


Shanghai Jiao Tong University




上海交通大学

软件工程

Module7: 软件测试

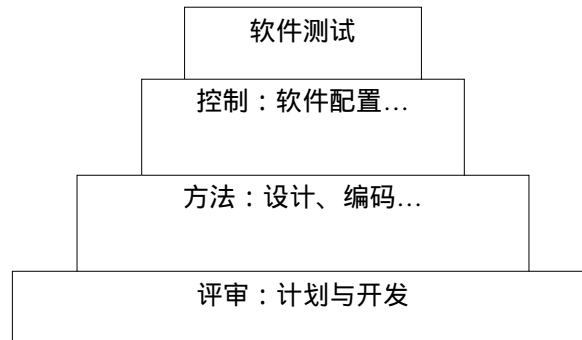
上海交通大学计算机系



软件测试

- ◆ 测试的概念和测试生命周期
- ◆ 测试用例设计
 - 黑盒测试
 - 白盒测试
- ◆ 测试策略和测试过程

质量保证的活动内容



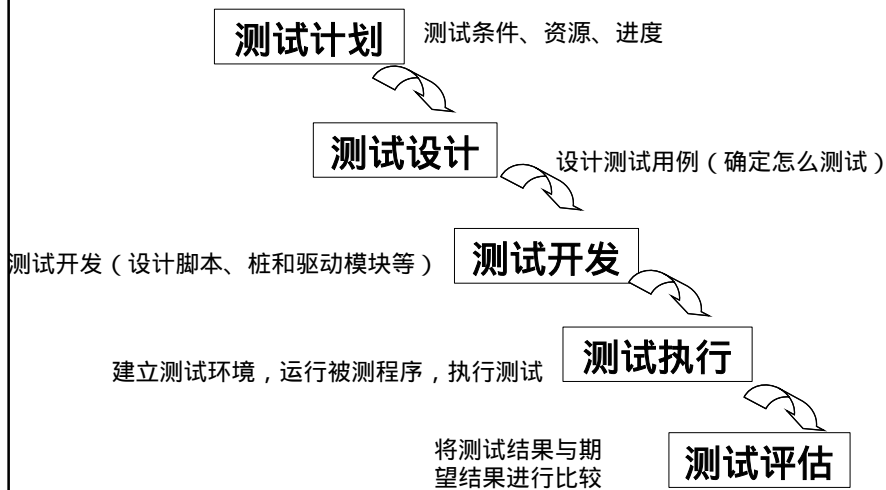
测试的基本概念

- ◆ 测试 (testing) 的目的与任务
 - 目的：发现程序的错误
 - 任务：通过执行程序，暴露潜在的错误
 - 成功的测试：发现了未曾发现的错误
- ◆ 调试 (debugging) 的目的与任务
 - 目的：定位和纠正错误
 - 任务：消除软件故障，保证程序的可靠运行

测试的原则

- ◆ 原则一：穷尽测试是不可能的
- ◆ 原则二：测试工作具有创造性和挑战性
- ◆ 原则三：测试旨在发现存在的缺陷
- ◆ 原则四：测试是有风险的
- ◆ 原则五：测试需要有计划性
- ◆ 原则六：测试需要有独立性

测试生存周期



- ◆ 测试计划、测试设计和测试开发在软件开发完成前进行
- ◆ 测试执行只能在软件开发完成后进行

1) 测试计划

- ◆ 测试计划主要内容：
 - 测试目标
 - 测试策略
 - 测试参与人员
 - 测试的进度安排
 - 测试的系统环境

Review 测试计划模板

测试团队 --测试经理

- ◆ 职责
 - 全面指导--测试计划
 - 收集信息
 - 项目报告--测试评估
- ◆ 要求
 - 有测试方法理论的知识
 - 懂项目管理
 - 熟悉测试工具

测试团队 --测试工程师 (设计者/开发者)

- ◆ 职责
 - 细化测试需求
 - 测试设计
 - 测试开发
- ◆ 要求
 - 有应用需求方面的知识
 - 熟悉测试工具
 - 软件开发技术和经验

测试团队 --测试工程师 (测试执行)

- ◆ 职责
 - 执行测试
 - 记录测试结果
 - 缺陷报告
 - 恢复错误
- ◆ 要求
 - 了解要测试的系统、网络、服务器等
 - 熟悉测试工具
 - 诊断找错的技术

2) 测试设计

- ◆ 任务：设计测试用例
- ◆ 测试用例 (test case) 是按一定顺序执行的与测试目标相关的一系列测试。其主要内容包括：
 - 前置条件 (Pre-conditions) Optional
 - 测试输入 (Test input)
 - 观察点 (Observation Points) Optional
 - 控制点 (Control Points) Optional
 - 期望结果 (Expected Results)
 - 后置条件 (Post-conditions) Optional

Review 测试用例文档模板

举例：测试用例

- ◆ 一个加法程序 `int Add(int a, int b)` 的测试用例
 - 测试输入 (Test input)
 - 输入 a 为 2, b 为 3
 - 期望结果 (Expected Results)
 - 输出结果为 5

3) 测试开发

- ◆ 任务：开发测试脚本、桩和驱动模块
- ◆ 测试脚本（test script）是具有正规语法的数据和指令的集合，在测试执行自动工具使用中，通常以文件形式保存

4) 测试执行

- ◆ 任务：执行测试用例
- ◆ 对于手动测试：
 - 测试者按事先准备好的手工过程进行测试，测试者输入数据、观察输出、记录发现的问题。
- ◆ 对于自动测试：
 - 可能只需要启动测试工具，并告诉工具执行哪些测试用例。

5) 测试评估

- ◆ 任务：将测试结果与期望输出进行比较，判断软件功能是否正确，编写缺陷报告和测试记录
- ◆ 软件缺陷(bug)是对软件产品预期属性的偏离
 - 对产品需求规约的偏离
 - 如：需求规定了 $a+b=> c$ ，而软件产品实际上做的是： $a+b= c$
 - 对用户期望的偏离，即用户要求未体现在需求规约中
 - 软件做了用户不希望做的事（画蛇添足）

缺陷等级

- ◆ 0级：灾难性的--系统崩溃、数据被破坏
- ◆ 1级：很严重的--数据被破坏
- ◆ 2级：严重的--特性不能运行，无法替代
- ◆ 3级：中等的--特性不能运行，可替代
- ◆ 4级：烦恼的--提示不正确，报警不确切
- ◆ 5级：轻微的--表面化的错误，拼写错等

缺陷报告 (Bug Report)

- ◆ 项目名称
- ◆ 测试日期
- ◆ 测试人
- ◆ 缺陷等级
- ◆ 缺陷描述

Review 缺陷报告模板

测试记录 (Test Record)

回归测试

- ◆ 为了保证软件返工时没有引进新的错误，要全部或部分地重复以前做过的测试。

终止测试的标准

- ◆ 在测试计划中规定终止测试的标准
- ◆ 终止测试的常用标准
 - 所有严重的缺陷都已纠正，剩余的缺陷密度少于0.01%
 - 100%测试覆盖度
 - 至少要查出***缺陷数量
 - 1000个CPU小时内不出错的操作概率大于99.5%

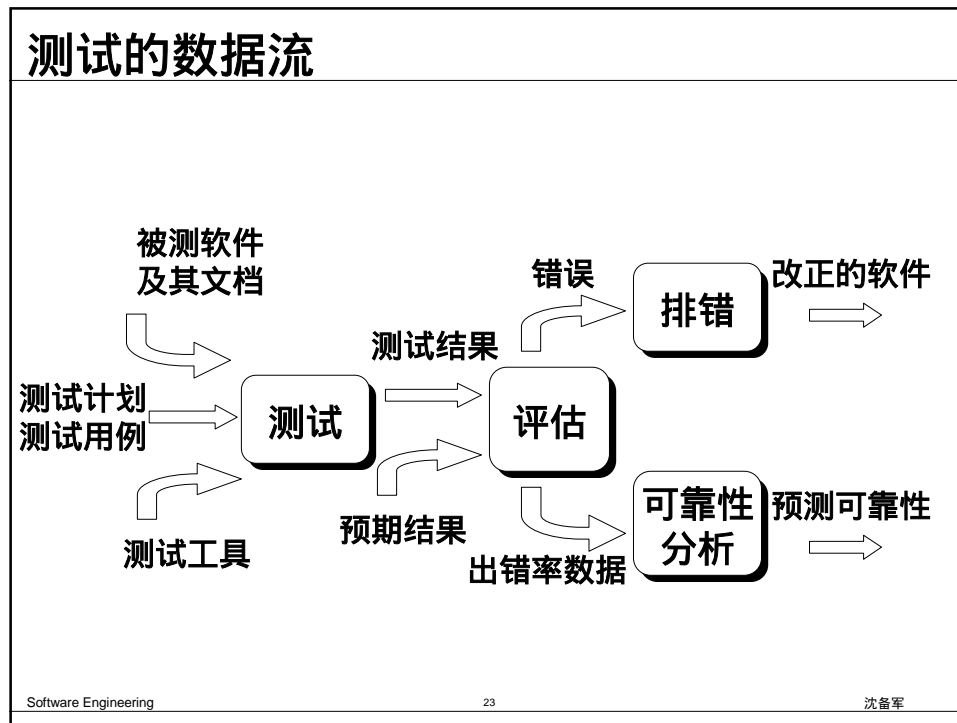
测试评估报告

- ◆ 在测试完成后撰写测试评估报告
- ◆ 测试评估报告主要内容：
 - 测试项目名称
 - 测试结果摘要
 - 基于需求的测试覆盖
 - 基于代码的测试覆盖
 - 建议措施
 - 统计分析图（测试进度、费用、缺陷...）

Review 测试评估报告模板

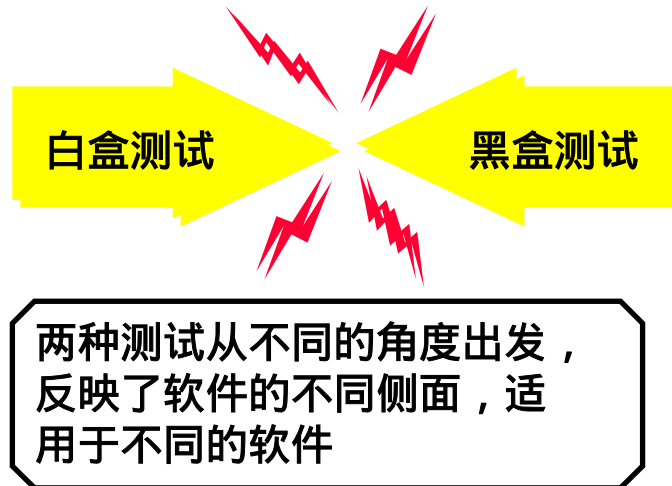
测试的文档

- ◆ 测试计划 – 在测试计划阶段
- ◆ 测试用例文档 – 在测试设计阶段
- ◆ 缺陷报告、测试记录 – 在测试执行阶段
- ◆ 测试评估报告 – 在测试完成后



- ### 软件测试
- ◆ 测试的概念和测试生命周期
 - ◆ 测试用例设计
 - 黑盒测试
 - 白盒测试
 - ◆ 测试策略和测试过程
- Software Engineering 24 沈备军

测试用例设计的两种通用方法



黑盒测试

- ◆ 黑盒测试把程序看成一个黑盒子，完全不考虑程序内部结构和处理过程。
- ◆ 黑盒测试是在程序接口进行测试，它只是检查程序功能是否按照需求规约正常使用。
- ◆ 黑盒测试又称功能测试、行为测试，在软件开发后期执行。



黑盒测试技术

- 1) 等价类划分法
- 2) 边界值划分法
- 3) 错误推测法
- 4) 因果图法

1) 等价类划分(equivalence partitioning)

- ◆ 基本概念
 - 试遍所有输入数据是不可能的
 - 等价类划分的办法是把程序的输入域划分成若干部分，然后从每个部分中选取少数代表性数据当作测试用例。
 - 输入的数据划分为合理等价类和不合理等价类
- ◆ 例子：
 - 输入数据：每个学生可以选取修1至3门课程则
 - 合理等价类：选修1至3门课程
 - 不合理等价类：没选修课程以及超过3门课程

确定等价类原则

(1) 如果输入条件规定了取值范围或值的个数，则可确定一个有效等价类和两个无效等价类。

- ◆ 例1 “...项数可以从1到999...”
 - 有效等价类为 “1 项数 999”
 - 无效等价类为 “项数<1” 及 “项数>999”
- ◆ 例2 “学生选课允许2门至4门”
 - 有效等价类： 选课2至4门
 - 无效等价类： 只选一门课或未选课
选课超过4门

确定等价类原则（续）

(2) 输入条件规定了输入值的集合，或是规定了“必须如何”的条件，则可确定一个有效等价类和一个无效等价类。

- 例：“必须以标识符以字母开头的字符串”
 - 有效等价类： 以字母开头的字符串
 - 无效等价类： 以非字母开头的字符串

(3) 如果确知，已划分的等价类中各元素在程序中的处理方式是不同的，则应将此等价类进一步划小。

设计测试用例

- (1) 设计一个测试用例，使其尽可能多地覆盖有效等价类，重复这一步，最终使得所有有效等价类均被覆盖。
- (2) 设计一个测试用例，使其只覆盖一个无效等价类，重复这一步，最终使得所有无效等价类均被覆盖。
- ◆ 例：项数可以从1到999
 - 有效等价类 “1 项数 999” -- 777
 - 无效等价类
 - “项数<1” --- 0
 - “项数>999” --- 1200

2) 边界值划分(boundary value analysis)

- ◆ 基本概念
 - 边界值划分法使被测程序在边界值及其附近运行，从而更有效地暴露程序中潜藏的错误
 - 不仅根据输入条件，它还根据输出情况设计测试用例
- ◆ 例子：
 - 输入条件-1.0到1.0
 - 则选择-1.0，1.0，-1.001和1.001等

边值分析遵循的原则

如果输入条件规定了取值范围，或是规定了值的个数，应以该范围的边界内及刚刚超出范围的边界外的值，或是分别对最大、最小个数及稍小于最小、稍大于最大个数作为测试用例。

- ◆ 例1：“重量在10公斤至50公斤范围内的邮件，其邮费计算公式为……”
 - 应取10及50，还应取10.01, 49.99, 9.99, 50.01等
- ◆ 例2：“某输入文件可包含1至255个记录，……”
 - 可取1和255，还应取0及256等

边值分析遵循的原则（续）

针对需求的每个输出条件使用前面的第 条原则。

- ◆ 例1: 计算出“每月保险金扣除额为0至1165.25元”
 - 可取0.00及1165.2、还可取 -0.01及1165.26等
- ◆ 例2: 情报检索系统要求每次“最多显示1~4条情报摘要”
 - 可取1和4，还应取0和5等。

边值分析遵循的原则（续）

如果程序规格说明中提到的输入或输出域是个有序的集合（如顺序文件、表格等），就应注意选取有序集的第一个和最后一个元素作为测试用例。

3) 错误推测法(error guessing)

- ◆ 基本概念
 - 猜测被测程序在哪些地方容易出错
 - 针对可能的薄弱环节来设计测试用例
- ◆ 例子：对一个排序程序，该检查：
 - 输入表为空
 - 输入表中只有一行
 - 输入表中所有的值具有相同的值
 - 输入表已经是排序的

4) 因果图法 (Cause-Effect Graphics)

- ◆ 检查输入条件的各种组合情况
- ◆ 从功能说明中找出因 (输入条件) 和果 (输出或程序状态的修改)
- ◆ 通过因果图功能说明转换成一张判定表, 然后为判定表的每一例设计测试用例

黑盒测试用例设计策略

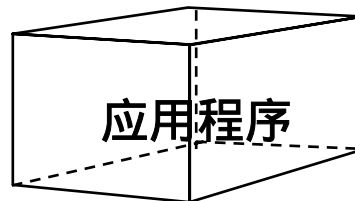
- ◆ 不管情况怎样, 都使用边值分析方法;
- ◆ 对输入和输出划分合格的和不合格的两个等价类, 作为补充;
- ◆ 如果规范中含有输入条件的组合, 便从因果图开始:
原因: 输入条件或输入条件的等价类;
结果: 输出条件或系统变换;
因果图: 连接原因与结果的布尔图;
- ◆ 使用“猜测”技巧, 增加一些测试用例

练习

- ◆ 某工厂公开招工，规定报名者年龄应在16周岁至35周岁之间(到2002年3月30日止)即出生年月不在上述范围内，将拒绝接受，并显示“年龄不合格”等出错信息。
- ◆ 请采用等价类划分法设计测试用例。

白盒测试

- ◆ 白盒测试的前提是可以把程序看成装在一个透明的白盒子里，也就是完全了解程序结构盒处理过程，这种方法按照程序内部逻辑测试程序，检验程序中每条通路是否按预定要求正确工作。
- ◆ 白盒测试又称玻璃盒测试，在软件开发早期（即编码阶段）执行。

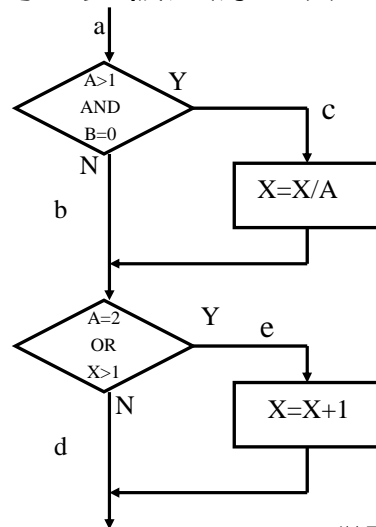


白盒测试技术

- 1) 语句覆盖法
- 2) 判定覆盖 (分支)
- 3) 条件覆盖
- 4) 判定/条件覆盖
- 5) 条件组合覆盖
- 6) 路径覆盖

1) 语句覆盖法

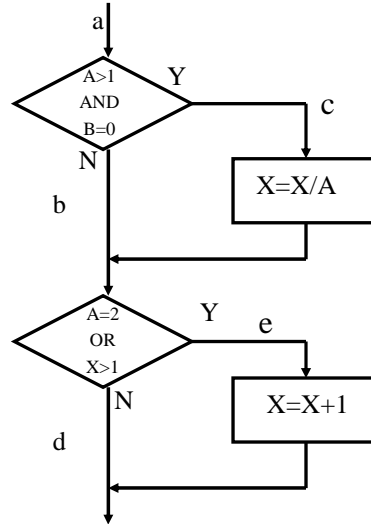
- ◆ 使得程序中的每一个语句至少被遍历一次
- ◆ 测试用例：
A=2, B=0, X=3



2) 判定覆盖 (分支)

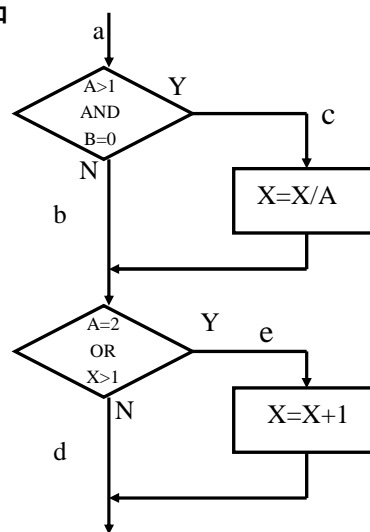
- ◆ 使得程序中每一个分支至少被遍历一次
- ◆ 测试用例

1. $A=3, B=0, X=1$
(沿路径 ace)
2. $A=1, B=0, X=0$
(沿路径 abd)



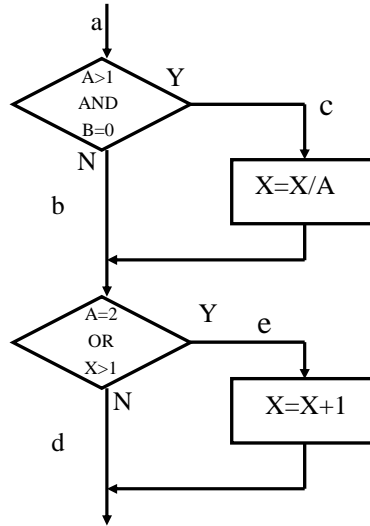
3) 条件覆盖

- ◆ 使得每个判定的条件获取各种可能的结果
 - ◆ 在a点 $A > 1, A = 1, B = 0, B = 1$
 - ◆ 在b点 $A = 2, A = 1, X > 1, X = 1$
 - ◆ 测试用例
1. $A=2, B=0, X=4$
(沿路径ace)
 2. $A=1, B=1, X=1$
(沿路径abd)



4) 判定/条件覆盖

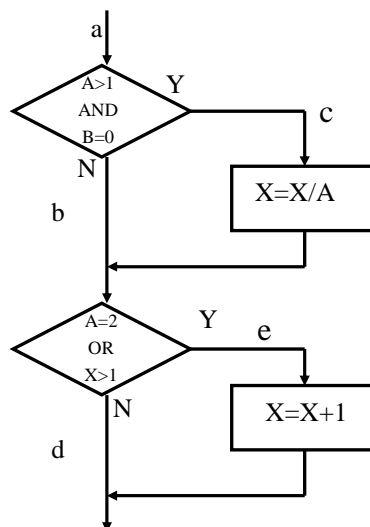
- ◆ 使得判定中的条件取得各种可能的值，并使得每个判定取得各种可能的结果
- ◆ 测试用例
 1. $A=2, B=0, X=4$
(沿路径ace)
 2. $A=1, B=1, X=1$
(沿路径abd)



5) 条件组合覆盖

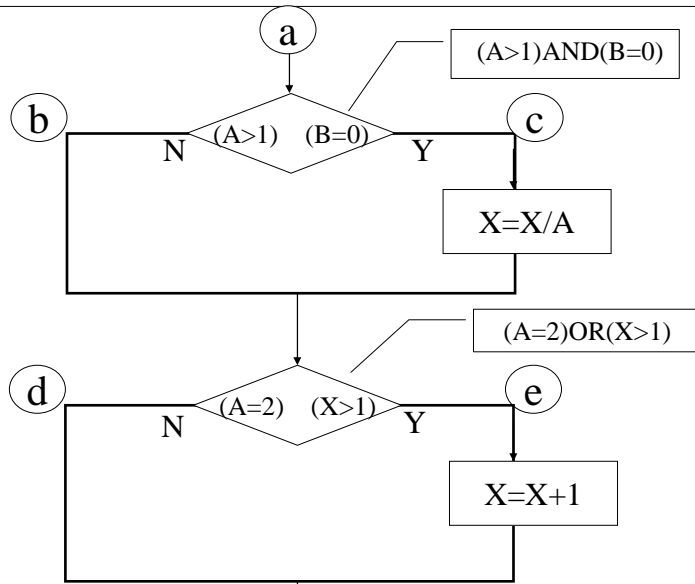
- ◆ 使得每个判定条件的各种可能组合都至少出现一次
- ◆ 要求

1. $A>1, B=0$	5. $A=2, X>1$
2. $A>1, B=1$	6. $A=2, X=1$
3. $A=1, B=0$	7. $A=1, X>1$
4. $A=1, B=1$	8. $A=2, X=1$
- ◆ 测试用例
 1. $A=2, B=0, X=4$
 2. $A=2, B=1, X=1$
 3. $A=1, B=0, X=2$
 4. $A=1, B=1, X=1$



6) 路径覆盖

覆盖
程序中
所有可
能的
路径



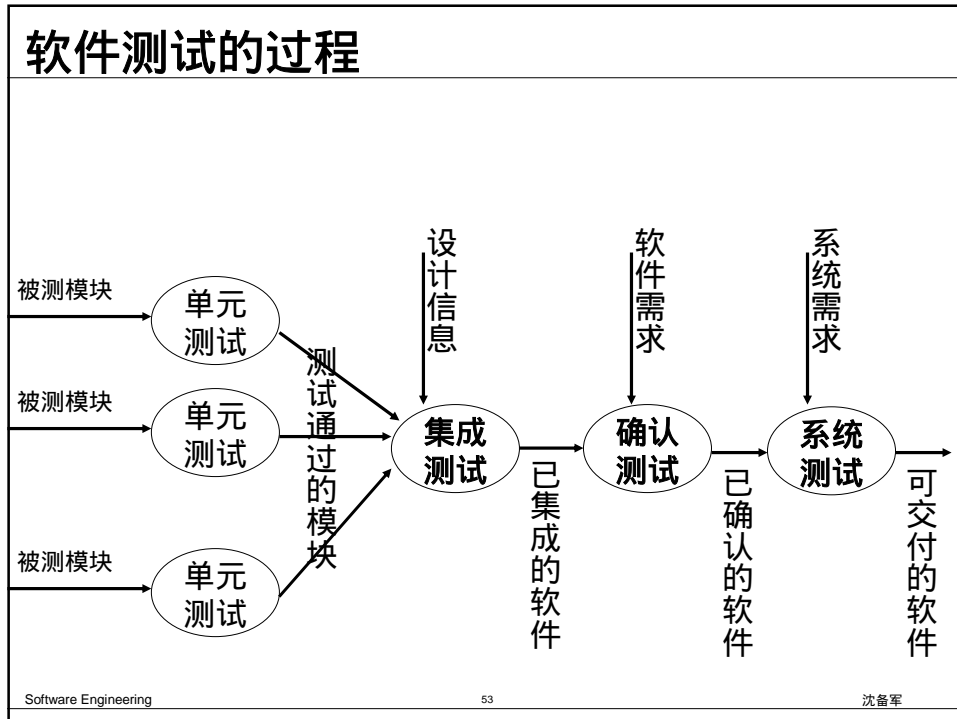
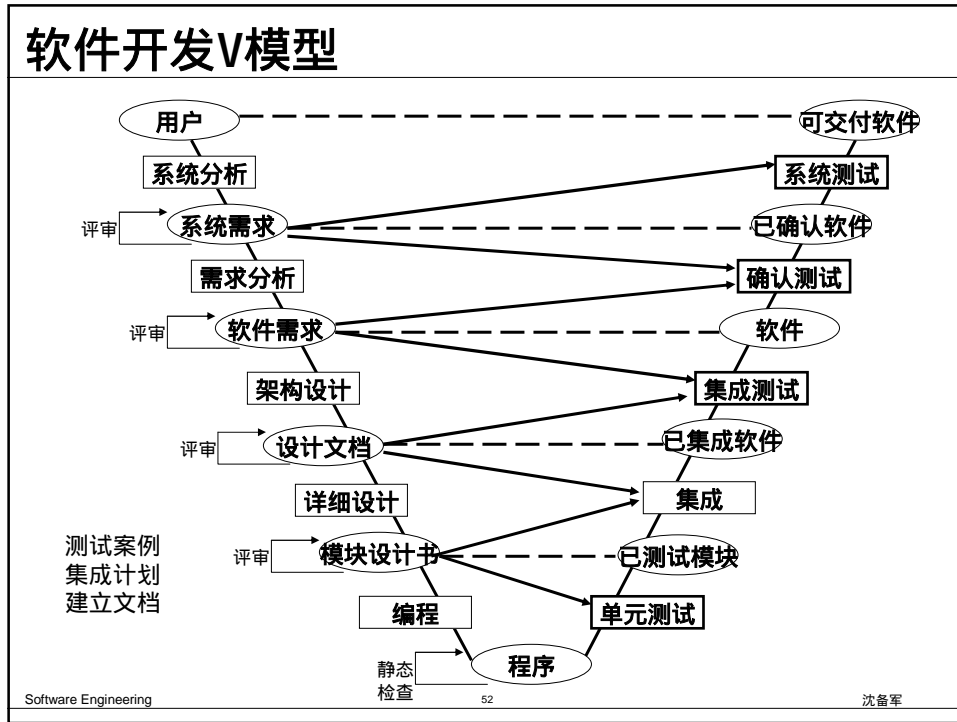
A	B	X	覆盖路径
2	0	3	a c e L ₁
1	0	1	a b d L ₂
2	1	1	a b e L ₃
3	0	1	a c d L ₄

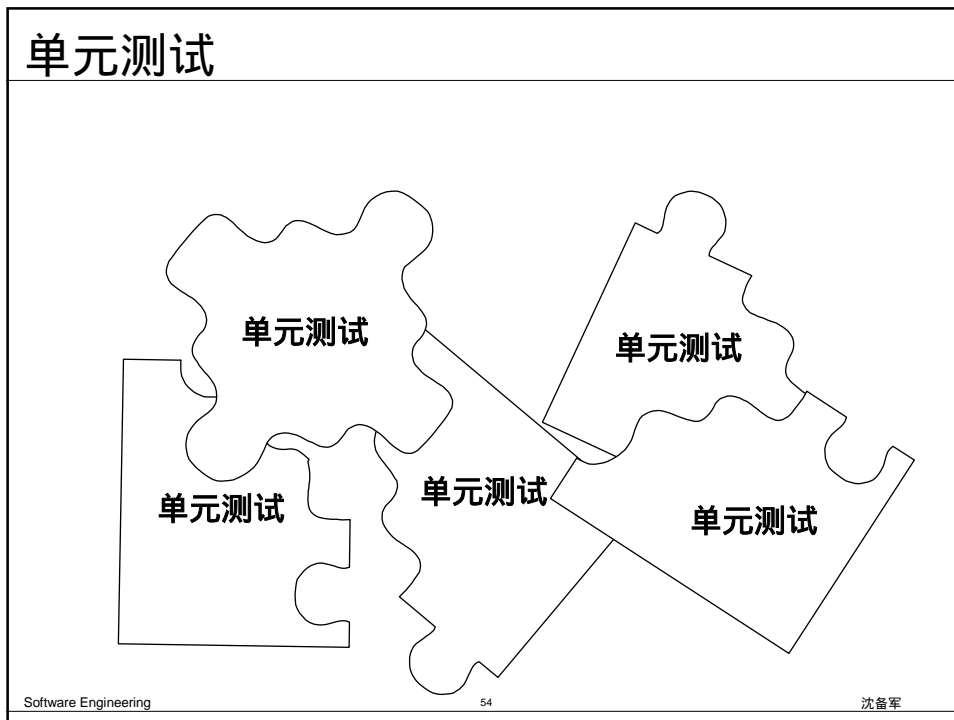
作业：测试用例设计

- ◆ 黑盒测试用例设计
 - 三角形分类程序
 - 见教材p342 第17.1题
- ◆ 白盒测试用例设计
 - 排序程序
 - 见教材p343 第17.4题

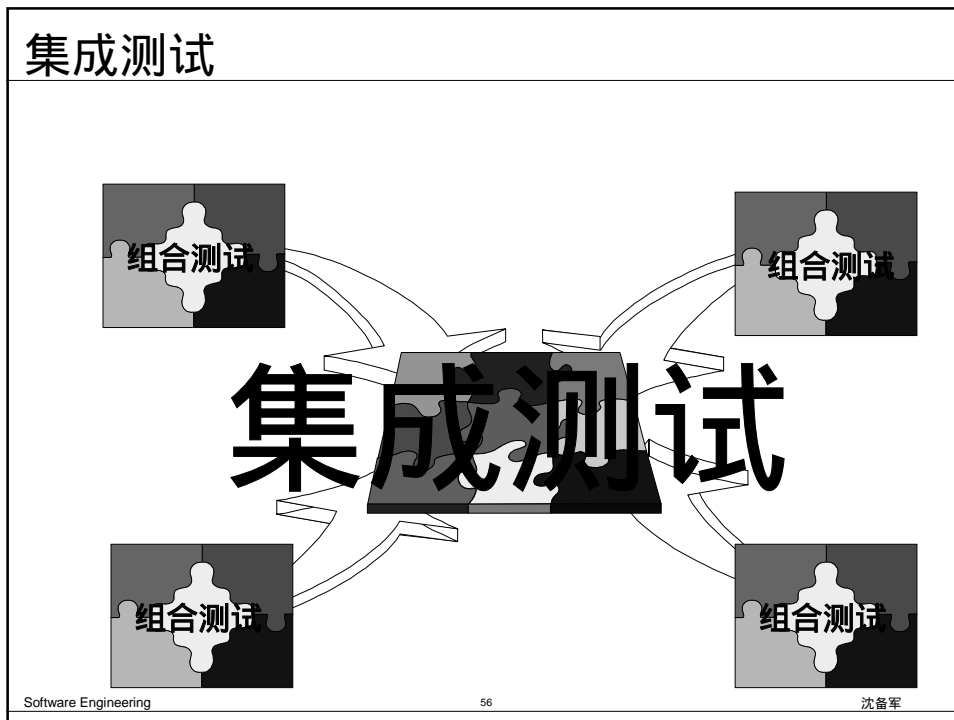
软件测试

- ◆ 测试的概念和测试生命周期
- ◆ 测试用例设计
 - 黑盒测试
 - 白盒测试
- ☀ ◆ 测试策略和测试过程





- ## 单元测试
- ◆ 目的
 - 通过模块测试，使其代码达到模块设计的要求
 - ◆ 任务
 - (1) 对模块代码进行编译，发现其语法错误；
 - (2) 确定模块的测试策略（通常采用白盒测试法），并据此设计一组测试用例；
 - (3) 用选定的测试用例对模块进行测试，直至满足测试终止标准为止；
 - (4) 编制单元测试评估报告。
- Software Engineering 55 沈备军



- ## 集成测试
- ◆ 目的
 - 将经过单元测试的模块逐步组装成具有良好一致性的完整的程序
 - ◆ 任务
 - 制订集成测试实施策略
 - 确定集成测试的实施步骤，设计测试用例
 - 逐一地添加模块，进行测试
- Software Engineering 57 沈备军

集成测试

- ◆ 策略与步骤
 - 自顶向下测试
 - 先广后深实施步骤
 - 先深后广实施步骤
 - 由底向上测试
 - 混合方式测试 (sandwich testing)
 - 对上层模块采取自顶向下测试
 - 对关键模块或子系统采取由底向上测试
 - 一次性集成测试

确认测试

- ◆ 目的
 - 确认组装好的程序是否满足软件需求 (SRS)
- ◆ 任务
 - 有效性测试 (黑盒测试)
 - 配置复审 (configuration review)
- ◆ 验收测试—针对专用应用软件
- ◆ alpha与beta测试—针对通用产品软件

系统测试

Software Engineering 60 沈备军

系统测试

- ◆ 目的
 - 软件安装到系统中以后，能否与系统的其余部分协调运行
- ◆ 任务
 - 测试是否与硬件协调运行
 - 测试是否和原来就有的其它软件协调运行
 - 测试是否完成系统需求对它的要求

Software Engineering 61 沈备军

系统测试技术

- ◆ 安全和存取控制测试
- ◆ 故障及恢复测试
- ◆ 性能测试
- ◆ 强度测试 (Stress Testing)

测试工具

- ◆ 性能测试工具
 - Rational's Quantify, Load test, Mercury Interactive's LoadRunner
- ◆ 可靠性测试工具
 - Rational's Purify, PureCoverage
- ◆ 单元测试工具
 - JUnit
- ◆ 功能测试工具
 - Rational's Robot, Mercury Interactive's WinRunner
- ◆ 测试数据生成工具
 - ASTI's DataGen
- ◆ 测试管理工具
 - Rational ClearQuest, Bugzilla (Freeware), Mantis (Freeware)
- ◆ 自行开发