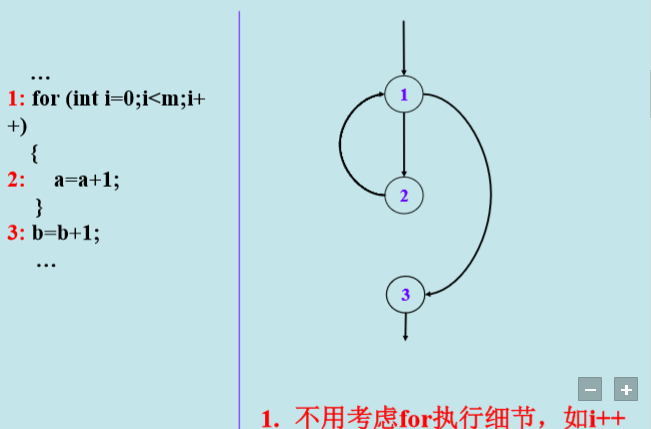
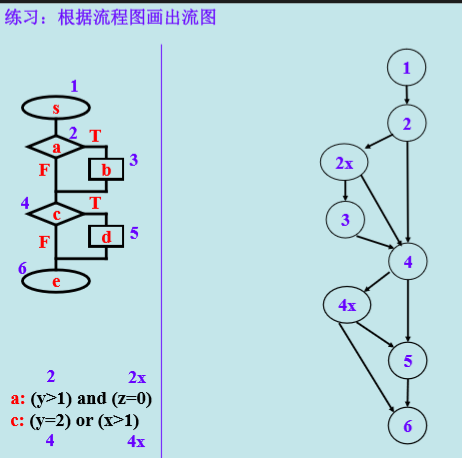
2.更少的覆盖路径——基本路径

近似的路径覆盖；数学和算法理论支撑

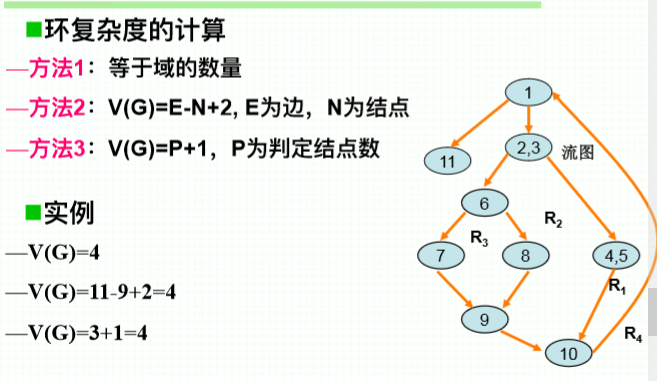
**流图**：用来描述程序中的路基控制流。**节点、边、域**

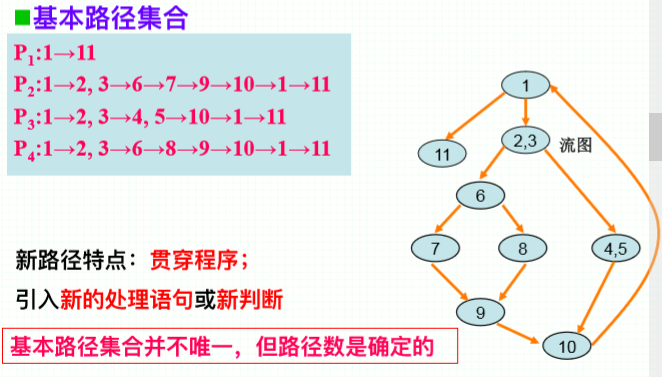


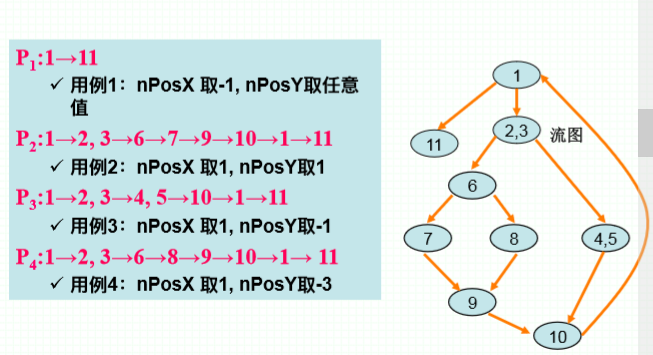


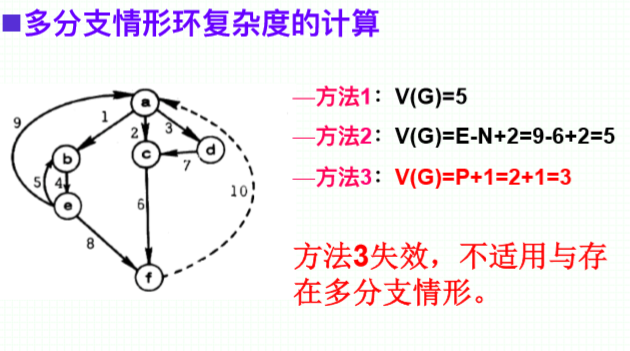


**环复杂度：**度量基本路径数，是所有语句被执行一次所需测试用例数的上限（具有最高环复杂度的模块蕴含错误的可能性最大，是测试中关注的焦点）

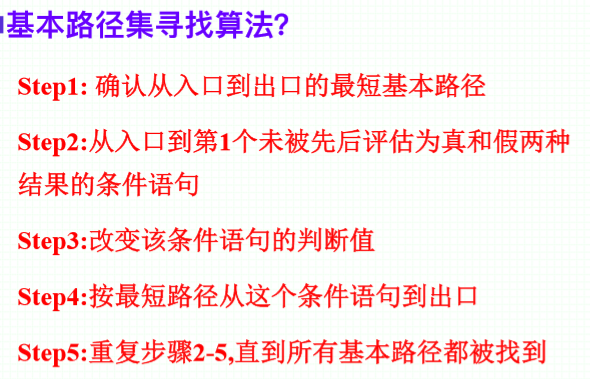




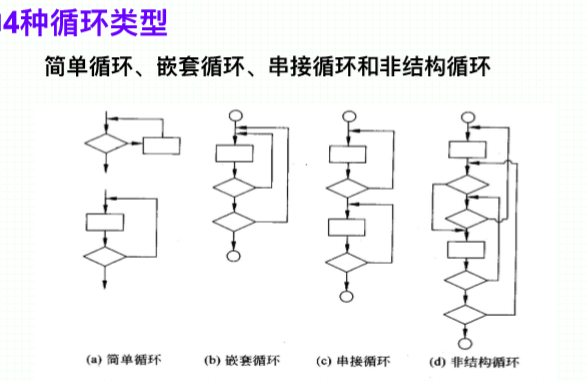


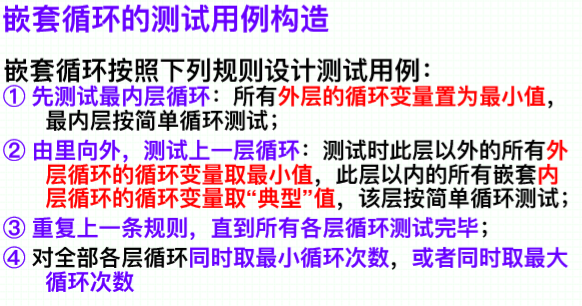
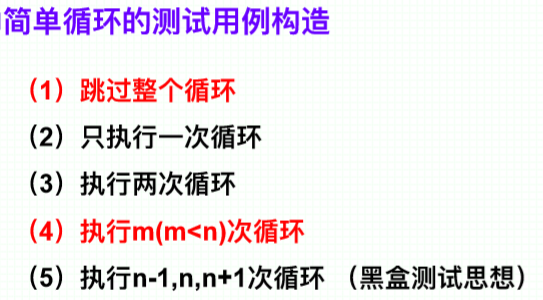


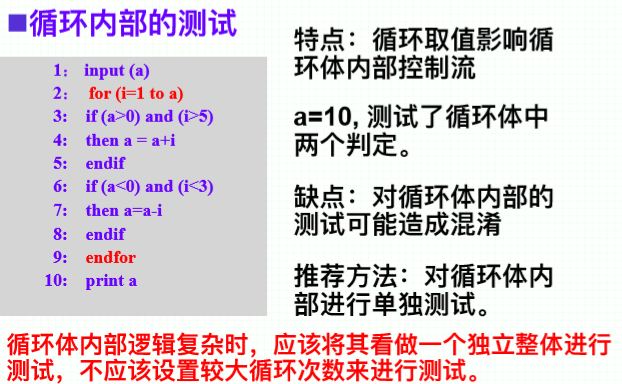
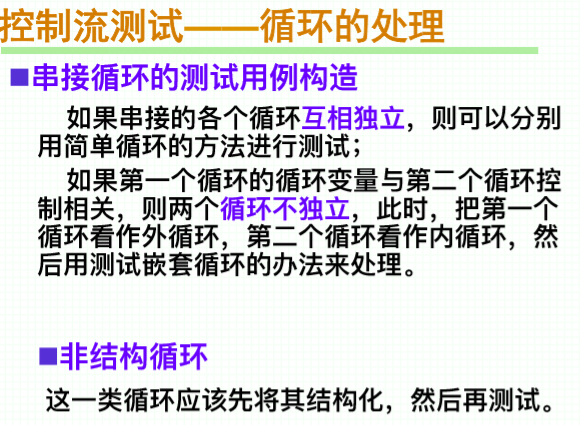
**基本路径的数学定义**：所谓基本路径，是指一组**路径集合**，其他**任何路径对应的向量**，都可以用这些**基本路径对应的向量通过线形运算组合**出来。**基本路径不唯一**

step2:也可以是从出口到入口的第一个

### 基于控制流的测试方法——循环的处理







### 基于数据流的测试方法——数据流覆盖

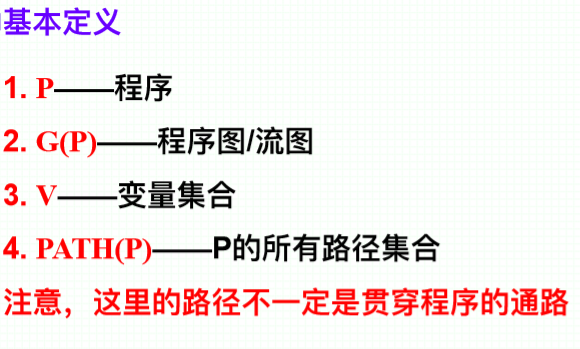
**数据流覆盖思想：**

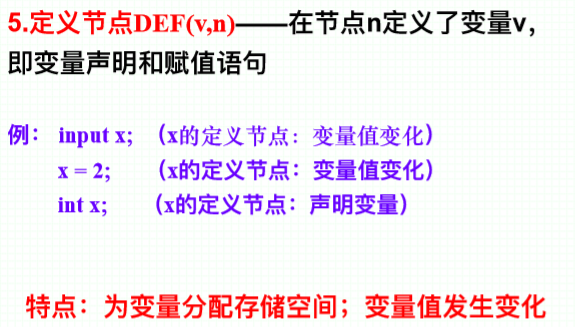
1.程序是对**数据的加工处理**过程，对程序的测试可从数据处理流程的角度进行考虑。

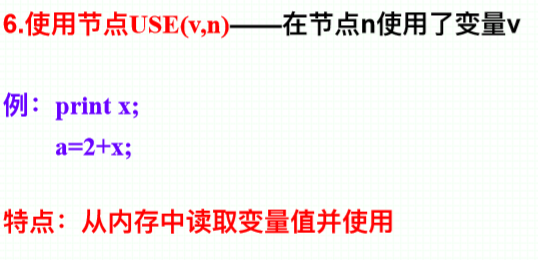
2.寻找关于某变量的**定义**和**使用**位置，思考程序在运行时该变量的值会如何变化。

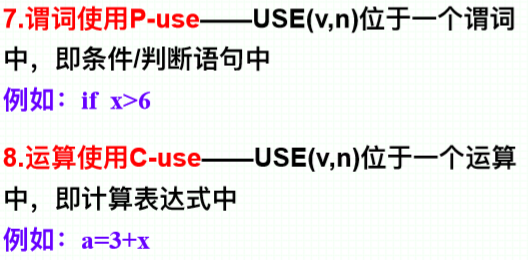
3.数据的处理过程**对应一定的控制流路径**

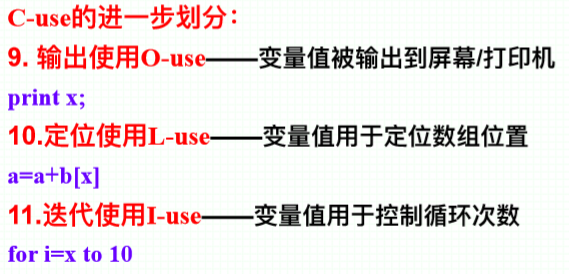
**数据流覆盖原理：考**虑从变量**定义**到其后变量**使用**之间的子路径

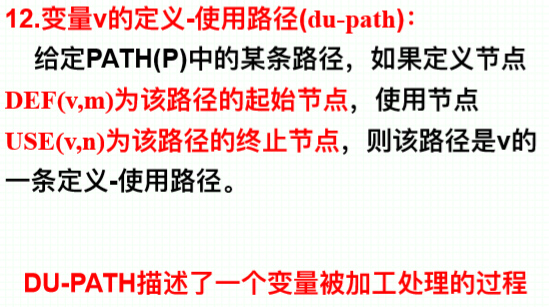


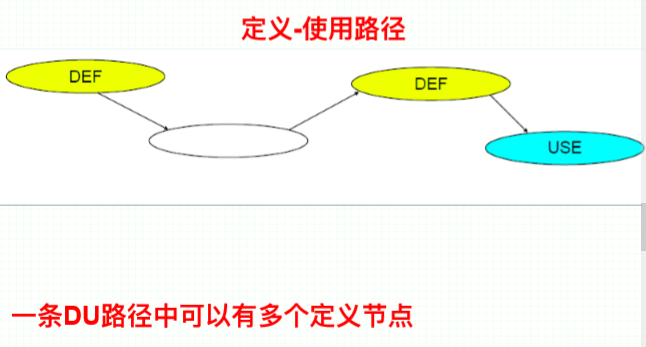


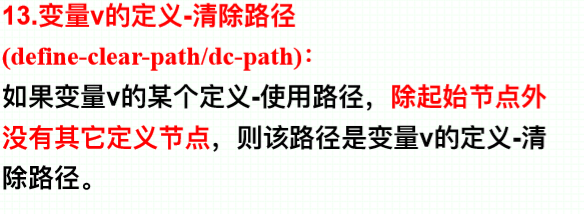


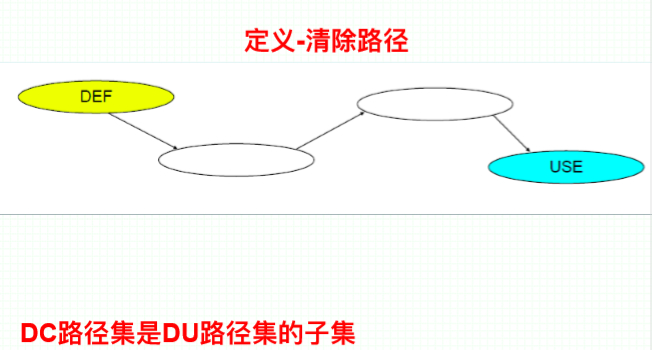












步骤：1.流图

2.分析变量的定义节点和使用节点

3.可能的路径数=use\*def

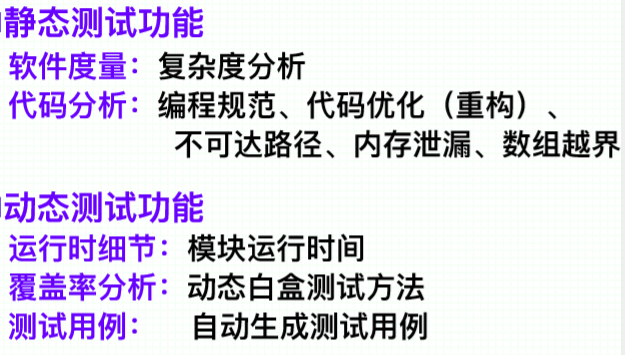
4.列出DU路径

5.约简DU路径

6.设计测试用例

## 白盒测试工具

****



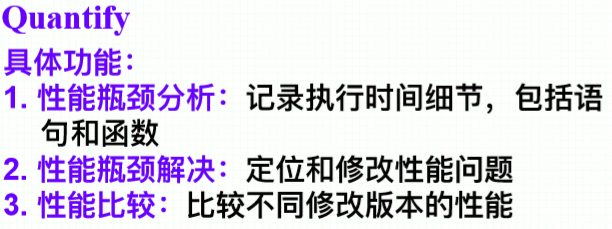
IBM Rational 测试工具：**LogiScope** (静态测试工具)



**Purify净化**

****

**Quantify量化**

****

**PureCoverage纯覆盖**

****

# 黑盒测试

## 黑盒测试基本概念

**黑盒vs白盒**：1.基于规格说明书；2.覆盖尽可能多的上层行为

**什么需要黑盒测试？**1.无法获得源代码；2.尽早黑盒测试的益处——发现软件功能缺陷；3.弥补遗漏的逻辑缺陷；4.适用于各个测试阶段（单元测试、集成测试、系统测试、回归测试）

**随机测试为什么不行？**1.缺陷分布不均匀；2.随机测试很难取到特定值；

**黑盒测试定义**：一种**基于规格说明**，**不要求考察代码**，以**用户视角**进行的测试

**黑盒测试意义**：1.检查**明确需求**和**隐含需求**；2.采用**有效输入**和**无效输入**；3.包含**用户视角**

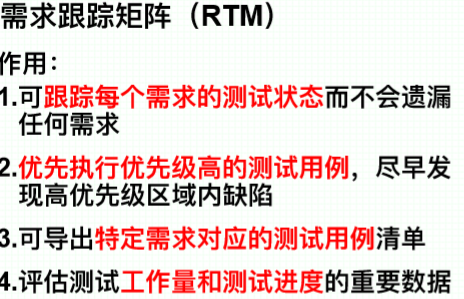
**黑盒测试基础：1.基于需求规格说明书的测试；2.考虑正面用例和负面用例的测试**

### 黑盒测试基础——基于需求的测试

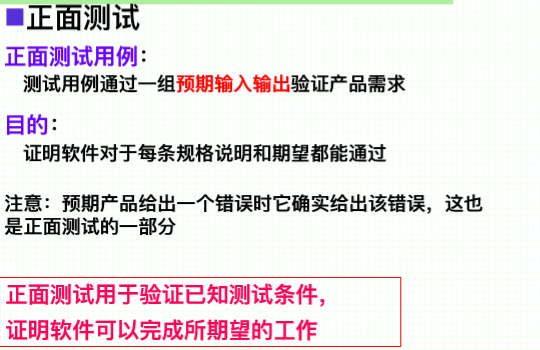
目的：确认软件需求规格说明书列出的需求

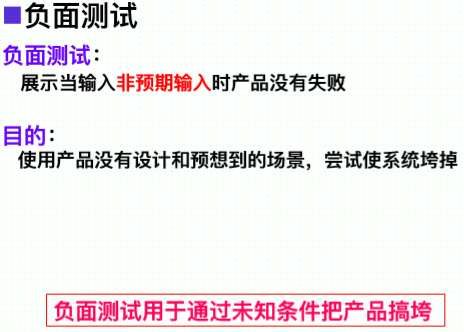
前提：需求规格已经过仔细评审；隐含需求明确化





### 黑盒测试基础——正面和负面测试







## 黑盒测试方法

1.等价类划分

**定义**：将软件**输入**和**输出**的**值域**进行**划分**，选择**少数样本**作为测试用例。

**等价类**：如果**软件行为**对**一组输入/输出值**来说是相同的，则称这组值为**等价类（一个等价划分）**。

**有效等价类**：**有意义**的输入输出数据构成的集合（**满足**规格说明的数据）

**无效等价类：无意义**的输入输出数据构成的集合（**不满足**规格说明的数据）

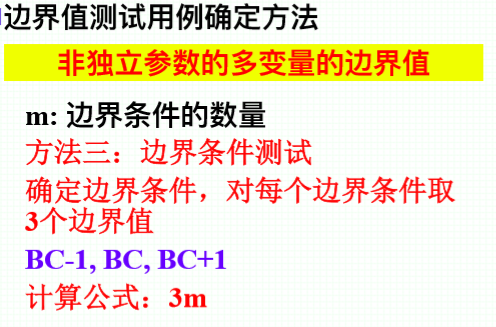
**等价类划分总结**：1.**非冗余性**：尽可能使等价类之间没有交叉；

2.**完整性**：等价类的并集等于输入/输出域

3.**局限**：忽略边界位置的测试用例

2.边界值分析

**边界值分析原理**：软件的两个主要缺陷源（**条件**：变量取值决定控制流走向；**边界**：变量值的”极限”）



**边界值分析vs等价类划分**

1.等价类划分是，往往要先确定边界值

2.边界值分析是等价类划分方法的补充

3.测试中需要将两者结合起来使用

3.因果分析法

4.决策表

5.基于模型的测试

6.正交数组测试