

软件质量与评测技术

Software Quality & Evaluation Technology

计算机学院 单纯
sherryshan@bit.edu.cn
2025年11月

测试设计技术

Test Design Techniques

计算机学院 单纯

sherryshan@bit.edu.cn

2025年12月8日

内容概览

- 1. 测试点的确定
- 2. 测试用例设计概述
- 3. 测试用例设计技术

1. 测试点的确定（1）

■ ISO质量体系

- 在概要设计或详细设计中应明确指出每个单元模块的测试要点、指标和方法

■ CMM质量体系

- 在系统的用例模型描述中应明确指出每个用例模型的优先级及用例工作流程，每一个用例模型为一个测试点，用例模型中每一个测试需求至少应有两个测试用例

1. 测试点的确定（2）

- 测试用例应由测试设计员或分析设计员来制定，而不是普通的测试员
- 测试点应由分析设计员确立，与测试人员无关
- 测试工作展开于项目立项后，而不是代码开发完成之后
- 测试对象不仅仅是源代码，还包括需求分析、需求规格说明书、概要设计、概要设计说明书、详细设计、详细设计说明书、使用手册等各阶段的文档

2. 测试用例设计概述（1）

- 测试用例是为了特定目的（如考察特定程序路径或验证是否符合特定的需求）而设计的测试数据及与之相关的测试规程的一个特定的集合，或称为有效地发现软件缺陷的最小测试执行单元

2. 测试用例设计概述（2）

- 软件质量的好坏很大程度上取决于测试用例的数量和质量。不论程序员的编程水平、软件设计水平有多高，软件过程执行得如何好，如果没有通过合适数量和质量的测试用例进行测试，其最终的软件质量都是难以保证的

2. 测试用例设计概述 (3)

- 2.1 测试用例的重要性
- 2.2 测试用例数和软件规模的关系
- 2.3 测试用例设计步骤

2.1 测试用例的重要性（1）

- 测试用例是测试人员执行测试的重要参考依据
 - 不同的测试人员根据相同的测试用例所得到的输出应该是一致的，测试实际上就是测试用例的计划、执行和跟踪的过程
- 良好的测试用例具有复用的功能，在测试过程中可以重复使用。因此，设计良好的测试用例可以大大节约时间，提高测试效率

2.1 测试用例的重要性（2）

- 测试用例是在长期测试实践中积累起来的。组织好这些测试用例，可帮助测试者有效使用它们
- 测试用例的通过率是检验程序代码质量的例证。衡量程序代码的质量，量化的标准应该是测试用例的通过率和软件缺陷（bug）的数目

2.1 测试用例的重要性（3）

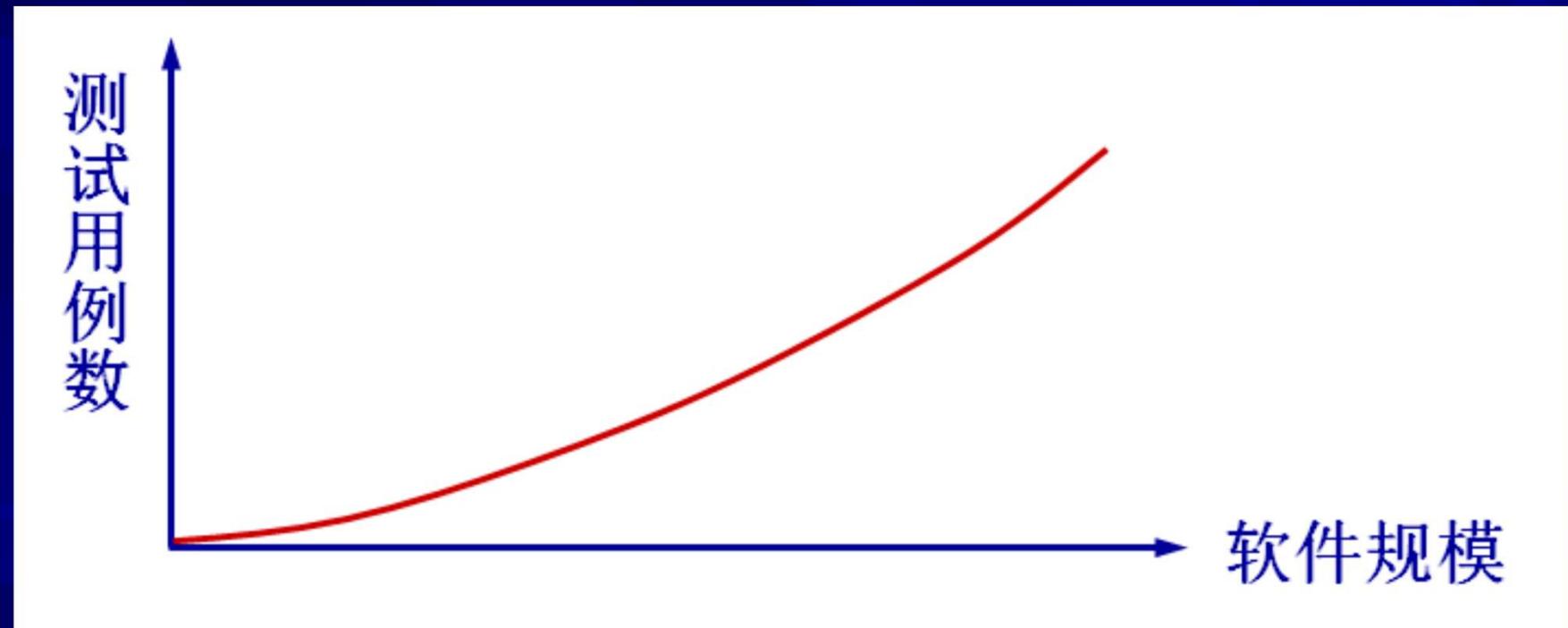
- 测试用例也是检验测试人员进度、工作量以及跟踪/管理测试人员的工作效率的因素。尤其适用于对于新的测试人员考核，从而更加合理做出测试安排和计划
- 测试用例不是每个人都可以编写的，它需要撰写者对用户场景、功能规格说明、产品的设计以及程序/模块的结构都有比较透彻的了解

2.2 测试用例数和软件规模的关系（1）

- 对软件质量的要求不同，测试用例数与程序源代码规模的比例不同。如航天软件的测试用例数与程序源代码行数的比例就高于普通民用软件
- 软件的规模不同，若要达到相同的软件质量，软件的测试用例数与程序代码行数的比例也不同，大规模软件应该比小规模软件更高些

2.2 测试用例数和软件规模的关系 (2)

- 测试用例数与软件规模的比例如下图所示



2.2 测试用例数和软件规模的关系 (3)

- 对于单元测试来说，软件规模与测试用例数基本上是成比例的
 - 对于集成测试和系统测试，测试用例数与软件规模不是简单的正比关系，软件规模越大，模块间关系越复杂，组合情况越多，测试用例数越大
- 测试用例的设计难度与软件规模也有关系
：软件规模越大，测试用例的设计难度就越大

2.3 测试用例设计步骤

- 2.3.1 为测试需求确定测试用例
- 2.3.2 为测试用例确定输入和输出
- 2.3.3 编写测试用例
- 2.3.4 评审测试用例
- 2.3.5 跟踪测试用例

2.3.1 为测试需求确定测试用例

- 测试需求：来源于规格说明书（用例、补充规约），设计规格。在测试计划中明确
- 测试需求编号：TR_XXXX_XX
- 每一个测试需求至少确定两个测试用例：正面、负面

2.3.2 为测试用例确定输入输出 (1)

- 输入是指在执行该测试用例时，由用户输入的与之交互的对象、字段和特定的数据值（或生成的对象状态）
- 输出即预期结果，是指执行该测试用例完毕后得到的状态或数据

2.3.2 为测试用例确定输入输出 (2)

- 在确定输入和输出参数时，采用以下原则
 - 在任何情况下都必须使用边界值分析方法。经验表明用这种方法设计出测试用例发现程序错误的能力最强
 - 必要时用等价类划分方法补充一些测试用例
 - 用错误推测法再追加一些测试用例
 - 对照程序逻辑，检查已设计出的测试用例的逻辑覆盖程度。如果没有达到要求的覆盖标准，应当再补充足够的测试用例
 - 如果程序的功能说明中含有输入条件的组合情况，则一开始就可以选择因果图法

2.3.3 编写测试用例

- 测试用例编号为： TC_ 测试需求标识
- 测试需求标识： 测试计划中的测试需求标识
- 测试目标状态和测试数据状态： 执行此用例前系统应具备的状态
- 输入（操作）： 为各输入数据（操作）的组合
- 输出（预期结果）： 测试用例执行后得到的状态或数据

测试用例的书写规范（1）

■ 主要元素如下

- 标识符：每个测试用例应有一个唯一的标识，作为引用的基本元素
- 测试项：测试用例应准确地描述被测试项及其特征。如做 Windows 应用程序的窗口测试，测试对象是整个应用程序用户界面，其特性要求包括窗口缩放、界面布局、菜单等

测试用例的书写规范（2）

■ 主要元素如下（续）

- 测试环境要求：用来表明执行该测试用例需要的测试环境，可根据被测模块对测试环境的需求来描述测试用例的测试环境
- 输入数据：用来执行测试用例的输入数据
- 对应输出数据：表示按照指定的环境和输入标准得到的期望输出结果
- 测试用例间的关联：用来标识该测试用例与其他的测试（或其他测试用例）间的依赖关系

2.3.4 评审测试用例（1）

■ 测试用例检查表（checklist）

- 是否每一个需求都有一个对应的测试用例来验证？
- 是否每一个设计元素都有其对应的测试用例来验证？
- 或事件顺序，它能够产生唯一的测试目标行为？
- 是否每个测试用例都阐述了预期结果？

2.3.4 评审测试用例（2）

- 是否每个测试用例（或相关的测试用例）都确定了初始的测试目标状态和测试数据状态？
- 测试用例是否包含了所有的单一边界？
- 测试用例是否包含了所有的业务数据流？
- 是否所有的测试用例名称，ID名称都与测试工件命名规约一致？

■ 参加人员

- 项目经理、系统分析员、测试设计员、测试员

2.3.5 跟踪测试用例

- 需求管理

- 需求→测试用例

- 测试用例是否覆盖了需求

- 需求→测试需求→测试用例

- 测试用例执行率、通过率

- 测试用例→测试用例执行结果

3. 测试用例设计技术

- 3.1 黑盒测试用例设计技术
- 3.2 白盒测试用例设计技术
- 3.3 面向对象测试用例设计技术

3.1 黑盒测试用例设计技术

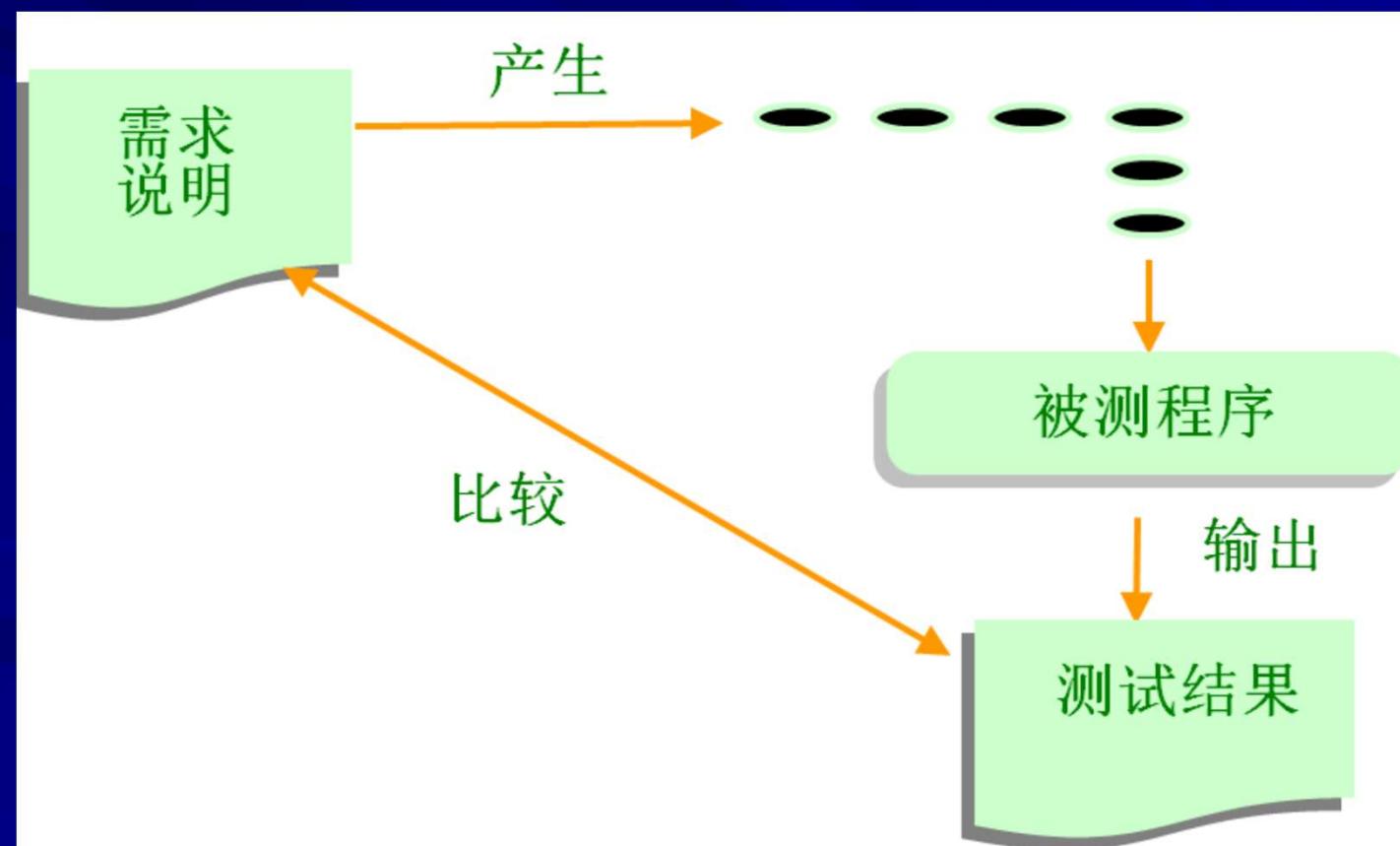
- 3.1.1 黑盒测试的概念
- 3.1.2 黑盒测试的作用
- 3.1.3 黑盒测试的局限
- 3.1.4 静态黑盒测试
- 3.1.5 动态黑盒测试
- 3.1.6 黑盒测试用例设计技术
- 3.1.7 结论

3.1.1 黑盒测试的概念（1）

- 黑盒测试是在程序接口进行的测试，它只检查程序功能是否能按照规格说明书的规定正常使用，程序是否能适当地接收输入数据产生正确的输出信息，并且保持外部信息的完整性

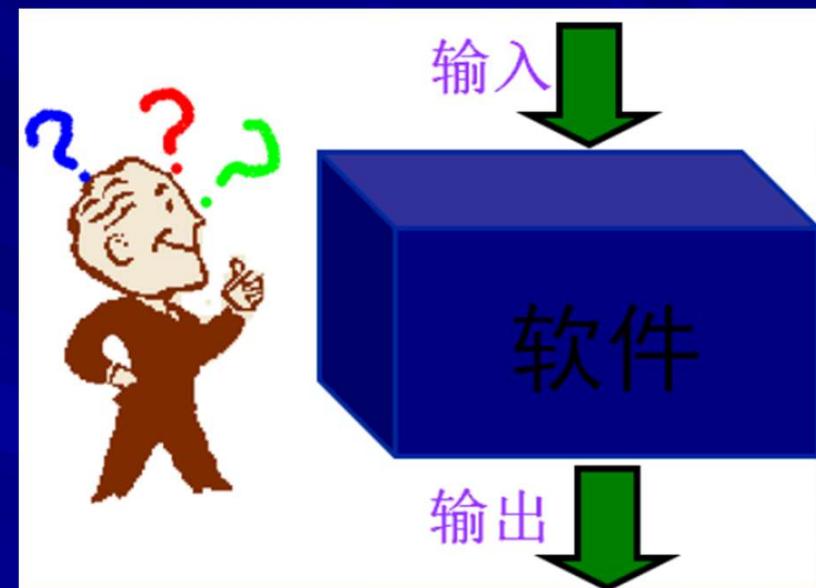
3.1.1 黑盒测试的概念（2）

- 黑盒测试依赖于测试环境下应用的需求说明或功能描述



3.1.1 黑盒测试的概念（3）

- 黑盒测试法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
- 在黑盒测试中，软件测试员只需要知道软件要做什么即可——而无法看到盒子中是如何运做的



3.1.2 黑盒测试的作用

■ 黑盒测试主要是为了发现以下几类错误

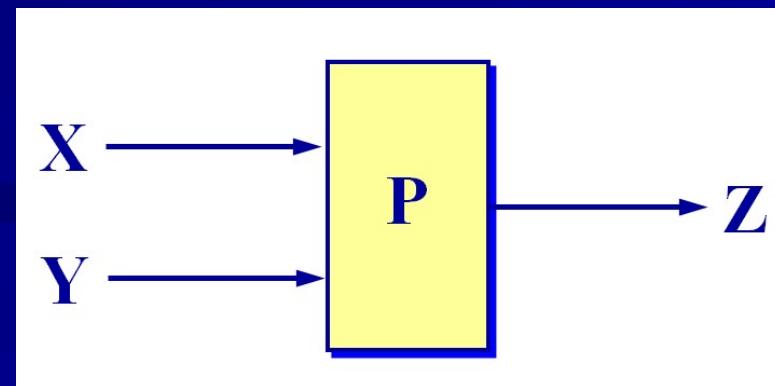
- 是否有不正确或遗漏了的功能？
- 在接口上，输入是否正确地接受？是否输出正确的结果？
- 是否有数据结构错误或外部信息（例如数据文件）访问错误？
- 性能上是否能够满足要求？
- 是否有初始化或终止性错误？

3.1.3 黑盒测试的局限（1）

- 用黑盒测试发现程序中的错误，必须在所有可能的输入条件和输出条件下确定测试数据，来检查程序是否都能产生正确的输出，但这是不可能的

示例

- 假设一个程序P有输入量X和Y及输出量Z。在字长为32位的计算机上运行。若X、Y取整数，按黑盒方法进行穷举测试
 - 可能采用的测试数据组： $2^{32} \times 2^{32} = 2^{64}$
 - 如果测试一组数据需要1毫秒，一年工作 365×24 小时，完成所有测试需5亿年



3.1.3 黑盒测试的局限（2）

- 一些外购软件、参数化软件包，由于无法得到源程序，只能采用黑盒测试方法进行检查
- 黑盒测试的测试数据是根据需求规格说明来决定的，但需求规格说明本身也不见得完全正确，如在需求规格说明中规定了多余的功能或是遗漏掉了某些功能，这些问题对于黑盒测试来说是查不出来的
- 第三方测试大多采用黑盒测试方法

3.1.4 静态黑盒测试

- 测试产品说明书属于静态黑盒测试。产品说明书是书面文档，而不是可执行程序，因此是静态的
- 无论针对何种形式的产品说明书，都可以利用静态黑盒技术进行测试
- 通过询问软件的设计者和编制者甚至可以测试没有写出来的产品说明书

对产品说明书进行高级审查

■ 测试产品说明书的第一步不是钻进去找软件缺陷，而是在一个高度上审视。审查产品说明书是为了找出**根本性**的大问题、疏漏或遗漏之处。也许这更像是研究而不是测试，但是研究主要是为了更好地了解软件要做什么

设身处地为客户着想

- 软件测试员第一次接到需要审查的产品说明书时，最容易做的事就是把自己当作客户。软件测试员必须了解被测试软件是否符合客户的要求
- 此时，熟悉软件应用领域的相关知识极有好处
 - 先进数通金融软件

研究现有的标准和规范

- 软件测试员要做的是“检验”软件产品中是否套用了正确的标准和规范，有无遗漏
- 一些可以做为标准和规范的例子
 - 公司惯用语和约定
 - 行业要求
 - 国家标准
 - 图形用户界面（GUI）
 - 硬件和网络标准

审查和测试同类软件（1）

- 了解软件最终结果的最佳方法是研究同类软件，例如竞争对手的产品或者小组开发的其他类似产品
 - iOS VS Android

审查和测试同类软件（2）

■ 在审查竞争产品时要注意的问题包括

- 规模
- 复杂性
- 测试性
- 质量/可靠性

产品说明书的低级测试技术

■ 产品说明书属性检查清单

— 优秀的产品说明书应具有的8个重要属性

- 完整
- 准确
- 精确
- 一致
- 贴切
- 合理
- 代码无关
- 可测试

产品说明书用语检查清单

■ 在审查产品说明书时，还要检查对问题的描述

- 总是、每一种、所有、没有、从不
- 当然、因此、明显、显然、必然
- 某些、有时、常常、通常、惯常、经常、大多、几乎
- 等等、诸如此类、依此类推
- 良好、迅速、廉价、高效、小、稳定
- 已处理、已拒绝、已忽略、已消除
- 如果...那么...（没有否则）

3.1.5 动态黑盒测试

- 不深入代码细节的软件测试方法称为动态黑盒测试（常被称为行为测试）。它是动态的，因为程序正在运行；它是黑盒子，因为测试员不知道程序如何工作

3.1.6 黑盒测试用例设计技术

- 3.1.6.1 等价类划分
- 3.1.6.2 边界值分析
- 3.1.6.3 错误猜测法
- 3.1.6.4 判定表法
- 3.1.6.5 因果图法

3.1.6.1 等价类划分

- A. 问题引入
- B. 等价类的定义
- C. 等价类的特点
- D. 等价类的划分
- E. 等价类测试的基本思想
- F. 等价类划分方法设计测试用例

A. 问题引入 (1)

■ 测试的矛盾

- 为保证软件产品的质量，我们需要进行完备（彻底）的测试；但这是不现实的，因为
 - 输入量太大
 - 输出结果太多
 - 软件实现途径太多



A. 问题引入 (2)

■ 测试的矛盾 (续)

— 从经济的角度来说，我们希望测试没有冗余

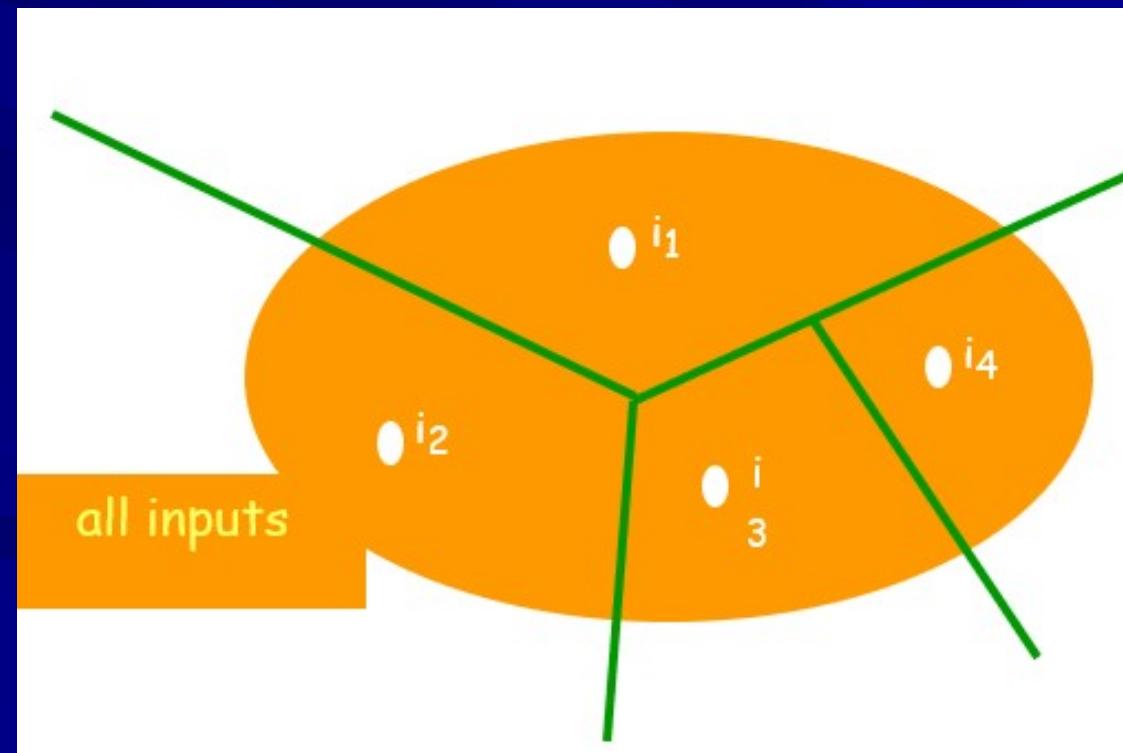
- 一个好的测试用例在于能发现至今未发现的错误
- 一个成功的测试是发现了至今未发现的错误的测试

■ 等价类方法可令测试事半功倍



B. 等价类的定义

- 等价类是输入域的某个子集合，而所有的等价类的并集是整个输入域。在子集合中，各个输入数据对于揭露程序中的错误是等效的



C. 等价类的特点

- 完备性
- 无冗余性
- 等价性

D. 等价类的划分

■ 划分

- 划分是指互不相交的一组子集，这些子集的并是整个集合

■ 给定集合B，以及B的一组子集A₁、A₂.....、A_n，这些子集是B的一个划分，当且仅当：

$$A_1 \cup A_2 \cup \dots \cup A_n = B, \text{ 且 } i \neq j \quad A_i \cap A_j = \emptyset$$

■ 划分对于测试的作用

- 完备性
- 无冗余性

E. 等价类测试的基本思想

■ 等价类测试的假设

- 测试某等价类的代表值，就等效于对这个等价类中其他值的测试

■ 等价类测试的思想

- 把全部的输入数据划分成若干个等价类，在每一个等价类中取一个数据来进行测试

F. 等价类划分方法设计测试用例 (1)

- 使用等价划分方法设计测试用例要经历划分等价类（列出等价类表）和选取测试用例两步
 - 划分等价类（EC）
 - 等价类是指某个输入域的子集合。在该子集合中，各个输入数据对于揭露程序中的错误都是等效的。测试某等价类的代表值就等价于对这一类其它值的测试

F. 等价类划分方法设计测试用例 (2)

等价类的划分有两种不同的情况：

- ① 有效等价类：是指对于程序的规格说明来说，是合理的，有意义的输入数据构成的集合
- ② 无效等价类：是指对于程序的规格说明来说，是不合理的，无意义的输入数据构成的集合

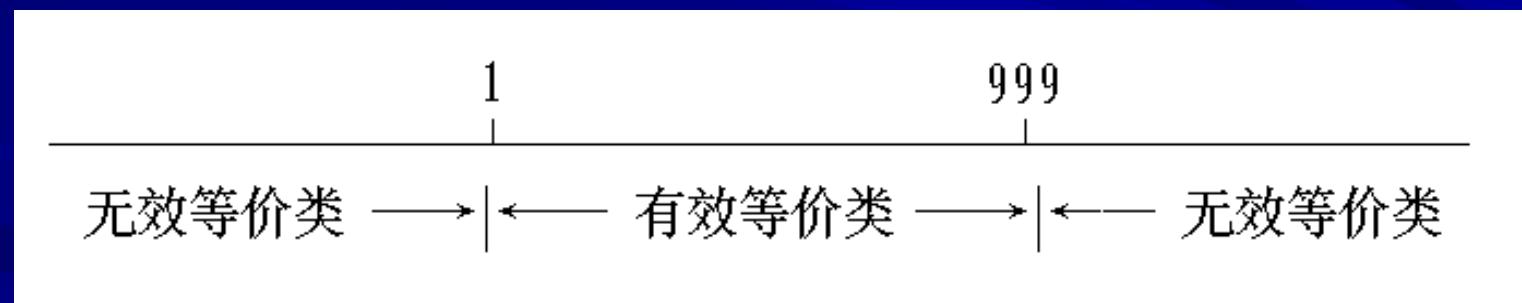
在设计测试用例时，要同时考虑有效等价类和无效等价类的设计

F. 等价类划分方法设计测试用例 (3)

划分等价类的原则：

(1) 如果输入条件规定了取值范围, 或值的个数, 则可以确立一个有效等价类和两个无效等价类。

例如，在程序的规格说明中，对输入条件有一句话：“.....项数可以从1到999.....”，则有效等价类是“ $1 \leq \text{项数} \leq 999$ ”，两个无效等价类是“项数 <1 ”或“项数 >999 ”。在数轴上表示成



F. 等价类划分方法设计测试用例 (4)

(2) 如果输入条件规定了输入值的集合，或者是规定了“必须如何”的条件，这时可确立一个有效等价类和一个无效等价类

例如，如果输入的第一个符号必须是数字，则定义一个有效等价类，第一个符号是数字；定义一个无效等价类，第一个符号不是数字

(3) 如果输入条件是一个布尔量，则可以确定一个有效等价类和一个无效等价类

例如，如果输入需要TOM、DICK或HAPPY这些名字之一，那么定义一个有效等价类（采用有效名字之一）和一个无效等价类（采用JOE这个名字）

F. 等价类划分方法设计测试用例 (5)

(4) 如果规定了输入数据的一组值，而且程序要对每个输入值分别进行处理。这时可为每一个输入值确立一个有效等价类，此外确定一个无效等价类，它是所有不允许的输入值的集合

例如，在教师上岗方案中规定对教授、副教授、讲师和助教分别计算分数，做相应的处理。因此可以确定4个有效等价类为教授、副教授、讲师和助教；一个无效等价类，它是所有不符合以上身份的人员的输入值的集合

F. 等价类划分方法设计测试用例 (6)

(5) 如果规定了输入数据必须遵守的规则，则可以确立一个有效等价类（符合规则）和若干个无效等价类（从不同角度违反规则）

例如，Pascal语言规定“一个语句必须以分号‘；’结束”。这时，可以确定一个有效等价类“以‘；’结束”，若干个无效等价类“以‘：’结束”、“以‘，’结束”、“以‘ ’结束”、“以LF结束”等

F. 等价类划分方法设计测试用例 (7)

– 选取测试用例

1. 在确立了等价类之后，建立等价类表，列出所有划分出的等价类

输入条件	有效等价类	无效等价类
.....
.....

F. 等价类划分方法设计测试用例 (8)

2. 再从划分出的等价类中按以下原则选择测试用例
 - 1) 为每一个等价类规定一个唯一编号
 - 2) 设计一个新的测试用例，使其尽可能多地覆盖尚未被覆盖的有效等价类，重复这一步，直到所有的有效等价类都被覆盖为止
 - 3) 设计一个新的测试用例，使其仅覆盖一个尚未被覆盖的无效等价类，重复这一步，直到所有的无效等价类都被覆盖为止

实例（1）

■ 在某一Pascal语言版本中规定

- “标识符是由字母开头，后跟字母或数字的任意组合构成。编译器能够区分的有效字符数为 8 个，最大字符数为 80 个。” 并且规定：“标识符必须先说明，再使用。” “在同一说明语句中，标识符至少必须有一个。”

Thank You