

# 白盒测试技术


中国软件评测中心  
测试中心  
2004年7月






## 教师介绍







## 课程安排




- § 理论讲解
- § 练习
- § 演示







## 白盒测试技术


- § 白盒测试与黑盒测试的差异？
- § 白盒测试包括什么？
- § 如何安排白盒测试？






## 一、白盒测试与黑盒测试的差异

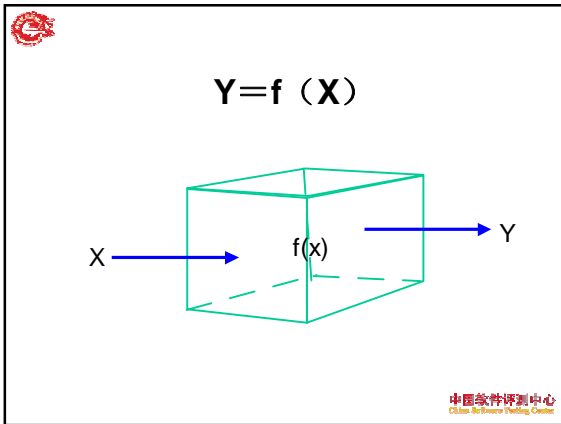




## 何谓白盒测试？

- § 相对黑盒测试而言
- § 也称结构测试或逻辑驱动测试
- § 前提
- § 目标
- § 重点
- § 穷举路径测试

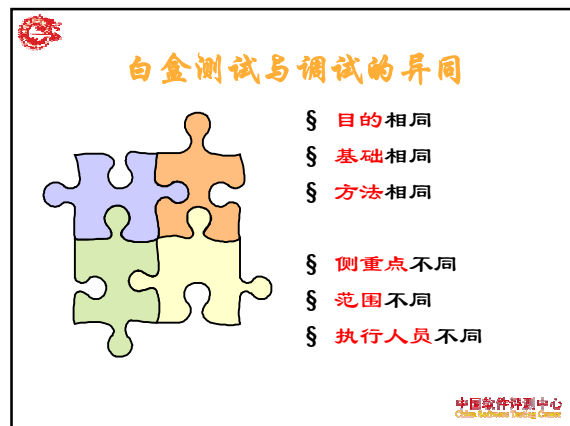
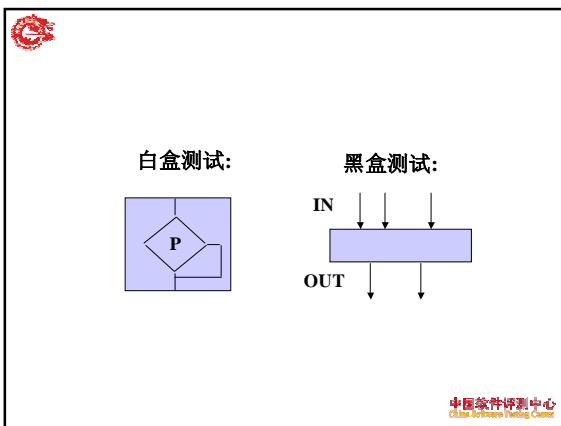




### 黑盒测试VS白盒测试

| 黑盒测试           | 白盒测试           |
|----------------|----------------|
| 不涉及程序结构        | 考查程序逻辑结构       |
| 用软件规格说明生成测试用例  | 用程序结构信息生成测试用例  |
| 可适用于从单元测试到系统联试 | 通常适用于单元测试和集成测试 |
| 某些代码段得不到测试     | 对所有逻辑路径进行测试    |

中国软件评测中心  
China Software Testing Center



## 二、白盒测试包括什么?

中国软件评测中心  
China Software Testing Center




 **白盒测试的方法**

- § 静态测试
- § 动态测试
- § 代码跟踪

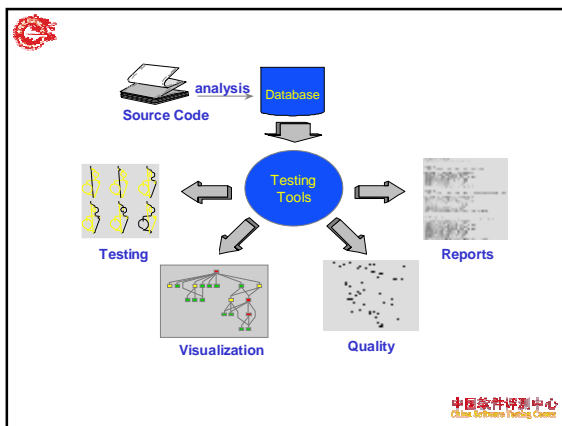


中国软件评测中心  
China Software Testing Center

 **静态测试**


- § 基本概念
- § 类别
  - 静态结构分析
  - 代码质量度量
  - 代码检查
- § 优势

中国软件评测中心  
China Software Testing Center



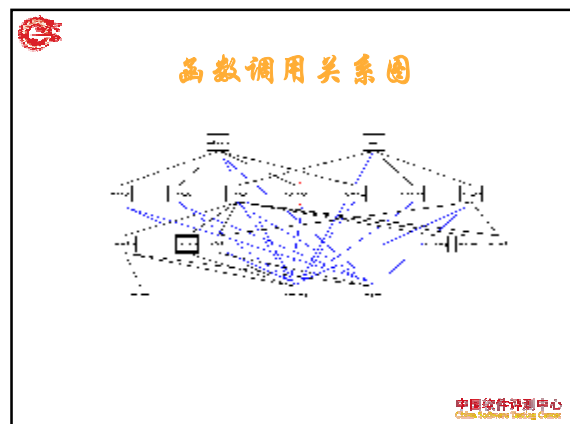
 **1、静态结构分析**

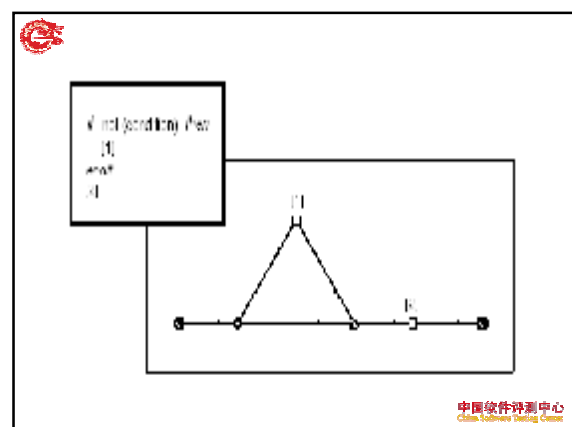
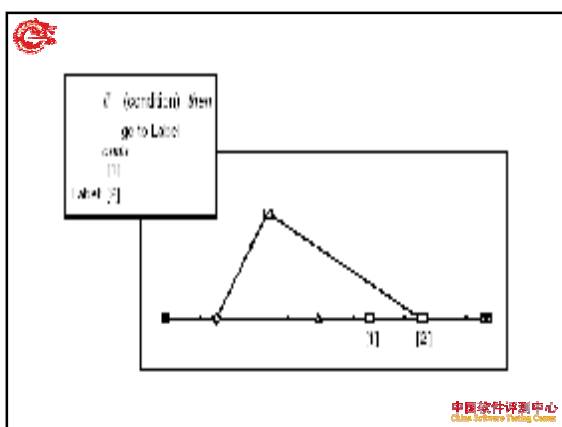
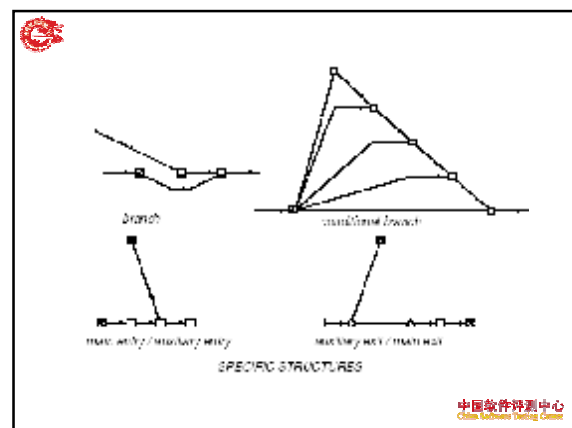
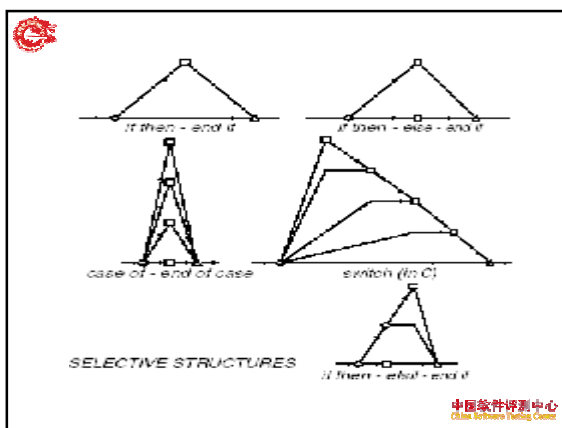
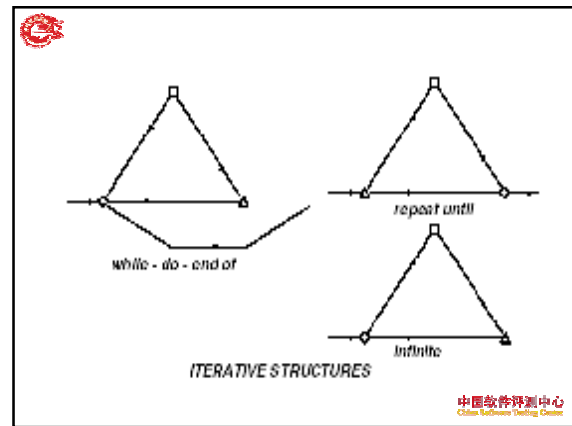
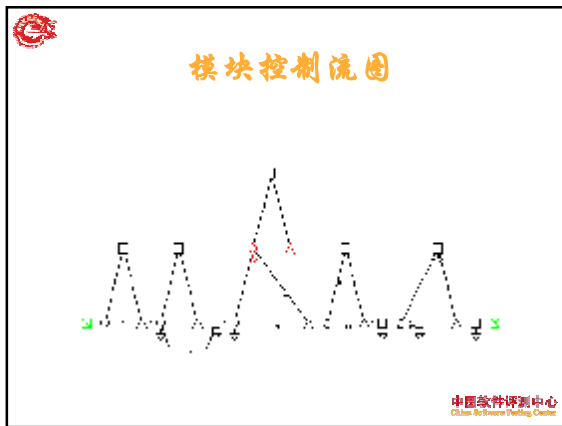
中国软件评测中心  
China Software Testing Center

 **静态结构分析**

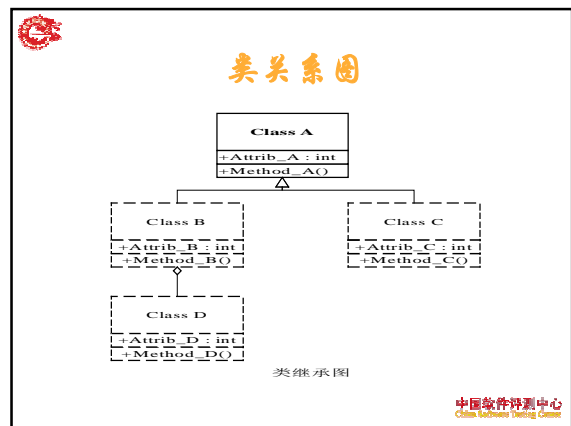
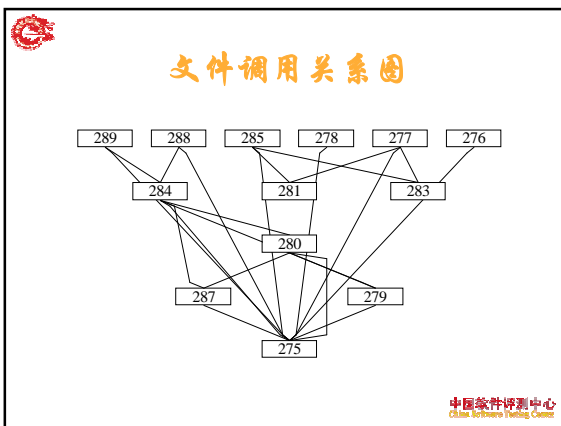
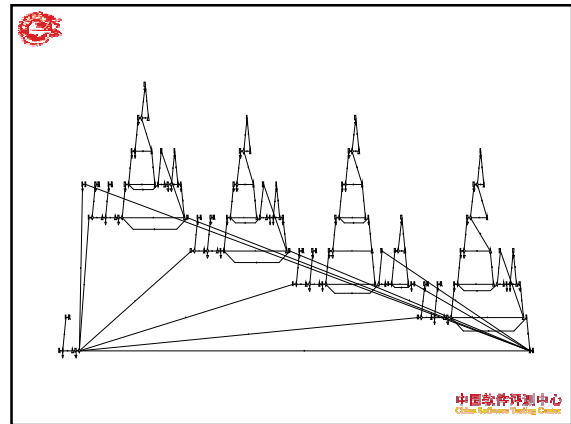
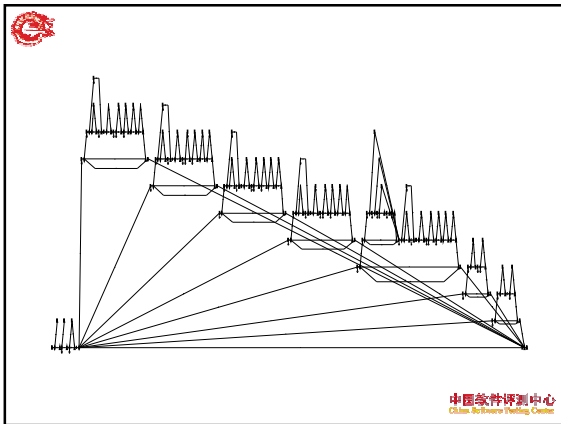
- § 是一种对代码的机械性的、程式化的特性进行分析的方法
- § 静态结构分析常需使用软件工具进行
- § 包括 **控制流分析、数据流分析、接口分析、表达式分析**
  - 系统结构图
  - 函数调用关系图
  - 文件调用关系图
  - 模块控制流图
  - 类关系图

中国软件评测中心  
China Software Testing Center









软件质量度量

- § 遵循 ISO 9126 标准，采取度量统计的方法分析程序的某些质量因素
- § 主要评估可维护性
- § 质量模型的建立

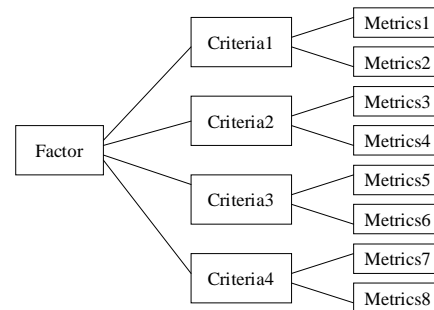
中国软件评测中心  
China Software Testing Center



## 质量模型

- § 质量因素 (Factor)  
依据各分类标准取值组合权重来计算
- § 分类标准 (Criteria)  
由一系列质量规则组成
- § 规则 (Metrics)  
量化的行为规范

中国软件评测中心  
China Software Testing Center



中国软件评测中心  
China Software Testing Center



## Line复杂度

- § 错误率与代码规模有关
- § 统计程序的源代码行数
- § 简单的、粗糙的方法

中国软件评测中心  
China Software Testing Center



## Halstead复杂度

- § Halstead复杂度是以程序中出现的运算符和运算元为计数对象

- a n1: 运算符
- a n2: 运算数
- a N1: 运算符总数
- a N2: 运算数总数

- § 优点
- § 缺点

中国软件评测中心  
China Software Testing Center



## Halstead复杂度(续)

- § Vocabulary: n
- § Observed Length: N
- § Estimated Length: N\*
- § Volume: V
- § Level: L
- § Difficulty: D
- § Mental Effort: E

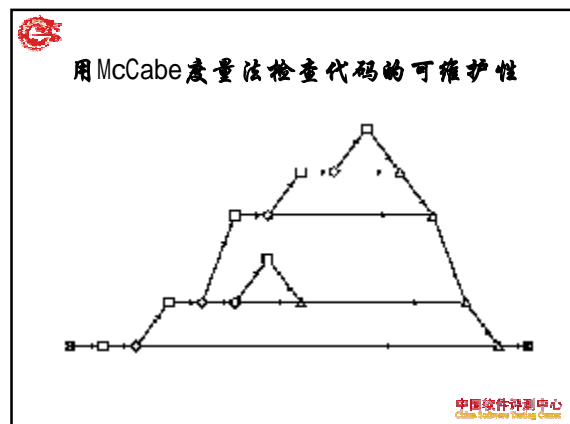
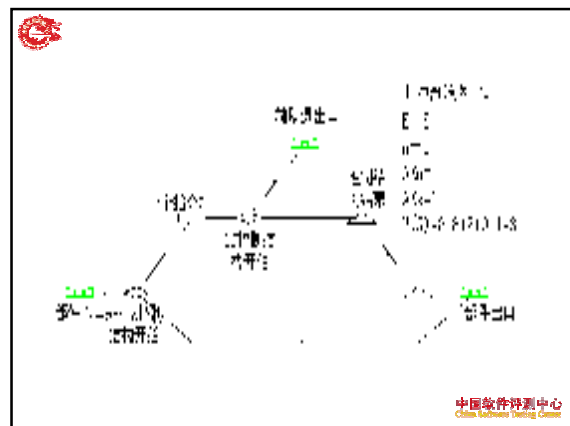
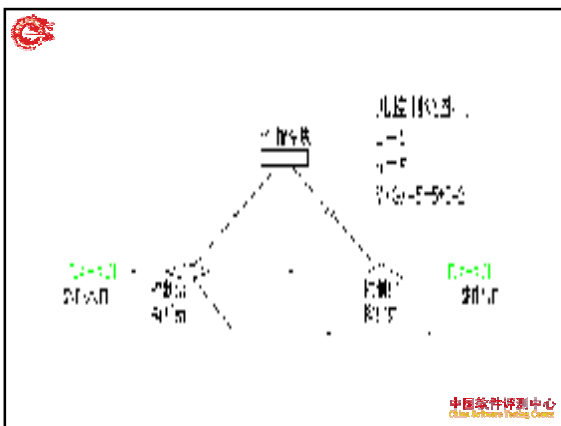
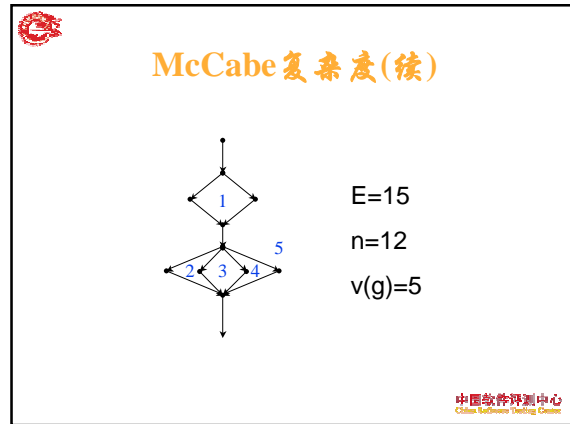
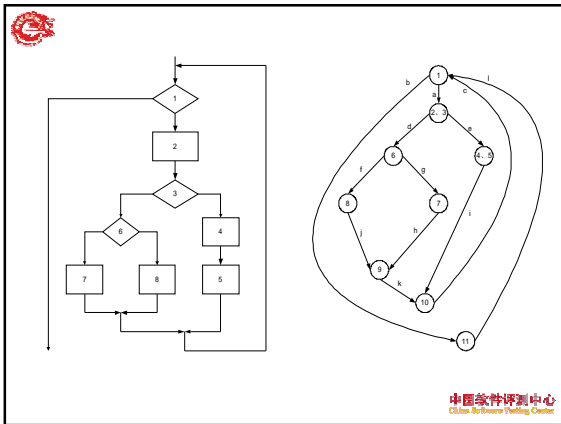
中国软件评测中心  
China Software Testing Center



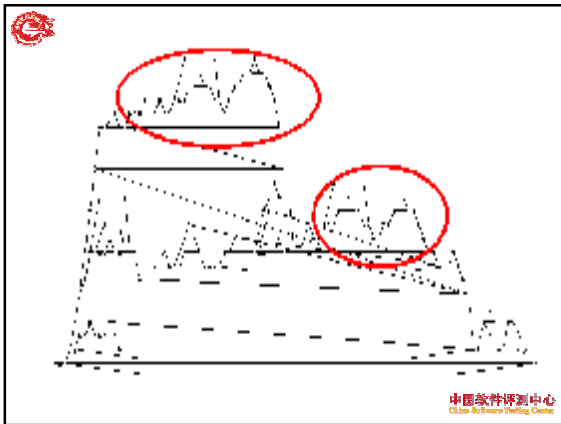
## McCabe复杂度

- § 图复杂度(Cyclomatic complexity) v(g)定义
- § 优点
- § 缺点
- § 图复杂度v(g)计算方法

中国软件评测中心  
China Software Testing Center







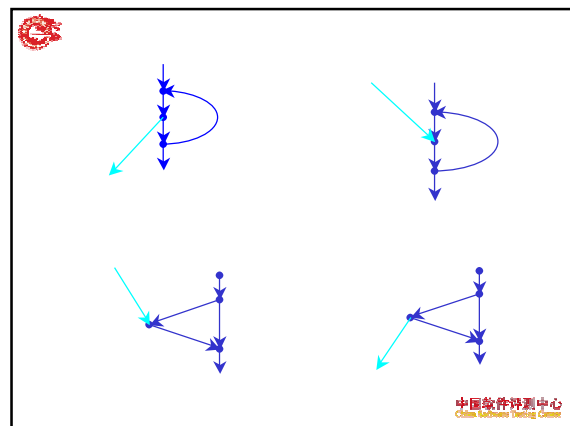
### McCabe复杂度(续)

- § 基本复杂度(Essential Complexity)  $ev(g)$
- § 依照结构化原则简化模块流程后，模块的复杂度，用于衡量程序非结构化程度
- § 优点：

中国软件评测中心  
China Software Testing Center

### 非结构化的逻辑

中国软件评测中心  
China Software Testing Center



### McCabe复杂度(续)

§ 基本复杂度 $ev(g)$ 的计算

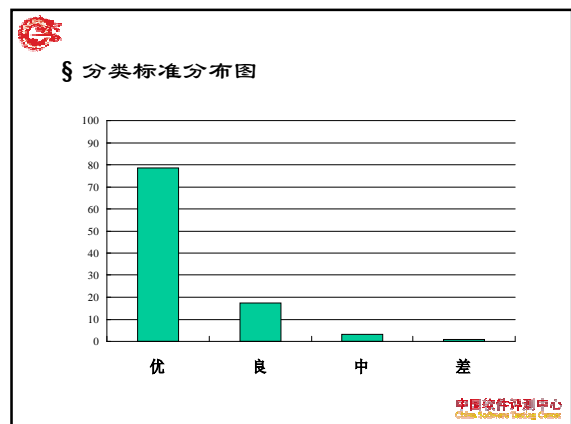
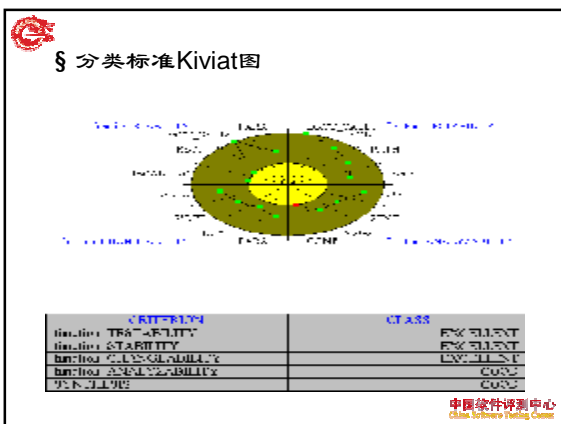
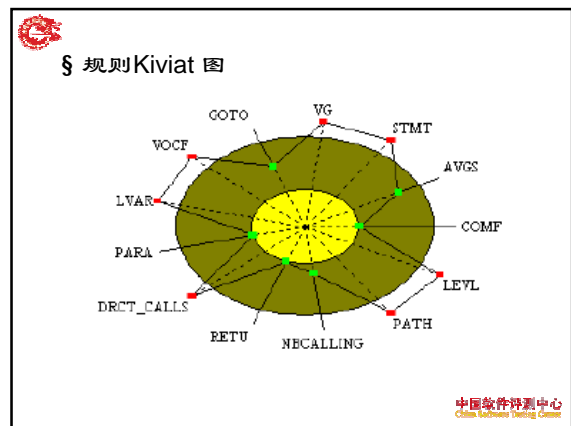
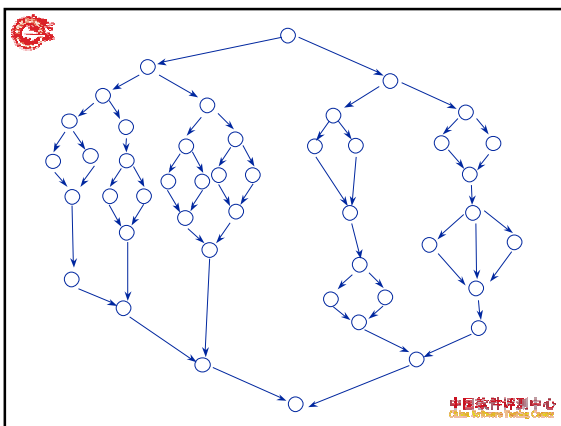
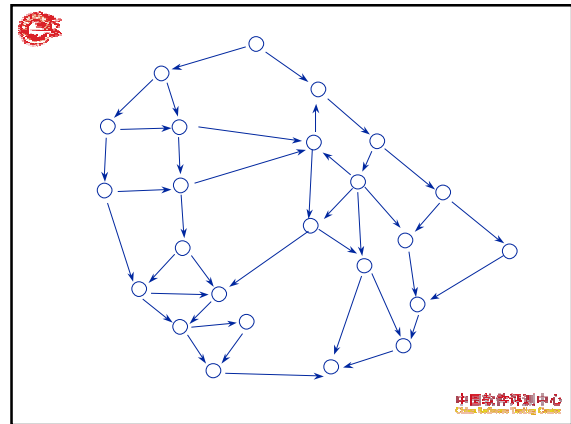
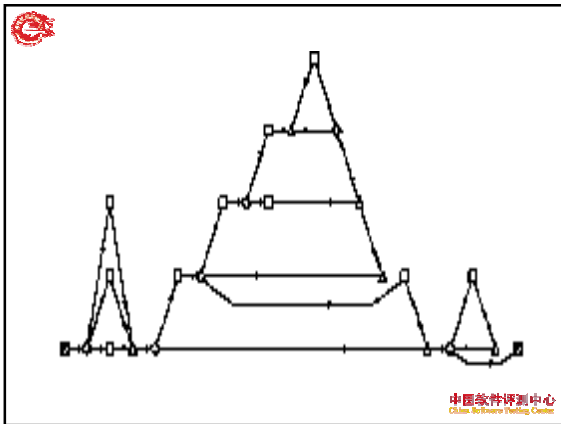
$v(g) = 4$   $ev(g) = 1$

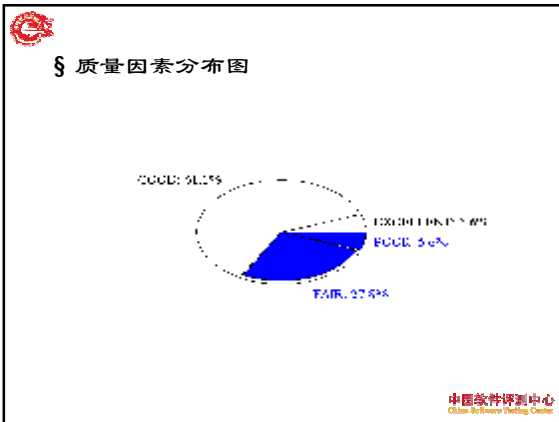
中国软件评测中心  
China Software Testing Center

### McCabe复杂度(续)

- § 模块设计复杂度(Module Design Complexity)  $iv(g)$
- § 设计复杂度(Design Complexity)  $S_0$
- § 集成复杂度(Integration Complexity)  $S_1$
- § 全局数据复杂度(Global Data Complexity)  $gdv(g)$
- § 局部数据复杂度(Specified Date Complexity)  $sdv(g)$
- § 病态数据复杂度(Pathological Complexity)  $pv(g)$

中国软件评测中心  
China Software Testing Center





main( )

```
{
  int a[10], i, j, tmp;
  for(i=0; i <= 10; i++)
    scanf("%d", &a[i]);
  for(i=0; i <= 10; i++)
    for(j=i+1; j <= 10; j++)
      if ( a[i] < a[j]) {
        a[i] = a[j];
      }
  for(i=0; i <= 10; i++)
    printf("%d ", a[i]);
}
```

中国软件评测中心  
China Software Testing Center

### 3、代码检查

中国软件评测中心  
China Software Testing Center

### 代码检查

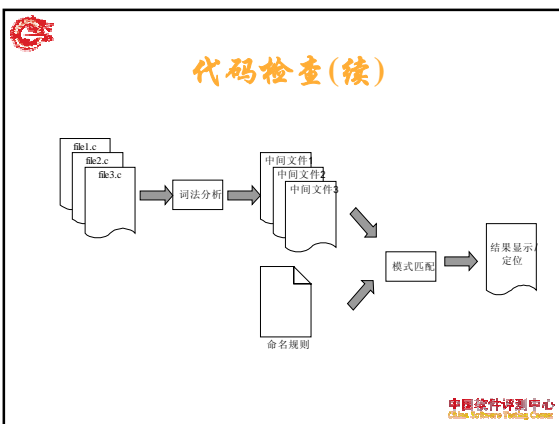
§ 目的

- 排除违背程序编写标准的问题
- 排除违背程序编程风格的问题
- 找出程序中不可移植部分
- 发现程序中不安全、不明确和模糊的部分
- 确保代码和设计的一致性
- 确保代码的逻辑表达的正确性
- 确保代码结构的合理性

§ 方式

- 桌面检查
- 人工走查
- 代码审查

中国软件评测中心  
China Software Testing Center



### 代码检查(续)

§ 确保代码编程标准有效的被执行

§ 提高代码质量，减轻动态测试负担

§ 代码可重复使用，降低项目风险与经费

§ 增加程序的可理解性，降低维护成本

§ 效率高：实践表明，人工走查平均能查出被测程序的30%~70 %的逻辑设计和编码缺陷，IBM代码审查会的查错效率高达80%

中国软件评测中心  
China Software Testing Center

 **代码检查(续)**

- § 需求描述文档
- § 程序设计文档
- § 程序的源代码清单
- § 代码编码标准
- § 代码缺陷检查表
- § ...




中国软件评测中心  
China Software Testing Center

 **代码检查(续)**

- § 变量命名和类型审查
- § 变量初始值检查
- § 变量作用范围检查
- § 程序逻辑审查
- § 程序语法检查
- § 程序结构检查




中国软件评测中心  
China Software Testing Center

 **代码编码标准**


- § ISO C++ standard (ISO/IEC 14882:1998(E))
- § Motor Industry Software Reliability Association制订的MISRA标准
- § 参考软件工程专家的建议，如：Scott Meyers和Martin Klaus
  - “Effective C++”
  - “More Effective C++”
  - “Examining C++ Program Analyzers”

中国软件评测中心  
China Software Testing Center

 **代码缺陷检查表**


- § 格式
- § 入口和出口的连接
- § 程序语言的使用
- § 存储器使用
- § 测试和转移
- § 性能
- § 可维护性
- § 逻辑
- § 可靠性

中国软件评测中心  
China Software Testing Center

 **代码示例 1**


```
int main()
{
    float a = 1.0;
    float b = 2.0;
    ...
    if ((a * 2 - b) == 0.0)
    {
        return 1;
    }
    return 0;
}
```

中国软件评测中心  
China Software Testing Center

 **代码示例 2**

```
void main()
{
    int a = 0, b = 0;
    ...
    switch (a)
    {
        case 0:
            b = -1;
        case 1:
            b = b / a;
            break;
    }
    ...
}
```

中国软件评测中心  
China Software Testing Center



```


int main()
{
    int sum, i;

    for (i = 1; i < 30000; i++)
    {
        sum = sum + i;
    }

    return 0;
}

```

中国软件评测中心  
China Software Testing Center




```

void func()
{
    float a = 72.0;
    float b = 55.0;
    long result = 0;

    result = (long)(a*b);
    return 1;
}

```

中国软件评测中心  
China Software Testing Center



```


int func()
{
    char c1 = 0;
    c1 = 145;

    if (c1 > 300)
    {
        return 3;
    }

    if (c1 > 25)
    {
        return 2;
    }
    return 1;
}

```

中国软件评测中心  
China Software Testing Center




```

void func()
{
    int i = 0;
    ...
    for (i = 0; i < 10; i++)
    {
        if (i == 5)
        {
            break;
        }
    }
    ...
    return;
}

```

中国软件评测中心  
China Software Testing Center




```

int foo103()
{
    int *pi = 0;
    char *pc = 0;

    if (pc == pi)
    {
        return 1;
    }
    else
    {
        return 0;
    }
}

```

中国软件评测中心  
China Software Testing Center




```

int main()
{
    char *c = "TEST";
    char d[20];
    ...
    memcpy(d, c, sizeof (d));
    ...
}

```

中国软件评测中心  
China Software Testing Center




```
int main()
{
    ...
    char b[8];
    strncpy(b, "This is a test", sizeof (b));
    printf("%s\n", b);
    ...
}
```

中国软件评测中心  
China Software Testing Center



## 动态测试


中国软件评测中心  
China Software Testing Center



### 动态测试特点

- § 必须生成测试数据来运行被测试程序，取得程序运行的真实情况、动态情况，进而进行分析
- § 测试质量依赖于测试数据
- § 生成测试数据、分析测试结果的工作量大，使开展测试工作费时、费力、费人
- § 动态测试中涉及多方面工作，人员多、设备多、数据多，要求有较好的管理和工作规程


中国软件评测中心  
China Software Testing Center



### 测试内容


- § 功能确认与接口测试
- § 逻辑覆盖率分析
- § 性能与效率分析
- § 内存分析

中国软件评测中心  
China Software Testing Center



## 4、功能确认与接口测试

中国软件评测中心  
China Software Testing Center



### 功能确认与接口测试要求

- § 根据软件概要设计说明书和详细设计说明书；
- § 验证程序和详细设计说明的一致性；
- § 检验每个软件单元模块能否正确的实现其功能，满足其性能和接口要求；
- § 单元测试中测试每个单元模块，集成测试中进一步把单元组装成部件并测试其正确性；

中国软件评测中心  
China Software Testing Center



## 功能确认与接口测试范围

- § 单元接口
- § 局部数据结构
- § 重要的执行路径
- § 错误处理的路径
- § 影响上述几点的边界条件

中国软件评测中心  
China Software Testing Center



## 单元接口测试

- § 输入参数的**数目**是否与模块变元数目相同？
- § 参数**属性**与变元属性是否一致？
- § 参数与变元的**单位**是否一致？
- § 传到被调用模块的变元的**数目**是否与参数的数目相同？
- § 传到被调用模块的变元的**属性**是否与参数的属性相同？
- § 传到被调用模块的变元的**单位**是否与参数的单位相同？

中国软件评测中心  
China Software Testing Center



## 单元接口测试

- § 引用内部函数时，变元次序属性和数目是否正确？
- § 是否引用了与当前入口无关的参数？
- § 公用于输入的变量有没有改变？
- § 在经过不同模块时，全局变量的定义是否一致？
- § 限制条件是否以参量的形式传递？

中国软件评测中心  
China Software Testing Center



## 扩展接口测试

- § 文件属性正确吗？
- § open语句正确吗？
- § 规定的格式是否与I/O语句相符？
- § 缓冲区的大小与记录的大小相匹配吗？
- § 使用文件前文件打开了吗？
- § 文件结束的条件安排了吗？
- § I/O错误能处理吗？
- § 在输出信息中有文字错误吗？

中国软件评测中心  
China Software Testing Center



## 局部数据结构测试

- § 不正确的或不一致的**数据说明**；
- § **初始化**有错或没有赋值；
- § 不正确的**变量名**（拼错或缩写了）；
- § 不一致的**数据类型**；
- § **上溢**、**下溢**或引用错。

中国软件评测中心  
China Software Testing Center



## 重要的执行路径

- § 指那些处在完成单元功能的算法、控制、数据处理等重要部位的执行路径，也指由于控制较复杂而易错的路径。
- 常见的错误有：
- § 误解或错误处理算术运算的优先次序；
- § 混用不同类的操作；
- § 初始化不正确；
- § 计算精度不够；
- § 表达式的符号表示不正确。

中国软件评测中心  
China Software Testing Center



## 比较操作的错误

- § 不同的数据类型比较；
- § 不正确的逻辑操作符或不正确的优先次序；
- § 本应相等，但因精度不够使之不等，如浮点数相等比较；
- § 不正确地比较变量，如使用错误的变量进行比较；
- § 不正常的或不存在的循环终止条件；
- § 当遇到发散的循环时无法跳出循环；
- § 错误地修改循环变量。

中国软件评测中心  
China Software Testing Center



## 错误处理路径

- § 错误的描述不易理解；
- § 指出的错误并不是所遇到的错误；
- § 出错时，还没有进行出错处理就先进行系统干预；
- § 错误边界条件的处理不正确；
- § 描述错误的信息不正确，不足以确定出错的原因；
- § 错误处理方式是否符合要求，如是否应重置或恢复；
- § 错误处理是否被适当地激发。

中国软件评测中心  
China Software Testing Center



```
int add( int a, int b )
{
    int result;

    result = a + b;

    return (result);
}
```

中国软件评测中心  
China Software Testing Center



```
int division( int a, int b )
{
    int result;

    result = a / b;

    return (result);
}
```

中国软件评测中心  
China Software Testing Center



```
#include <math.h>
#include <stdio.h>

void main()
{
    int a, b, result;

    scanf( "%d %d", &a, &b );

    result = add( a, b );
    printf( "a+b= %d \n", result );

    result = division( a, b );
    printf( "a/b= %d \n", result );

    return;
}
```

中国软件评测中心  
China Software Testing Center



## 5、逻辑覆盖率分析

中国软件评测中心  
China Software Testing Center





## 逻辑覆盖率分析

- § 依据被测程序的逻辑结构设计测试用例，驱动被测程序运行完成测试。
- § 语句覆盖 Statement Coverage
- § 判定覆盖 Decision Coverage
- § 条件覆盖 Condition Coverage
- § 分支条件组合覆盖 Condition/Decision Coverage
- § 多条件覆盖 Multiple Condition Coverage
- § 修正条件/判定覆盖 MC/DC
- § 路径覆盖 Path Coverage

中国软件评测中心  
China Software Testing Center



## 语句覆盖

- § Statement Coverage
- § C1覆盖、行覆盖 (line coverage)、段覆盖 (segment coverage) 和基本块覆盖 (basic block coverage)
- § 每一个可执行语句至少执行一次
- § 主要缺点是对一些控制结构很迟钝

中国软件评测中心  
China Software Testing Center



## 语句覆盖示例

```
if ( A && ( B || C ) )    x=1;
else                    x=0;
```

|                 | 用例1  | 用例2 |
|-----------------|------|-----|
| A               | T    | F   |
| B               | T    | F   |
| C               | T    | F   |
| A && ( B    C ) | T    | F   |
| 语句覆盖率           | 100% |     |

中国软件评测中心  
China Software Testing Center



## 语句覆盖示例-2

|                 | 用例1  | 用例2 |
|-----------------|------|-----|
| A               | T    | F   |
| B               | T    | F   |
| C               | T    | F   |
| A && ( B    C ) | T    | F   |
| A    ( B    C ) | T    | F   |
| 语句覆盖率           | 100% |     |

中国软件评测中心  
China Software Testing Center



## 判定覆盖

- § Decision Coverage
- § C2覆盖、分支覆盖 (branch coverage)、所有边界覆盖 (all-edges coverage)、基本路径覆盖 (basis path coverage)、判定到判定路径覆盖 (decision-decision-path或DDP testing)
- § 程序中每个判定至少都获得一次“真”值和“假”值 (例如if-statement和while-statement)


中国软件评测中心  
China Software Testing Center



## 判定覆盖(续)

- § 这个度量包括IF、For、SWITCH、exception handlers、interrupt handlers等的覆盖
- § 缺点：这个度量忽略了BOOL型表达式内部的BOOL取值。整个BOOL型表达式被认为是取值一个TRUE和FALSE，而不考虑是否内部包含了logical-and 或 logical-or 操作符

中国软件评测中心  
China Software Testing Center




### 判定覆盖示例

$A \&\& (B || C)$

|                   | 1   | 2 | 3 | 4 | 5   | 6 | 7 | 8 |
|-------------------|-----|---|---|---|-----|---|---|---|
| A                 | F   | F | F | F | T   | T | T | T |
| B                 | F   | F | T | T | F   | F | T | T |
| C                 | F   | T | F | T | F   | T | F | T |
| $A \&\& (B    C)$ | F   | F | F | F | F   | T | T | T |
| 判定覆盖率             | 50% |   |   |   | 50% |   |   |   |


中国软件评测中心  
China Software Testing Center



### 判定覆盖示例-2

|                     | 1   | 2 | 3 | 4 | 5   | 6 | 7 | 8 |
|---------------------|-----|---|---|---|-----|---|---|---|
| A                   | F   | F | F | F | T   | T | T | T |
| B                   | F   | F | T | T | F   | F | T | T |
| C                   | F   | T | F | T | F   | T | F | T |
| $A \&\& (B    C)$   | F   | F | F | F | F   | T | T | T |
| $A \&\& (B \&\& C)$ | F   | F | F | F | F   | F | F | T |
| 判定覆盖率               | 50% |   |   |   | 50% |   |   |   |

中国软件评测中心  
China Software Testing Center



### 条件覆盖


§ Condition Coverage

§ 每个判定中的每个条件的可能值至少满足一次

§ logical-and 和 logical-or 独立起来。条件覆盖独立的度量每一个子表达式。这个度量和 decision coverage 相似，但是对控制流更敏感

§ 但是，完全的条件覆盖并不能保证完全的判定覆盖

中国软件评测中心  
China Software Testing Center




### 条件覆盖示例

$A \&\& (B || C)$

|                   | 1    | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|-------------------|------|---|---|---|---|---|---|---|
| A                 | F    | F | F | F | T | T | T | T |
| B                 | F    | F | T | T | F | F | T | T |
| C                 | F    | T | F | T | F | T | F | T |
| $A \&\& (B    C)$ | F    | F | F | F | F | T | T | T |
| 条件覆盖率             | 100% |   |   |   |   |   |   |   |
| 判定覆盖率             | 50%  |   |   |   |   |   |   |   |

中国软件评测中心  
China Software Testing Center



### 分支条件组合覆盖

§ Condition/Decision Coverage


§ 是条件覆盖 (condition coverage) 和分支覆盖 (decision coverage) 的一个混血

§ 判定中每个条件的所有可能至少出现一次，并且每个判定本身的判定结果也至少出现一次

§ 它有两者的简单性但是没有两者的缺点

§ 但是，没有考虑单个判定对整体结果的影响

中国软件评测中心  
China Software Testing Center



### 分支条件组合覆盖示例

$A \&\& (B || C)$

|                     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---------------------|---|---|---|---|---|---|---|---|
| A                   | F | F | F | F | T | T | T | T |
| B                   | F | F | T | T | F | F | T | T |
| C                   | F | T | F | T | F | T | F | T |
| $A \&\& (B    C)$   | F | F | F | F | F | T | T | T |
| $A \&\& (B \&\& C)$ | F | F | F | F | F | F | F | T |

中国软件评测中心  
China Software Testing Center



## 多条件覆盖

### § Multiple Condition Coverage

- § 判定条件中条件的每一个可能组合至少出现一次
- § 相对于条件覆盖，即通过logical-and和logical-or把子表达式独立起来相比，多条件覆盖需要的测试用例是用一个条件的逻辑操作符的真值表来确定的

中国软件评测中心  
China Software Testing Center



## 多条件覆盖(续)

- § 对于C, C++和Java等具有short circuit operators的语言，多条件覆盖的益处是它进行一个彻底的测试
- § 缺点是它可能是非常冗长乏味的来决定一个需要的测试用例的最小设置，尤其是对于非常复杂的BOOL型表达式
- § 可能会有路径遗漏

中国软件评测中心  
China Software Testing Center



## 多条件覆盖示例

A && (B || C)

|           | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|-----------|---|---|---|---|---|---|---|---|
| A         | F | F | F | F | T | T | T | T |
| B         | F | F | T | T | F | F | T | T |
| C         | F | T | F | T | F | T | F | T |
| A&&(B  C) | F | F | F | F | F | T | T | T |

中国软件评测中心  
China Software Testing Center



## 多条件覆盖示例(续)

```
if ((a > 1) && (b == 0))
{
    x = x / a;
}
if ((a == 2) || (x > 1))
{
    x = x + 1;
}
```

中国软件评测中心  
China Software Testing Center



## 多条件覆盖示例(续)

1. A>1,B=0 T1,T2
2. A>1,B!=0 T1,-T2
3. A<=1,B=0 -T1,T2
4. A<=1,B!=0 -T1,-T2
5. A=2,X>1 T3,T4
6. A=2,X<=1 T3,-T4
7. A!=2,X>1 -T3,T4
8. A!=2,X<=1 -T3,-T4

中国软件评测中心  
China Software Testing Center



## 多条件覆盖示例(续)

| 用例 | ABX   | 覆盖组合 | 覆盖条件            | 执行路径 |
|----|-------|------|-----------------|------|
| 1  | 2 0 3 | 1, 5 | T1,T2,T3,T4     | T T  |
| 2  | 2 1 1 | 2, 6 | T1,-T2,T3,-T4   | F T  |
| 3  | 1 1 1 | 4, 8 | -T1,-T2,-T3,-T4 | F F  |
| 4  | 1 0 3 | 3, 7 | -T1,T2,-T3,T4   | F T  |

中国软件评测中心  
China Software Testing Center



## 修正条件/判定覆盖

- § Modified Condition/Decision Coverage
- § 也被称为MC/DC 和MCDC。
- § 这个度量需要足够的测试用例来确定每个条件能够影响到包含的判定的结果。
- § “航空运输和装备系统软件认证标准”
- § 由欧美的航空/航天制造厂商和使用单位联合制定（波音公司等）、RTCA颁布
- § 目前在国外的国防、航空航天领域广泛应用

中国软件评测中心  
China Software Testing Center



## 修正条件/判定覆盖(续)

- § 每一个程序模块的入口和出口点都要考虑要至少被调用一次，每个程序的判定到所有可能的结果值要至少转换一次
- § 程序的判定被分解为通过逻辑操作符(AND, OR, etc.)连接为BOOL条件。每一个条件对于判定的结果值是独立的，或者说单条件的变化将导致判决的变化

中国软件评测中心  
China Software Testing Center



## 修正条件/判定覆盖示例

A&&B

|      | 用例1 | 用例2 | 用例3 | 用例4 |
|------|-----|-----|-----|-----|
| A    | T   | T   | F   | F   |
| B    | T   | F   | T   | F   |
| A&&B | T   | F   | F   | F   |

中国软件评测中心  
China Software Testing Center



## 修正条件/判定覆盖示例(续)

A&&B

|      | 用例1 | 用例2 | 用例3 | 用例4 |
|------|-----|-----|-----|-----|
| A    | T   | T   | F   | F   |
| B    | T   | F   | T   | F   |
| A&&B | T   | F   | F   | F   |

中国软件评测中心  
China Software Testing Center



## 修正条件/判定覆盖示例(续)

A&&B

|      | 用例1 | 用例2 | 用例3 | 用例4 |
|------|-----|-----|-----|-----|
| A    | T   | T   | F   | F   |
| B    | T   | F   | T   | F   |
| A&&B | T   | F   | F   | F   |

中国软件评测中心  
China Software Testing Center



## 修正条件/判定覆盖示例(续)

A&&B

|      | 用例1 | 用例2 | 用例3 | 用例4 |
|------|-----|-----|-----|-----|
| A    | T   | T   | F   | F   |
| B    | T   | F   | T   | F   |
| A&&B | T   | F   | F   | F   |

中国软件评测中心  
China Software Testing Center



### 修正条件/判定覆盖示例(续)

A && B

|        | 用例1 | 用例2 | 用例3 | 用例4 |
|--------|-----|-----|-----|-----|
| A      | T   | T   | F   | F   |
| B      | T   | F   | T   | F   |
| A && B | T   | F   | F   | F   |

中国软件评测中心  
China Software Testing Center



### 修正条件/判定覆盖示例(续)

A && (B || C)

|               | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---------------|---|---|---|---|---|---|---|---|
| A             | T | T | T | T | F | F | F | F |
| B             | T | T | F | F | T | T | F | F |
| C             | T | F | T | F | T | F | T | F |
| A && (B    C) | T | T | T | F | F | F | F | F |

中国软件评测中心  
China Software Testing Center



### 修正条件/判定覆盖示例(续)

A && (B || C)

|               | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---------------|---|---|---|---|---|---|---|---|
| A             | T | T | T | T | F | F | F | F |
| B             | T | T | F | F | T | T | F | F |
| C             | T | F | T | F | T | F | T | F |
| A && (B    C) | T | T | T | F | F | F | F | F |

中国软件评测中心  
China Software Testing Center



### 修正条件/判定覆盖示例(续)

A && (B || C)

|               | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---------------|---|---|---|---|---|---|---|---|
| A             | T | T | T | T | F | F | F | F |
| B             | T | T | F | F | T | T | F | F |
| C             | T | F | T | F | T | F | T | F |
| A && (B    C) | T | T | T | F | F | F | F | F |

中国软件评测中心  
China Software Testing Center



### 修正条件/判定覆盖示例(续)

A && (B || C)

|               | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---------------|---|---|---|---|---|---|---|---|
| A             | T | T | T | T | F | F | F | F |
| B             | T | T | F | F | T | T | F | F |
| C             | T | F | T | F | T | F | T | F |
| A && (B    C) | T | T | T | F | F | F | F | F |

中国软件评测中心  
China Software Testing Center



### 修正条件/判定覆盖示例(续)

A && (B || C)

|               | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---------------|---|---|---|---|---|---|---|---|
| A             | T | T | T | T | F | F | F | F |
| B             | T | T | F | F | T | T | F | F |
| C             | T | F | T | F | T | F | T | F |
| A && (B    C) | T | T | T | F | F | F | F | F |
| A             | 5 | 6 | 7 |   | 1 | 2 | 3 |   |
| B             |   | 4 |   | 2 |   |   |   |   |
| C             |   |   | 4 | 3 |   |   |   |   |

中国软件评测中心  
China Software Testing Center



§ 一个路径就是一个从函数的入口到函数的出口的唯一的系列分支

## 路径覆盖(续)

## § 使用独立路径数

```

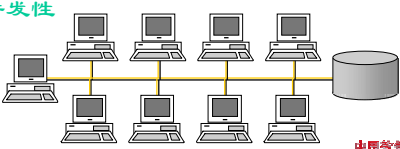
1  # Create the data frame
2  data = data.frame(
3    # Create the variables
4    x = 1:10,
5    y = 1:10,
6    z = 1:10
7  )
8
9  # Print the data frame
10 print(data)
11
12 # Create the data frame with 10 rows and 3 columns
13 data = data.frame(
14   # Create the variables
15   x = 1:10,
16   y = 1:10,
17   z = 1:10
18 )
19
20 # Print the data frame
21 print(data)
22
23 # Create the data frame with 10 rows and 3 columns
24 data = data.frame(
25   # Create the variables
26   x = 1:10,
27   y = 1:10,
28   z = 1:10
29 )
30
31 # Print the data frame
32 print(data)
33
34 # Create the data frame with 10 rows and 3 columns
35 data = data.frame(
36   # Create the variables
37   x = 1:10,
38   y = 1:10,
39   z = 1:10
40 )
41
42 # Print the data frame
43 print(data)
44
45 # Create the data frame with 10 rows and 3 columns
46 data = data.frame(
47   # Create the variables
48   x = 1:10,
49   y = 1:10,
50   z = 1:10
51 )
52
53 # Print the data frame
54 print(data)
55
56 # Create the data frame with 10 rows and 3 columns
57 data = data.frame(
58   # Create the variables
59   x = 1:10,
60   y = 1:10,
61   z = 1:10
62 )
63
64 # Print the data frame
65 print(data)
66
67 # Create the data frame with 10 rows and 3 columns
68 data = data.frame(
69   # Create the variables
70   x = 1:10,
71   y = 1:10,
72   z = 1:10
73 )
74
75 # Print the data frame
76 print(data)
77
78 # Create the data frame with 10 rows and 3 columns
79 data = data.frame(
80   # Create the variables
81   x = 1:10,
82   y = 1:10,
83   z = 1:10
84 )
85
86 # Print the data frame
87 print(data)
88
89 # Create the data frame with 10 rows and 3 columns
90 data = data.frame(
91   # Create the variables
92   x = 1:10,
93   y = 1:10,
94   z = 1:10
95 )
96
97 # Print the data frame
98 print(data)
99
100 # Create the data frame with 10 rows and 3 columns
101 data = data.frame(
102   # Create the variables
103   x = 1:10,
104   y = 1:10,
105   z = 1:10
106 )
107
108 # Print the data frame
109 print(data)
110
111 # Create the data frame with 10 rows and 3 columns
112 data = data.frame(
113   # Create the variables
114   x = 1:10,
115   y = 1:10,
116   z = 1:10
117 )
118
119 # Print the data frame
120 print(data)
121
122 # Create the data frame with 10 rows and 3 columns
123 data = data.frame(
124   # Create the variables
125   x = 1:10,
126   y = 1:10,
127   z = 1:10
128 )
129
130 # Print the data frame
131 print(data)
132
133 # Create the data frame with 10 rows and 3 columns
134 data = data.frame(
135   # Create the variables
136   x = 1:10,
137   y = 1:10,
138   z = 1:10
139 )
140
141 # Print the data frame
142 print(data)
143
144 # Create the data frame with 10 rows and 3 columns
145 data = data.frame(
146   # Create the variables
147   x = 1:10,
148   y = 1:10,
149   z = 1:10
150 )
151
152 # Print the data frame
153 print(data)
154
155 # Create the data frame with 10 rows and 3 columns
156 data = data.frame(
157   # Create the variables
158   x = 1:10,
159   y = 1:10,
160   z = 1:10
161 )
162
163 # Print the data frame
164 print(data)
165
166 # Create the data frame with 10 rows and 3 columns
167 data = data.frame(
168   # Create the variables
169   x = 1:10,
170   y = 1:10,
171   z = 1:10
172 )
173
174 # Print the data frame
175 print(data)
176
177 # Create the data frame with 10 rows and 3 columns
178 data = data.frame(
179   # Create the variables
180   x = 1:10,
181   y = 1:10,
182   z = 1:10
183 )
184
185 # Print the data frame
186 print(data)
187
188 # Create the data frame with 10 rows and 3 columns
189 data = data.frame(
190   # Create the variables
191   x = 1:10,
192   y = 1:10,
193   z = 1:10
194 )
195
196 # Print the data frame
197 print(data)
198
199 # Create the data frame with 10 rows and 3 columns
200 data = data.frame(
201   # Create the variables
202   x = 1:10,
203   y = 1:10,
204   z = 1:10
205 )
206
207 # Print the data frame
208 print(data)
209
210 # Create the data frame with 10 rows and 3 columns
211 data = data.frame(
212   # Create the variables
213   x = 1:10,
214   y = 1:10,
215   z = 1:10
216 )
217
218 # Print the data frame
219 print(data)
220
221 # Create the data frame with 10 rows and 3 columns
222 data = data.frame(
223   # Create the variables
224   x = 1:10,
225   y = 1:10,
226   z = 1:10
227 )
228
229 # Print the data frame
230 print(data)
231
232 # Create the data frame with 10 rows and 3 columns
233 data = data.frame(
234   # Create the variables
235   x = 1:10,
236   y = 1:10,
237   z = 1:10
238 )
239
240 # Print the data frame
241 print(data)
242
243 # Create the data frame with 10 rows and 3 columns
244 data = data.frame(
245   # Create the variables
246   x = 1:10,
247   y = 1:10,
248   z = 1:10
249 )
250
251 # Print the data frame
252 print(data)
253
254 # Create the data frame with 10 rows and 3 columns
255 data = data.frame(
256   # Create the variables
257   x = 1:10,
258   y = 1:10,
259   z = 1:10
260 )
261
262 # Print the data frame
263 print(data)
264
265 # Create the data frame with 10 rows and 3 columns
266 data = data.frame(
267   # Create the variables
268   x = 1:10,
269   y = 1:10,
270   z = 1:10
271 )
272
273 # Print the data frame
274 print(data)
275
276 # Create the data frame with 10 rows and 3 columns
277 data = data.frame(
278   # Create the variables
279   x = 1:10,
280   y = 1:10,
281   z = 1:10
282 )
283
284 # Print the data frame
285 print(data)
286
287 # Create the data frame with 10 rows and 3 columns
288 data = data.frame(
289   # Create the variables
290   x = 1:10,
291   y = 1:10,
292   z = 1:10
293 )
294
295 # Print the data frame
296 print(data)
297
298 # Create the data frame with 10 rows and 3 columns
299 data = data.frame(
300   # Create the variables
301   x = 1:10,
302   y = 1:10,
303   z = 1:10
304 )
305
306 # Print the data frame
307 print(data)
308
309 # Create the data frame with 10 rows and 3 columns
310 data = data.frame(
311   # Create the variables
312   x = 1:10,
313   y = 1:10,
314   z = 1:10
315 )
316
317 # Print the data frame
318 print(data)
319
320 # Create the data frame with 10 rows and 3 columns
321 data = data.frame(
322   # Create the variables
323   x = 1:10,
324   y = 1:10,
325   z = 1:10
326 )
327
328 # Print the data frame
329 print(data)
330
331 # Create the data frame with 10 rows and 3 columns
332 data = data.frame(
333   # Create the variables
334   x = 1:10,
335   y = 1:10,
336   z = 1:10
337 )
338
339 # Print the data frame
340 print(data)
341
342 # Create the data frame with 10 rows and 3 columns
343 data = data.frame(
344   # Create the variables
345   x = 1:10,
346   y = 1:10,
347   z = 1:10
348 )
349
350 # Print the data frame
351 print(data)
352
353 # Create the data frame with 10 rows and 3 columns
354 data = data.frame(
355   # Create the variables
356   x = 1:10,
357   y = 1:10,
358   z = 1:10
359 )
360
361 # Print the data frame
362 print(data)
363
364 # Create the data frame with 10 rows and 3 columns
365 data = data.frame(
366   # Create the variables
367   x = 1:10,
368   y = 1:10,
369   z = 1:10
370 )
371
372 # Print the data frame
373 print(data)
374
375 # Create the data frame with 10 rows and 3 columns
376 data = data.frame(
377   # Create the variables
378   x = 1:10,
379   y = 1:10,
380   z = 1:10
381 )
382
383 # Print the data frame
384 print(data)
385
386 # Create the data frame with 10 rows and 3 columns
387 data = data.frame(
388   # Create the variables
389   x = 1:10,
390   y = 1:10,
391   z = 1:10
392 )
393
394 # Print the data frame
395 print(data)
396
397 # Create the data frame with 10 rows and 3 columns
398 data = data.frame(
399   # Create the variables
400   x = 1:10,
401   y = 1:10,
402   z = 1:10
403 )
404
405 # Print the data frame
406 print(data)
407
408 # Create the data frame with 10 rows and 3 columns
409 data = data.frame(
410   # Create the variables
411   x = 1:10,
412   y = 1:10,
413   z = 1:10
414 )
415
416 # Print the data frame
417 print(data)
418
419 # Create the data frame with 10 rows and 3 columns
420 data = data.frame(
421   # Create the variables
422   x = 1:10,
423   y = 1:10,
424   z = 1:10
425 )
426
427 # Print the data frame
428 print(data)
429
430 # Create the data frame with 10 rows and 3 columns
431 data = data.frame(
432   # Create the variables
433   x = 1:10,
434   y = 1:10,
435   z = 1:10
436 )
437
438 # Print the data frame
439 print(data)
440
441 # Create the data frame with 10 rows and 3 columns
442 data = data.frame(
443   # Create the variables
444   x = 1:10,
445   y = 1:10,
446   z = 1:10
447 )
448
449
```

```
if ((a > 1) && (b == 0))
{
    x = x / a;
}
if ((a == 2) || (x > 1))
{
    x = x + 1;
}
```

## 6、性能与效率分析

**性能与效率分析**


- § 单个函数的性能分析
- § 函数的调用频率分析
- § 代码执行频率分析
- § 响应时间
- § 并发性



中国软件评测中心  
China Software Testing Center

**性能与效率分析**

- § 在源代码的入口与出口进行插桩，然后收集时间数据来测量代码的运行时间与调用次数
- § 针对影响性能的代码段作优化处理



中国软件评测中心  
China Software Testing Center

中国软件评测中心  
China Software Testing Center

| Item     | Value | #Incalls | #Calls | Time/Call | MinTime  | MaxTime  | CallstackSize | % Total Time |
|----------|-------|----------|--------|-----------|----------|----------|---------------|--------------|
| InitData | 1     | 1        | 1      | 0.000000  | 0.000000 | 0.000000 | 0             | 0.00%        |
| InitData | 1     | 1        | 1      | 0.000000  | 0.000000 | 0.000000 | 0             | 0.00%        |
| InitData | 1     | 1        | 1      | 0.000000  | 0.000000 | 0.000000 | 0             | 0.00%        |
| InitData | 1     | 1        | 1      | 0.000000  | 0.000000 | 0.000000 | 0             | 0.00%        |
| InitData | 1     | 1        | 1      | 0.000000  | 0.000000 | 0.000000 | 0             | 0.00%        |
| InitData | 1     | 1        | 1      | 0.000000  | 0.000000 | 0.000000 | 0             | 0.00%        |
| InitData | 1     | 1        | 1      | 0.000000  | 0.000000 | 0.000000 | 0             | 0.00%        |
| InitData | 1     | 1        | 1      | 0.000000  | 0.000000 | 0.000000 | 0             | 0.00%        |
| InitData | 1     | 1        | 1      | 0.000000  | 0.000000 | 0.000000 | 0             | 0.00%        |
| InitData | 1     | 1        | 1      | 0.000000  | 0.000000 | 0.000000 | 0             | 0.00%        |

中国软件评测中心  
China Software Testing Center

| Item     | Value | #Incalls | #Calls | Time/Call | MinTime  | MaxTime  | CallstackSize | % Total Time |
|----------|-------|----------|--------|-----------|----------|----------|---------------|--------------|
| InitData | 1     | 1        | 1      | 0.000000  | 0.000000 | 0.000000 | 0             | 0.00%        |
| InitData | 1     | 1        | 1      | 0.000000  | 0.000000 | 0.000000 | 0             | 0.00%        |
| InitData | 1     | 1        | 1      | 0.000000  | 0.000000 | 0.000000 | 0             | 0.00%        |
| InitData | 1     | 1        | 1      | 0.000000  | 0.000000 | 0.000000 | 0             | 0.00%        |
| InitData | 1     | 1        | 1      | 0.000000  | 0.000000 | 0.000000 | 0             | 0.00%        |
| InitData | 1     | 1        | 1      | 0.000000  | 0.000000 | 0.000000 | 0             | 0.00%        |
| InitData | 1     | 1        | 1      | 0.000000  | 0.000000 | 0.000000 | 0             | 0.00%        |
| InitData | 1     | 1        | 1      | 0.000000  | 0.000000 | 0.000000 | 0             | 0.00%        |
| InitData | 1     | 1        | 1      | 0.000000  | 0.000000 | 0.000000 | 0             | 0.00%        |
| InitData | 1     | 1        | 1      | 0.000000  | 0.000000 | 0.000000 | 0             | 0.00%        |

中国软件评测中心  
China Software Testing Center

| Item     | Value | #Incalls | #Calls | Time/Call | MinTime  | MaxTime  | CallstackSize | % Total Time |
|----------|-------|----------|--------|-----------|----------|----------|---------------|--------------|
| InitData | 1     | 1        | 1      | 0.000000  | 0.000000 | 0.000000 | 0             | 0.00%        |
| InitData | 1     | 1        | 1      | 0.000000  | 0.000000 | 0.000000 | 0             | 0.00%        |
| InitData | 1     | 1        | 1      | 0.000000  | 0.000000 | 0.000000 | 0             | 0.00%        |
| InitData | 1     | 1        | 1      | 0.000000  | 0.000000 | 0.000000 | 0             | 0.00%        |
| InitData | 1     | 1        | 1      | 0.000000  | 0.000000 | 0.000000 | 0             | 0.00%        |
| InitData | 1     | 1        | 1      | 0.000000  | 0.000000 | 0.000000 | 0             | 0.00%        |
| InitData | 1     | 1        | 1      | 0.000000  | 0.000000 | 0.000000 | 0             | 0.00%        |
| InitData | 1     | 1        | 1      | 0.000000  | 0.000000 | 0.000000 | 0             | 0.00%        |
| InitData | 1     | 1        | 1      | 0.000000  | 0.000000 | 0.000000 | 0             | 0.00%        |
| InitData | 1     | 1        | 1      | 0.000000  | 0.000000 | 0.000000 | 0             | 0.00%        |

**7、内存分析**

中国软件评测中心  
China Software Testing Center



## 内存分析

## § 了解程序内存分配的真实情况

- <sup>a</sup> 监测内存使用情况，发现对内存的不正常使用
- <sup>a</sup> 在系统崩溃前发现内存泄露错误

## § 发现内存分配错误

- <sup>a</sup> 精确显示发生错误时的上下文情况
- <sup>a</sup> 指出发生错误的原由
- <sup>a</sup> 在问题出现前发现征兆



中国软件评测中心



## 内存错误

## § 内存分配未成功，却使用了它

### § 内存分配虽然成功，但是尚未初始化就引用它

### § 内存分配成功并且已经初始化，但操作越过了内存的边界

## § 忘记了释放内存，造成内存泄露

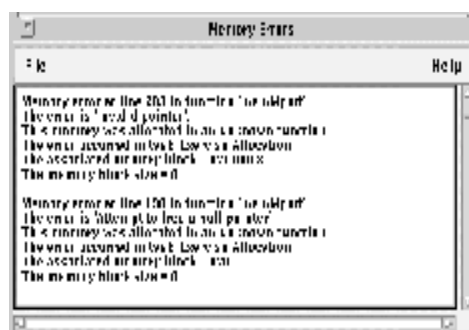
## § 释放了内存却继续使用它

中国软件评测中心

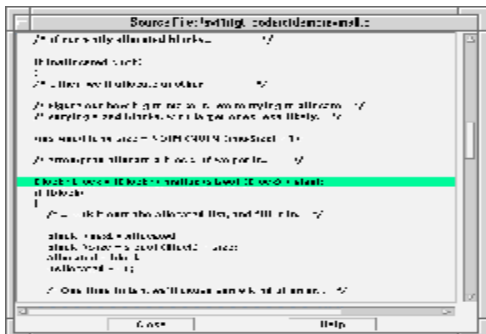


| File    | Package Name  | File Name | Line | Class | Type | Min | Max | Avg | StDev | Min | Max | Avg | StDev | Unit |
|---------|---------------|-----------|------|-------|------|-----|-----|-----|-------|-----|-----|-----|-------|------|
| Control | org.apache... | ...       | 218  | ...   | ...  | 4.0 | 4.0 | 4.0 | 0.0   | 1.0 | 1.0 | 1.0 | 0.0   | ...  |
| Control | org.apache... | ...       | 318  | ...   | ...  | 4.0 | 4.0 | 4.0 | 0.0   | 1.0 | 1.0 | 1.0 | 0.0   | ...  |
| Control | org.apache... | ...       | 112  | ...   | ...  | 3.0 | 4.0 | 3.0 | 0.0   | 1.0 | 1.0 | 1.0 | 0.0   | ...  |
| Control | org.apache... | ...       | 400  | ...   | ...  | 3.0 | 4.0 | 3.0 | 0.0   | 1.0 | 1.0 | 1.0 | 0.0   | ...  |
| Control | org.apache... | ...       | 344  | ...   | ...  | 3.0 | 4.0 | 3.0 | 0.0   | 1.0 | 1.0 | 1.0 | 0.0   | ...  |
| Control | org.apache... | ...       | 100  | ...   | ...  | 2.0 | 3.0 | 2.0 | 0.0   | 1.0 | 1.0 | 1.0 | 0.0   | ...  |

中国软件评测中心



中国软件评测中心



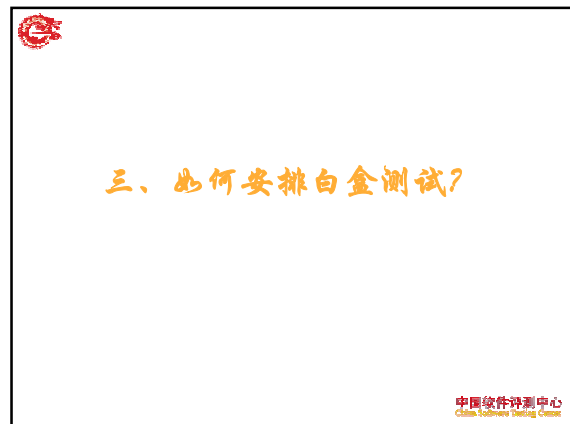
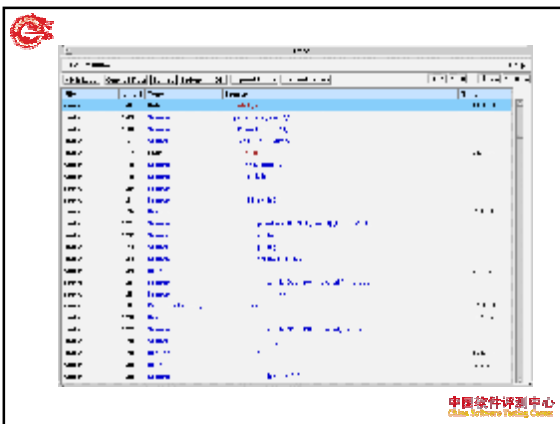
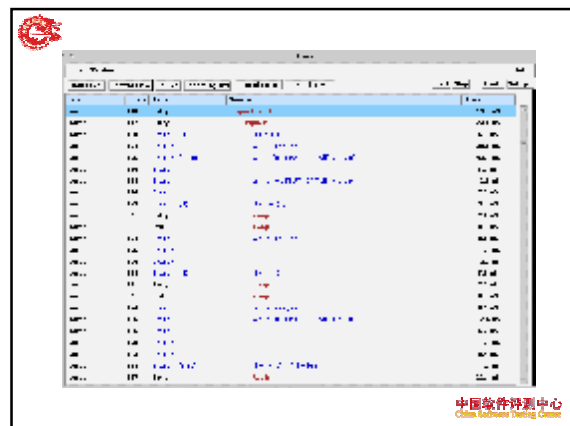
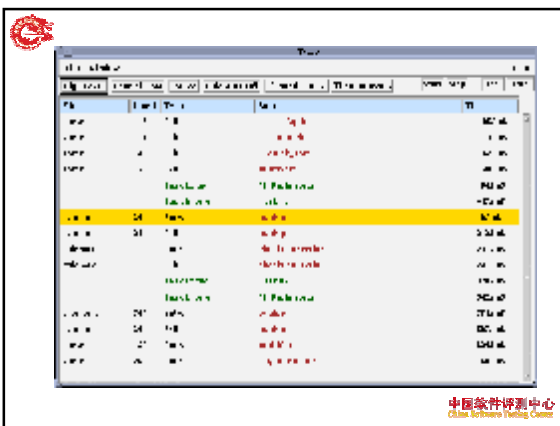
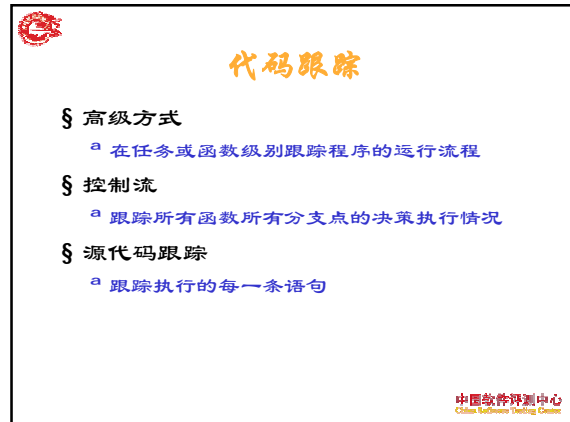
中国软件评测中心




```
int i;
CString strTitle0[] = {_T("站名"),_T("经度"),_T("纬度"),_T("高度")};
unsigned short usLen0[] =
{70,135,135,40};
.....
for ( i = 0; i < 5; i++ )
    m_list Equip_deploy.InsertColumnn(i,strTitle0[i],LVCFMT_LEFT,usLen0[i],-1);
```

中国软件评测中心








## 按阶段

- § 单元测试
- § 集成测试
- § 系统测试
- § 验收测试


中国软件评测中心  
China Software Testing Center



## 软件测试方法在实际中的应用

- § 黑盒测试和白盒测试相结合
- § 静态和动态相结合
- § 代码审查和静态结构分析、代码质量度量相结合


中国软件评测中心  
China Software Testing Center



## 软件测试完成准则

- § 使用了特定的测试用例
- § 查出了预定数目的错误
- § 错误强度曲线下降到预定的水平
- § 达到测试计划中所规定的测试项
- § 其它标准


中国软件评测中心  
China Software Testing Center



## 测试工具


- § 纯软件的测试工具
- § 纯硬件的测试工具（如逻辑分析仪和仿真器等）
- § 软硬件结合的测试工具

中国软件评测中心  
China Software Testing Center



|         | 基本原理        | 缺陷              |
|---------|-------------|-----------------|
| 纯软件测试工具 | 软件打点        | 性能数据不精确         |
| 纯硬件测试工具 | 信号采样        | 一般用于系统运行频率不高的情况 |
| 软硬件结合工具 | 软件打点 + 总线捕获 |                 |

中国软件评测中心  
China Software Testing Center



## 白金测试工具

- § 静态结构分析
- § 代码质量度量
- § 代码检查
- § 功能确认与接口测试
- § 覆盖率分析
- § 性能分析
- § 内存分析

中国软件评测中心  
China Software Testing Center

