SoftTest 北京软测

软件测试基础

测知识回顾-测试用例设计

主讲人：

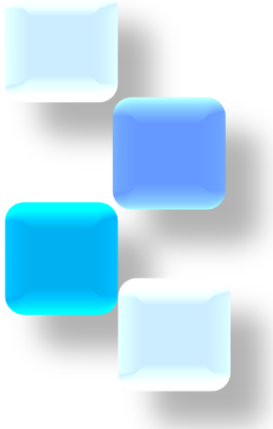





上节课程回顾

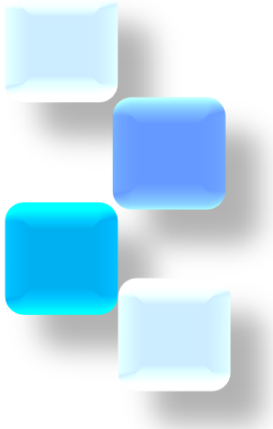
本节课程目标

测试用例设计



测试用例概述
测试用例的组成元素与范例
白盒测试用例设计方法
黑盒测试用例设计方法
测试类型与测试用例设计
设计测试用例的策略选择
测试用例设计工具
测试用例编写
案例研究：测试用例设计

测试用例设计



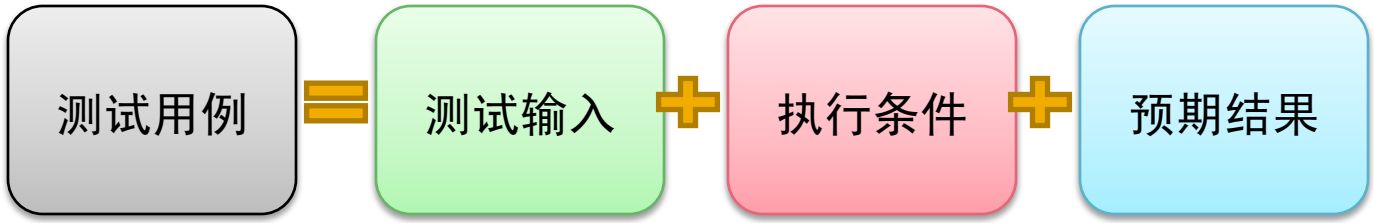
测试用例概述
测试用例的组成元素与范例
白盒测试用例设计方法
黑盒测试用例设计方法
测试类型与测试用例设计
设计测试用例的策略选择
测试用例设计工具
测试用例编写
案例研究：测试用例设计

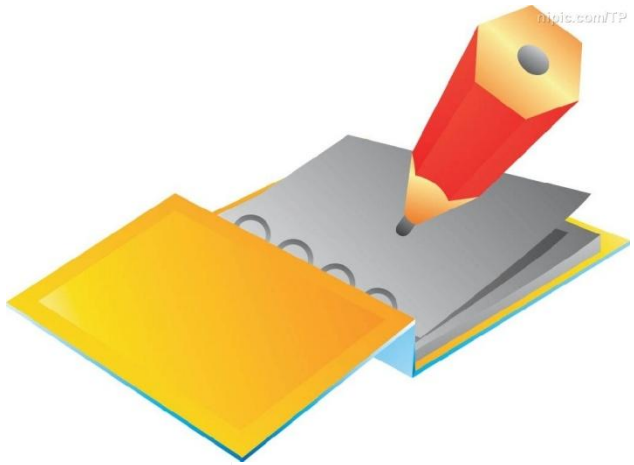
测试用例概述

- 如何以最少的人力、资源投入，在最短的时间内完成测试，发现软件系统的缺陷，保证软件的优良品质，是软件公司探索和追求的目标。
- 测试用例是测试工作的指导，是软件测试的必须遵守的准则。更是软件测试质量稳定的根本保障。

什么是测试用例？

- 测试用例的定义
  - 测试用例是为特定目标开发的测试输入、执行条件和预期结果的集合。





为什么使用测试用例（1）

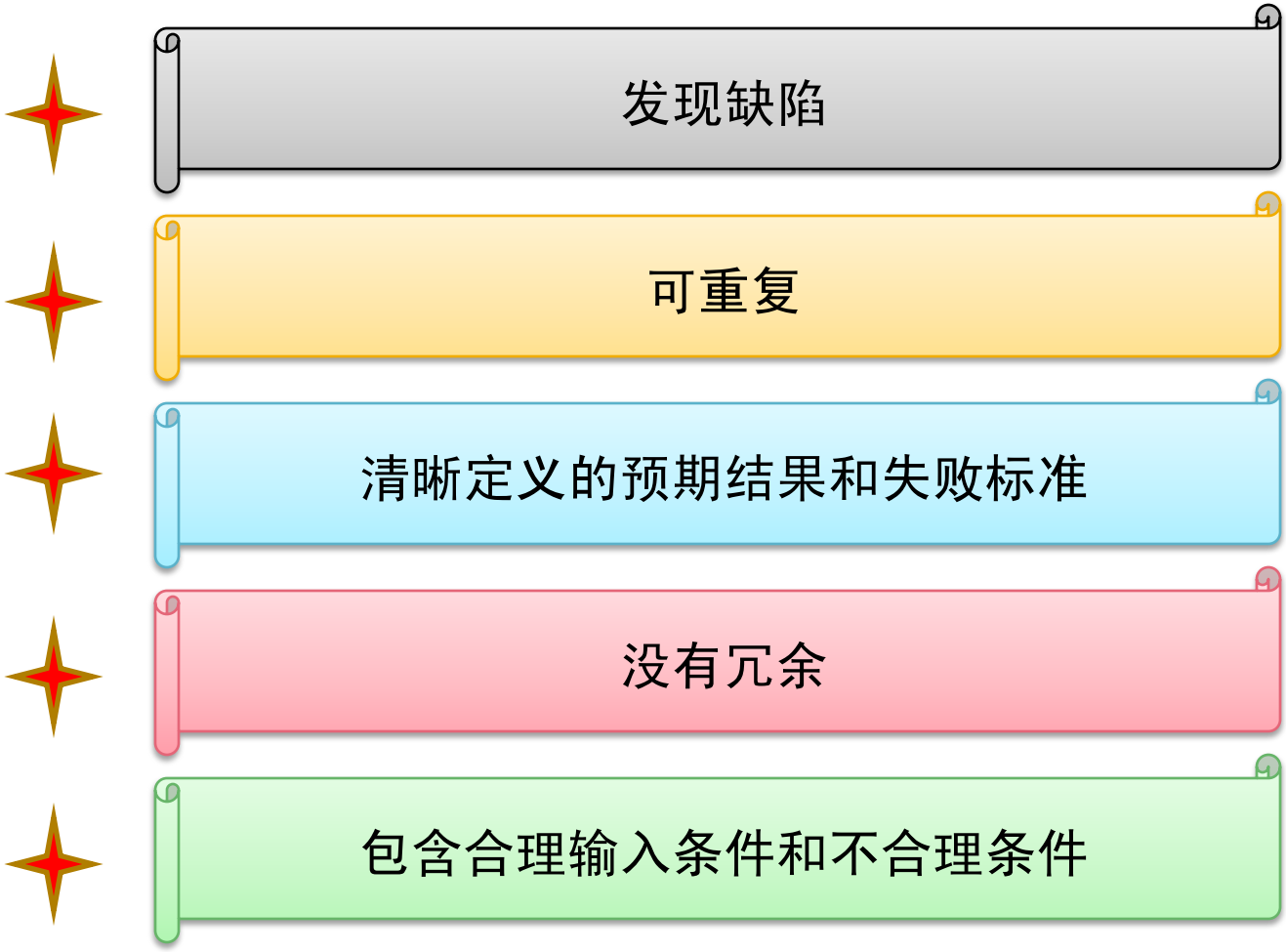
- 在开始实施测试之前设计好测试用例，避免盲目测试并提高测试效率，减少测试的不完全性；
- 测试用例的使用令软件测试的实施重点突出、目的明确；
- 根据测试用例的多少和执行难度，估算测试工作量，便于测试项目的时间和资源管理与跟踪；
- 减少回归测试的复杂程度；
- 在软件版本更新后只需修正少量的测试用例便可展开测试工作，降低工作强度、缩短项目周期；
- 功能模块的测试用例的通用化和复用化则会使软件测试易于开展，并随着测试用例的不断细化其效率也不断攀升；

为什么使用测试用例（2）

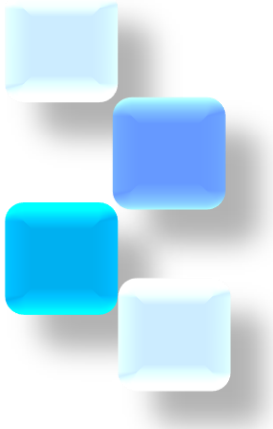
- 根据测试用例的操作步骤和执行结果，可以方便地书写软件测试缺陷报告；
- 可以根据测试用例的执行等级，实施不同级别的测试；
- 为分析软件缺陷和程序模块质量提供依据；
- 便于大型软件测试项目外包测试指导基础；
- **总结：**
  - 软件测试是有组织性、步骤性和计划性的，为了能将软件测试的行为转换为可管理的、具体量化的模式，需要创建和维护测试用例。

好测试用例的特点

- 好的测试用例



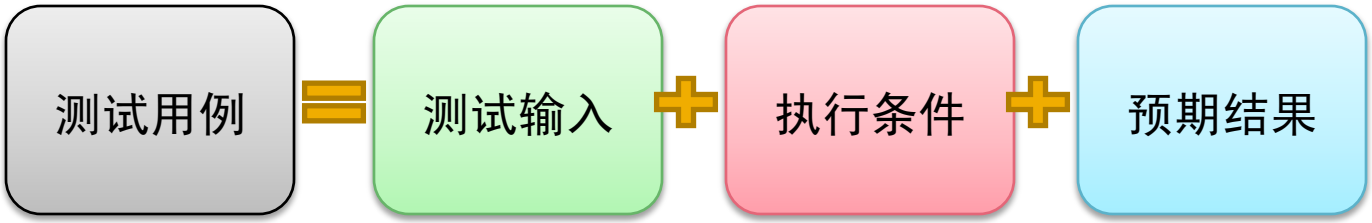
测试用例设计



测试用例概述
测试用例的组成元素与范例
白盒测试用例设计方法
黑盒测试用例设计方法
测试类型与测试用例设计
设计测试用例的策略选择
测试用例设计工具
测试用例编写
案例研究：测试用例设计

测试用例的组成元素与范例

- 测试用例组成



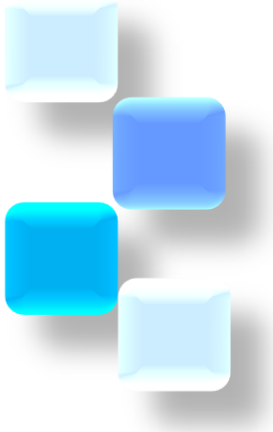
测试用例的组成元素与范例

- 测试用例编号ID
- 测试用例标题
- 测试的模块

- 测试输入条件
- 期望的输出结果
- 其它说明

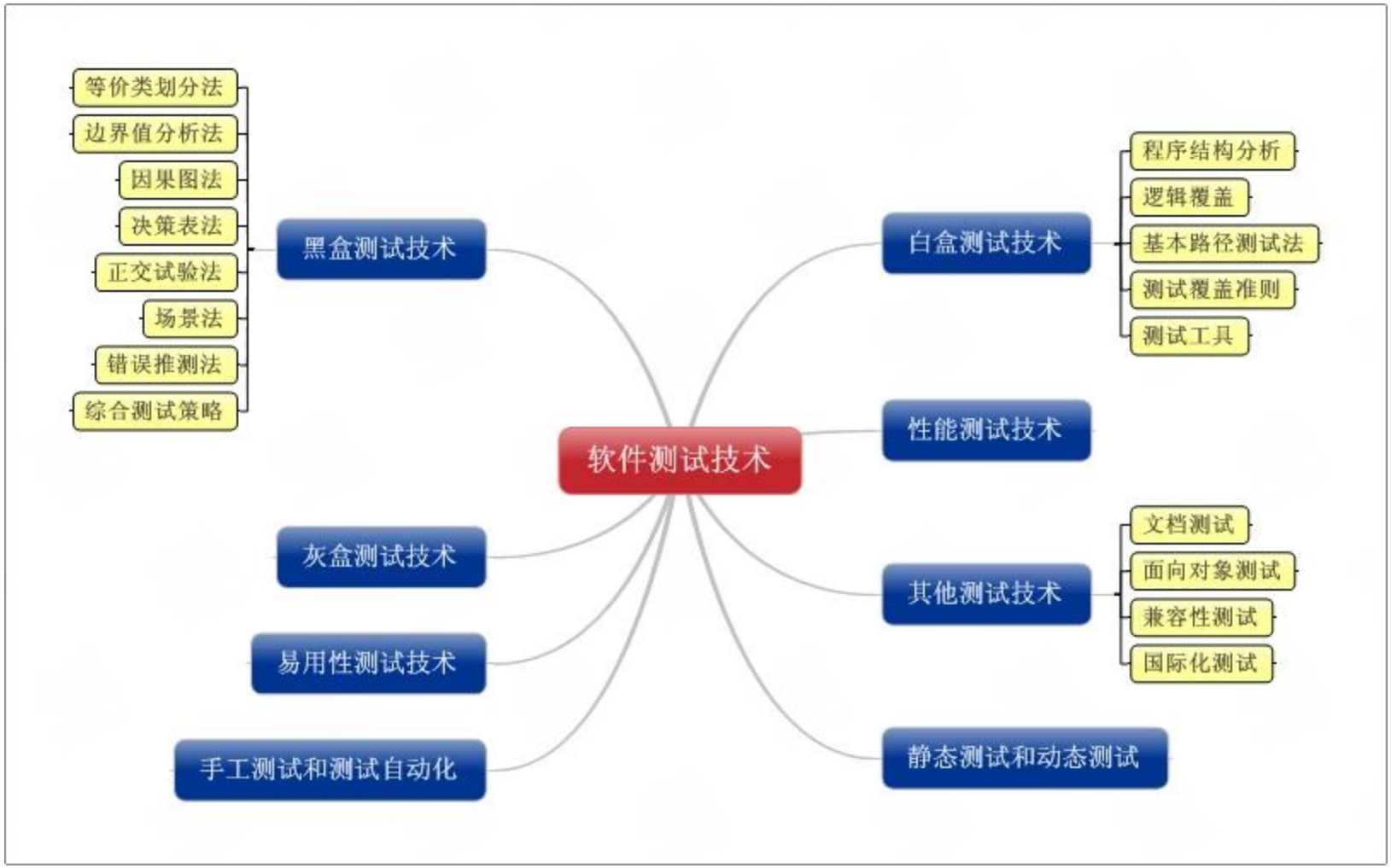
ID	类型	标题	测试步骤	期望的结果	说明
001	登录	输入正确密码	用户在登录界面输入正确的密码“***”后，按回车键	程序提示登录成功	
002	登录	输入错误密码	用户在登录界面输入错误的密码“***”后，按回车键	程序提示输入密码错误，请重新输入	
003	登录	不输入的空密码	用户在登录界面没有输入任何密码使密码为空后，按回车键	程序提示用户没有输入密码，请输入	程序应该告知用户没有输入密码，而不是密码错误

测试用例设计



测试用例概述
测试用例的组成元素与范例
白盒测试用例设计方法
黑盒测试用例设计方法
测试类型与测试用例设计
设计测试用例的策略选择
测试用例设计工具
测试用例编写
案例研究：测试用例设计

软件测试技术简介



白盒测试用例设计方法

- 什么是白盒测试
  - 白盒测试也称为结构测试，把程序看作一个透明的盒子，测试程序的代码书写结构和逻辑问题

白盒测试用例设计方法

- 白盒测试用例的设计方法
  - 逻辑覆盖
  - 基本路径测试

逻辑覆盖

- 为了衡量测试的覆盖程度，需要建立一些标准，目前常用的一些覆盖标准从低到高分别是：

语句覆盖	判定覆盖	条件覆盖
判定/条件覆盖	条件组合覆盖	路径覆盖

逻辑覆盖-语句覆盖

- “语句覆盖”是一个比较弱的测试标准，它的含义是：选择足够的测试用例，使得程序中每个语句至少都能被执行一次。

逻辑覆盖– 分支覆盖

- “分支覆盖”(或称判定覆盖)标准的含义是：执行足够的测试用例，使得程序中的每一个分支至少都通过一次。

逻辑覆盖– 条件覆盖

- “条件覆盖”的含义是：执行足够的测试用例，使得判定中的每个条件获得各种可能的结果。

逻辑覆盖- 分支 / 条件覆盖

- “分支 / 条件覆盖” 的含义是：执行足够的测试用例，使得分支中每个条件取到各种可能的值，并使每个分支取到各种可能的结果

逻辑覆盖-条件组合覆盖

- “条件组合覆盖”的含义是：执行足够的例子，使得每个判定中条件的各种可能组合都至少出现一次。
- 满足“条件组合覆盖”的测试用例是一定满足“分支覆盖”、“条件覆盖”和“分支/条件覆盖”的。

逻辑覆盖-路径测试

- 路径测试就是设计足够多的测试用例，覆盖被测试对象中的所有可能路径。

基本路径测试法-概念

- 基本路径测试：在程序控制流图的基础上，通过计算环路复杂度，导出基本可执行路径的集合，然后依据此设计测试用例。设计出的测试用例要保证程序的每个可执行语句至少执行一次

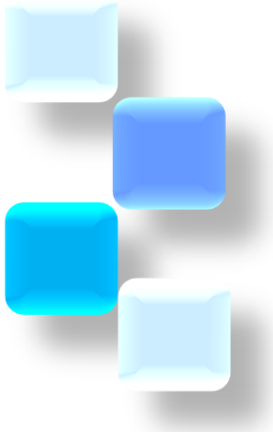
基本路径测试法-步骤

- 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。包括以下4个步骤：

白盒测试用例设计方法

- 白盒测试用例注意事项
  - 由于测试路径可能非常多，由于时间和资源问题，选出足够多的路径测试
  - 由于深入到程序编码，通常开发人员协助测试人员书写白盒测试用例

测试用例设计



测试用例概述
测试用例的组成元素与范例
白盒测试用例设计方法
黑盒测试用例设计方法
测试类型与测试用例设计
设计测试用例的策略选择
测试用例设计工具
测试用例编写
案例研究：黑盒测试用例设计

黑盒测试用例设计方法

- 什么是黑盒测试
  - 黑盒测试被称为功能测试或数据驱动测试，是针对软件的功能需求进行测试。在测试时，把被测程序视为一个不能打开的黑盒子，在完全不考虑程序内部结构和内部特性的情况下进行。

黑盒测试技术内容

等价类测试用例设计方法

- 黑盒测试用例的设计方法
  - 等价类划分：
    - 在分析需求规格说明的基础上划分等价类，列出等价类表。
    - 将程序可能的输入数据分成若干个子集，从每个子集选取一个代表性的数据作为测试用例，等价类是某个输入域的子集，在该子集中每个输入数据的作用是等效的。
    - 等价类的分类：有效等价类和无效等价类。有效等价类是有意义的、合理的输入数据，可以检查程序是否实现了规格说明中所规定的功能和性能。无效等价类与有效等价类的意义相反。
    - 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。这样的测试才能确保软件具有更高的可靠性。

确定等价类的方式

- 在输入条件规定了取值范围或值的个数的情况下，则可以确立一个有效等价类和两个无效等价类。
- 在输入条件规定了输入值的集合或者规定了“必须如何”的条件的情况下，可以确立一个有效等价类和一个无效等价类。
- 在输入条件是一个布尔量的情况下，可确定一个有效等价类和一个无效等价类。
- 在规定了输入数据的一组值(假定n个)，并且程序要对每一个输入值分别处理的情况下，可确立n个有效等价类和一个无效等价类。
- 在规定了输入数据必须遵守的规则的情况下，可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)。

根据等价类创建测试用例的步骤

- 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
…	…	…
…	…	…

- 为每个等价类规定一个唯一的编号；
- 设计一个新的测试用例，使其尽可能多地覆盖尚未覆盖的有效等价类。重复这一步，最后使得所有有效等价类均被测试用例所覆盖；
- 设计一个新的测试用例，使其只覆盖一个无效等价类。重复这一步使所有无效等价类均被覆盖。

边界值测试用例设计方法

- 边界值分析法：
  - 程序的很多错误发生在输入或输出范围的边界上，因此针对各种边界情况设置测试用例，可以发现不少程序缺陷。
  - 设计方法：
    - 确定边界情况（输入或输出等价类的边界）
    - 选取正好等于、刚刚大于或刚刚小于边界值作为测试数据

确定边界值的方式

- 如果输入条件规定了值的范围，则应取刚达到这个范围的边界的值，以及刚刚超越这个范围边界的值作为测试输入数据。
- 如果输入条件规定了值的个数，则用最大个数、最小个数、比最小个数少一、比最大个数多一的数作为测试数据。
- 如果程序的规格说明给出的输入域或输出域是有序集合，则应选取集合的第一个元素和最后一个元素作为测试用例。
- 如果程序中使用了一个内部数据结构，则应当选择这个内部数据结构的边界上的值作为测试用例。
- 分析规格说明，找出其他可能的边界条件。

错误推测法测试用例设计

- 基于经验和直觉推测程序中所有可能存在的各种错误，从而有针对性地设计测试用例。
- 发现程序经常出现的错误的方法：
  - 单元测试中发现的模块错误；
  - 产品的以前版本曾经发现的错误；
  - 输入数据为0或字符为空；
  - 当软件要求输入时(比如在文本框中),不是没有输入正确的信息，而是根本没有输入任何内容，单单按了Enter键；
  - 这种情况在产品说明书中常常忽视，程序员也可能经常遗忘，但是在实际使用中却时有发生。程序员总会习惯性的认为用户要么输入信息，不管是看起来合法的或非法的信息，要不就会选择Cancel键放弃输入，

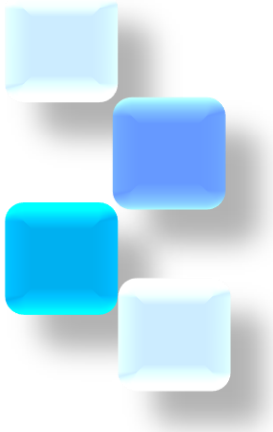
测试场景法设计测试用例

- 现在的软件几乎都是用事件触发来控制流程的，事件触发时的情景便形成了场景，而同一事件不同的触发顺序和处理结果就形成事件流。这种在软件设计方面的思想也可引入到软件测试中，可以比较生动地描绘出事件触发时的情景，有利于测试设计者设计测试用例，同时使测试用例更容易理解和执行。

因果图法测试用例设计

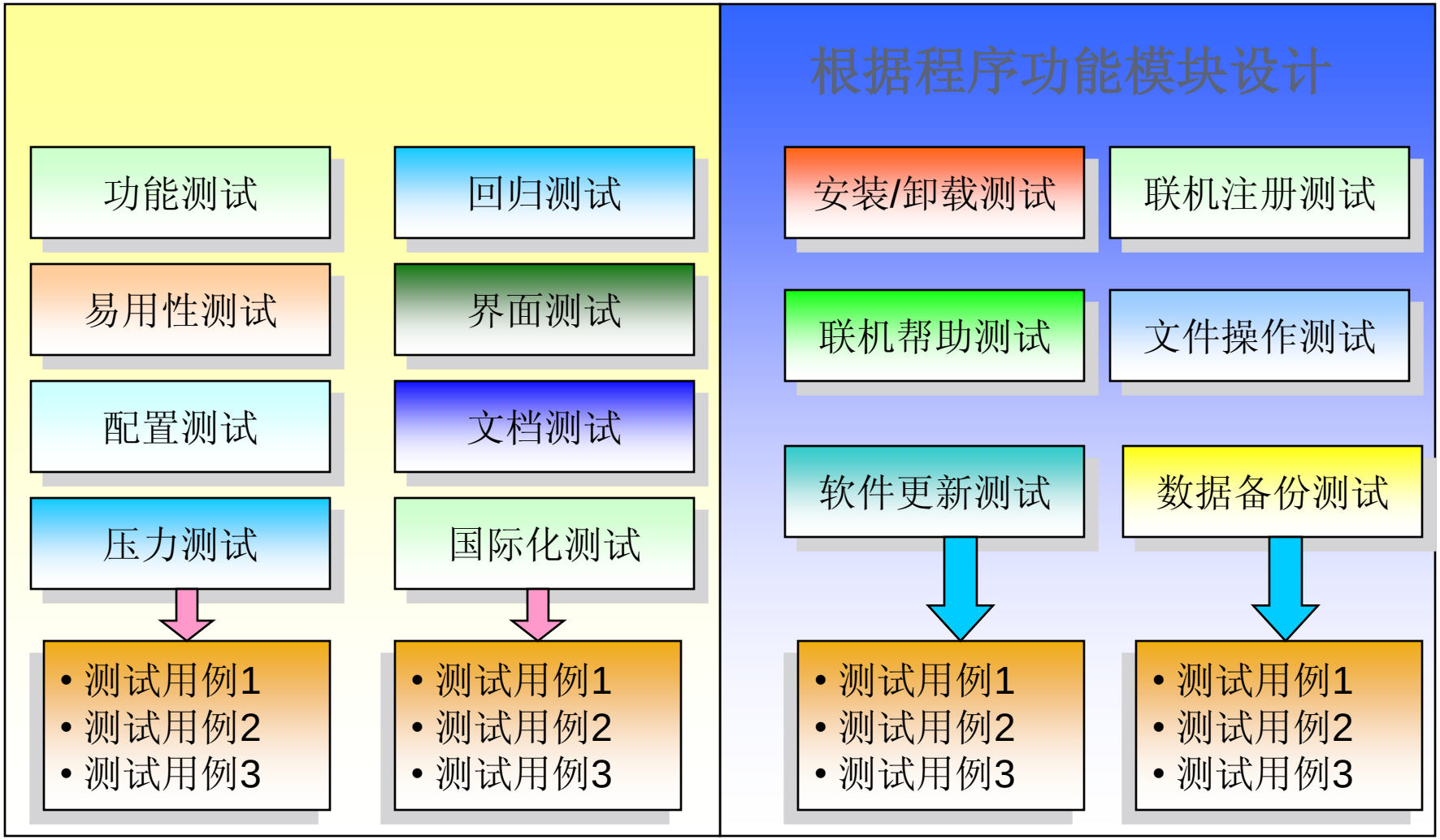
- 多种输入条件的组合，产生多种结果设计测试用例。
- 设计方法：
  - 分析软件规格说明文档描述的哪些是原因（输入条件），哪些是结果（输出条件），给每个原因和结果赋予一个标示符。
  - 找出原因与结果，原因与原因之间的对应关系，划出因果图
  - 在因果图上标上哪些不可能发生的因果关系，表明约束或限制条件
  - 根据因果图，创建判定表，将复杂的逻辑关系和多种条件组合很具体明确的表示出来
  - 把判定表的每一行作为依据设计测试用例。

测试用例设计

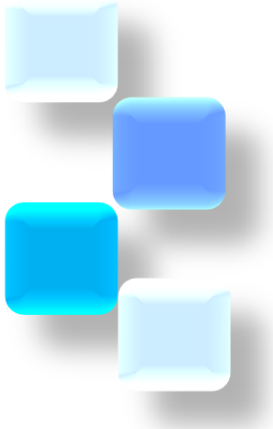


测试用例概述
测试用例的组成元素与范例
白盒测试用例设计方法
黑盒测试用例设计方法
测试类型与测试用例设计
设计测试用例的策略选择
测试用例设计工具
测试用例编写
案例研究：黑盒测试用例设计

测试类型与测试用例设计



测试用例设计



测试用例概述
测试用例的组成元素与范例
白盒测试用例设计方法
黑盒测试用例设计方法
测试类型与测试用例设计
设计测试用例的策略选择
测试用例设计工具
测试用例编写
案例研究：测试用例设计

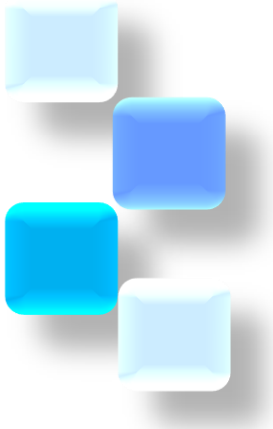
设计测试用例的策略选择

- 测试用例的设计方法不是单独存在的，具体到每个测试项目里都会用到多种方法，每种类型的软件有各自的特点，每种测试用例设计的方法也有各自的特点，针对不同软件如何利用这些黑盒方法是非常重要的。
- 在实际测试中，往往是综合使用各种方法才能有效提高测试效率和测试覆盖度，这就需要认真掌握这些方法的原理，积累更多的测试经验，以有效提高测试水平。
- 首先进行等价类划分，包括输入条件和输出条件的等价划分，将无限测试变成有限测试，这是减少工作量和提高测试效率的最有效方法。

设计测试用例的策略选择

- 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。
- 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。
- 对于业务流清晰的系统，可以利用场景法贯穿整个测试案例过程，在案例中综合使用各种测试方法。

测试用例设计



测试用例概述
测试用例的组成元素与范例
白盒测试用例设计方法
黑盒测试用例设计方法
测试类型与测试用例设计
设计测试用例的策略选择
测试用例设计工具
测试用例编写
案例研究：测试用例设计

测试用例设计工具

设计依据：

- 软件需求文档
- 软件设计文档

常见问题：

- 软件文档不全或没有文档
- 没有完成编码就开始设计测试用例

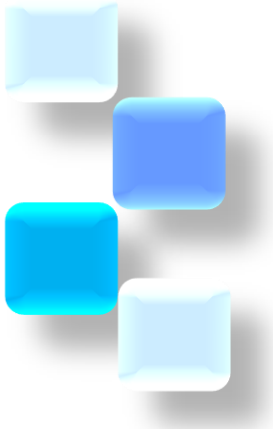
通用设计工具：

- Microsoft Word
- Microsoft Excel
- Microsoft Access

设计工具专用：

- IBM Rational TestManager
- Mercury Interactive TestDirector

测试用例设计



测试用例概述
测试用例的组成元素与范例
白盒测试用例设计方法
黑盒测试用例设计方法
测试类型与测试用例设计
设计测试用例的策略选择
测试用例设计工具
测试用例编写
案例研究：测试用例设计

测试用例编写规范（1）

1. 对于每个功能，从类型1至类型N依次撰写相应用例
2. 对于不满足要求的非常规类型，可以不写相应的用例
3. 对于边界、空值、格式错误、溢出这几个类型，一个功能如有多个数据项测试类型相同，则可以放在一个用例里
4. 测试用例均为最小的用例覆盖要求；对于没有提及的用例类型，视业务需求情况，撰写相应用例
5. 在测试过程中，输入数据可在测试用例规定的范围内做一定变化
6. 对于每个功能点，必须通过一组（一个或多个）用例满足其业务覆盖：对于某条记录的每个状态，对于能进行的每个操作，都生成一个用例（即对业务功能流程中的每个角色，每个功能操作，生成一个用例）

测试用例编写规范（2）

6.1 常规的测试用例：

- 1. 对于一个功能一个模块（页面），每个数据项输入或选中典型的取值，生成一个用例
- 2. 对于一个功能多个模块（页面），多个模块（页面）一起生成一个用例
- 3. 对于多个功能一个模块（页面），每个功能生成一个用例
- 4. 每个功能操作需覆盖，如删除对话框点击确定、取消分别生成2个用例步骤。
- 5. 输入框测试，在允许范围内尽可能覆盖多的字符类别，如中文、英文、数字等
- 6. 对于每个功能点，必须通过一组（一个或多个）用例满足其业务覆盖：对于某条记录的每个状态，对于能进行的每个操作，都生成一个用例（即对业务功能流程中的每个角色，每个功能操作，生成一个用例）

测试用例编写规范（3）

6.2 初始化的测试用例：

- 进入功能模块（页面）后，某些控件会初始化填入数据，生成一个用例确保所有的初始数据正确

测试用例编写规范（4）

6.3 边界的测试用例

- 1. 每个数据项，生成一个边界用例（含最大、最小两个边界值）
- 2. 字符串数据以字符串长度为计量单位
- 3. 布尔值数据的所有取值都需测试
- 4. 多个复选框一组时，需测同时都被选中及都不被选中
- 5. 下拉菜单、列表框、单选按钮组为最大、最小的2个取值

测试用例编写规范（5）

6.4 空值的测试用例：

- 对于每个必填数据项，都生成一个用例（不提供空值的除外，比如无空值的下拉框、有缺省值的单选按钮组），则预期结果提示该数据项为空

测试用例编写规范（6）

6.5　格式错误的测试用例：

- 1.对于输入框数据项，都生成一个用例，预期结果提示该数据项格式错误
- 2.日期输入框
- 3.数字输入框
- 4.字符串输入框：Email、邮编、用户名等带格式要求的

测试用例编写规范（7）

6.6 溢出的测试用例：

- 1.对于输入框数据项，都生成一个取值范围外的测试用例，预期结果提示该数据项超出范围日期输入框
- 2.范围的日期输入框，需添加上边界日期小于下边界日期的用例
- 3.数字输入框（如‘金额’一般为正整数，填入一个负数）
- 4.字符串输入框：超出规定长度的字符串

测试用例编写规范（8）

6.7 关联的测试用例：

- 对于相互关联的两个或多个数据项，生成一个用例，确保当一个数据项改变时，其他数据项的变化正确

测试用例编写规范（9）

6.8　唯一值的测试用例：

- 某些业务的数据字段要求是唯一的，生成一或两个用例（新建、编辑），使得输入数据与原有数据在该字段重复，预期结果为页面返回该数据已存在的提示

测试用例编写规范（10）

6.9　权限不足的测试用例：

- 对于功能模块，生成一个用例，以没有权限的用户身份访问，预期结果为提示权限不足

测试用例编写规范（11）

6.10　角色权限的测试用例：

- 业务功能流程涉及一到多个角色，对于每个角色，都生成一个用例，预期结果为用户以这个角色登陆时，他仅能执行权限允许的操作

测试用例编写细则-命名细则

- 由于项目的实际需求和测试的工作需要，分以下几个等级来规范测试用例的命名
  - 1. 一级目录使用各项目的顶级菜单名称来命名，如维护、业务、查询三大类；
  - 2. 二级目录使用顶级菜单下的二级菜单名称类命名，用户可根据名字判别该用例是测试哪个模块的；
  - 3. 各用例根据各用例的功能来命名，尽量做到简洁明了。同一个目录下的用例名字字数最好相同；

测试用例编写细则-编号规则

- 每个测试用例都有自己唯一的编号。根据工作的实际需要，我们规定在每个用例名称前面必须写上用例编号，用例编号的定义分以下几大类：
  - 1、根据需求编写测试用例：

  需求编号＋用例一级目录号＋用例二级目录号＋用例号

  R001 ＋ 01 ＋ 01 01

  - 2、根据功能编写测试用例：

  用例一级目录号＋用例二级目录号＋用例号

  F001 ＋ 001 ＋ 001

测试用例编写细则-用例划分

- 在编写测试用例时，我们会根据系统模块的具体情况从不同的角度去考虑测试用例的编写，有些是通过操作步骤来编写，有些则是根据功能条件来编写，更有可能是根据测试目的来编写，为了区分这些用例，我们规定在每种用例前写上对应的编码。具体见下表：

需求	功能	业务	性能
R(Requirement)	F(Function)	B(Business)	P(Performance)

测试用例编写方法-编写前准备

- 测试用例编写准备
- 从配置管理员处申请软件配置：《需求规格说明书》和《设计说明书》；根据需求规格说明书和设计说明书，详细理解用户的真正需求，并且对软件所实现的功能已经准确理解，然后着手制订测试用例。

测试用例编写方法（1）

- 测试用例要包括欲测试的功能、应输入的数据和预期的输出结果。测试数据应该选用少量、高效的测试数据进行尽可能完备的测试；基本目标是：设计一组发现某个错误或某类错误的测试数据，测试用例应覆盖方面：
  - 1. 正确性测试：输入用户实际数据以验证系统是满足需求规格说明书的要求；测试用例中的测试点应首先保证要至少覆盖需求规格说明书中的各项功能，并且正常。
  - 2. 容错性（健壮性）测试：程序能够接收正确数据输入并且产生正确（预期）的输出，输入非法数据（非法类型、不符合要求的数据、溢出数据等），程序应能给出提示并进行相应处理。把自己想象成一名对产品操作一点也不懂的客户，在进行任意操作。

测试用例编写方法（2）

- 3. 完整（安全）性测试：对未经授权的人使用软件系统或数据的企图，系统能够控制的程度，程序的数据处理能够保持外部信息（数据库或文件）的完整。
- 4. 接口间测试：测试各个模块相互间的协调和通信情况，数据输入输出的一致性和正确性。
- 5. 数据库测试：依据数据库设计规范对软件系统的数据库结构、数据表及其之间的数据调用关系进行测试。
- 6. 边界值分析法：确定边界情况（刚好等于、稍小于和稍大于和刚刚大于等价类边界值），针对我们的系统在测试过程中主要输入一些合法数据/非法数据，主要在边界值附近选取。
- 7. 压力测试：输入10条记录运行各个功能，输入30条记录运行，输入50条记录运行。。。进行测试。

测试用例编写方法（4）

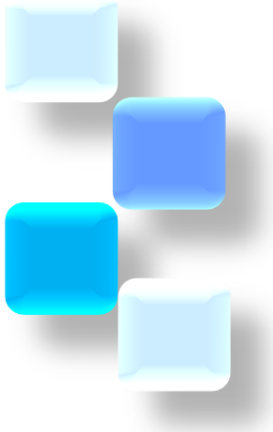
- 8. 等价划分：将所有可能的输入数据（有效的和无效的）划分成若干个等价类。
- 9. 错误推测：主要是根据测试经验和直觉，参照以往的软件系统出现错误之处。
- 10. 效率：完成预定的功能，系统的运行时间（主要是针对数据库而言）。
- 11. 可理解（操作）性：理解和使用该系统的难易程度（界面友好性）。
- 12. 可移植性：在不同操作系统及硬件配置情况下的运行性。
- 13. 回归测试：按照测试用例将所有的测试点测试完毕，测试中发现的问题开发人员
- 14. 比较测试：将已经发版的类似产品或原有的老产品与测试的产品同时运行比较，或与已往的测试结果比较。

测试用例编写方法（4）

【说明：针对不同的测试类型和测试阶段，测试用例编写的侧重点有所不同】

- 1. 其中第1、2、6、8、9、13项为模块（组件、控件）测试、组合（集成）测试、系统测试都涉及并重点测试的方面。
- 2. 单元（模块）测试（组件、控件）测试：重点测试第5项。
- 3. 组合（集成）测试：重点进行接口间数据输入及逻辑的测试，即第4项。
- 4. 系统测试：重点测试第3、7、10、11、12、14项。
- 5. 其中压力测试和可移植性测试如果是公司的系列产品，可以选用其中有代表性的产品进行一次代表性测试即可。
- 6. GMPS基础测试用例设计完成后，其他的测试项目只编写设计与之不同部分的测试用例。
- 7. 对于每个测试项目测试的测试用例不是一成不变的，随着测试经验的积累或在测试其他项目发现有测试不充分的测试点时，可以不断的补充完善测试项目的测试用例。

测试用例设计



测试用例概述
测试用例的组成元素与范例
白盒测试用例设计方法
黑盒测试用例设计方法
测试类型与测试用例设计
设计测试用例的策略选择
测试用例设计工具
测试用例编写
案例研究：测试用例设计

案例研究1：ATM取款机模拟器

如何操作？（1）

- 插卡

如何操作？（2）

- 选择小键盘输入用户密码，ATM显示器显示“******”，点击“确定”按钮

如何操作？（3）

- 如果密码正确我们可以进入个人账户主界面：

如何操作？（4）

- 查询余额

如何操作？（5）

- 取款

如何操作？（6）

- 当前快捷操作的有“100”、“200”、“500”、“自定义取款”，选择“自定义取款”

如何操作？（7）

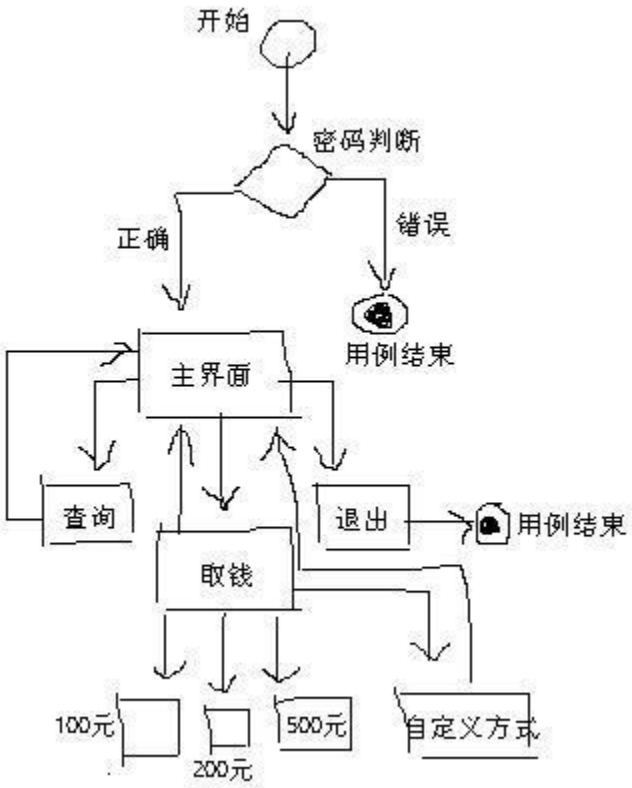
- 再次查询余额

如何操作？（8）

- 返回-退卡

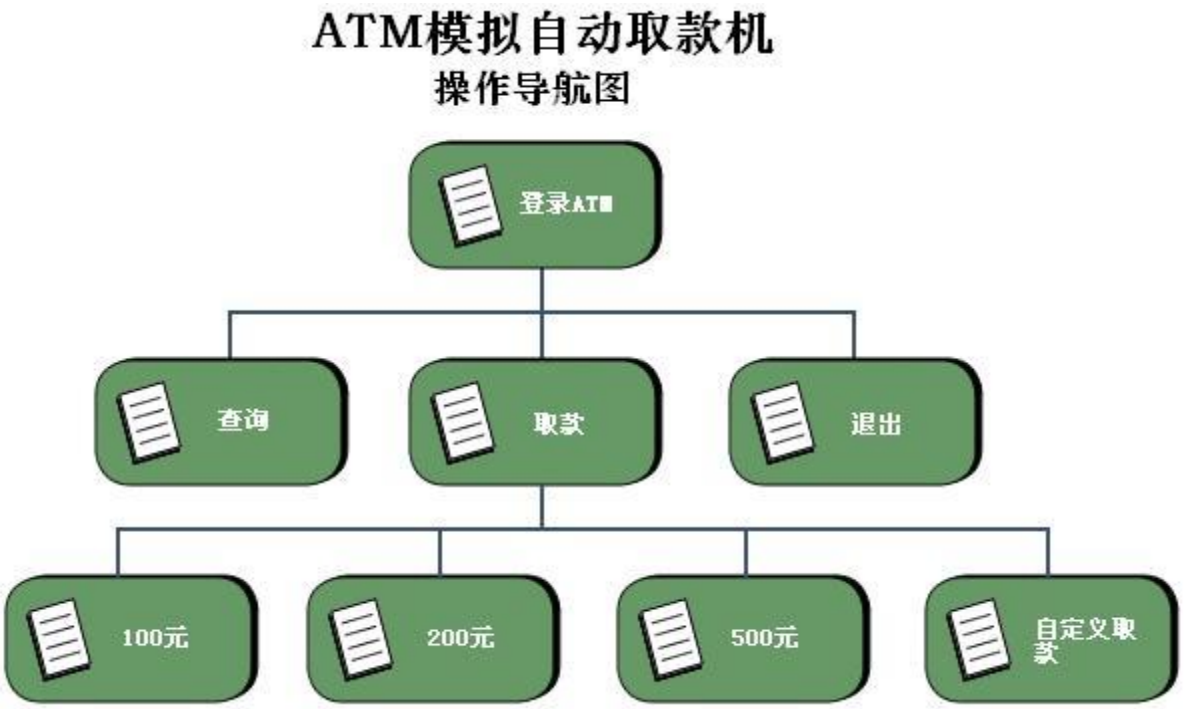
分析

- 业务流程



UI界面的导航图

- UI界面的导航图



测试用例设计-登录（1）

- **1. 简要说明**

本用例针对普通用户登录ATM取款机系统的功能操作测试。本用例不包含用户密码后台验证测试。

本用例的主角是普通用户，已知密码设定“123456”为正确。

2. 操作顺序

ATM取款机初始化完毕插卡后，本测试用例就开始使用了。

基本操作顺序 - 输入用户密码

1. 初始界面，等待用户密码输入。
2. 普通用户点击键盘“1”。
3. 普通用户点击键盘“2”。
4. 普通用户点击键盘“3”。
5. 普通用户点击键盘“4”。
6. 普通用户点击键盘“5”。
7. 普通用户点击键盘“6”。
8. 系统后台验证普通用户密码，正确。
9. 系统切入ATM取款机普通用户个人帐户界面。

测试用例设计-登录（2）

- **备选流**

1. 初始界面，等待用户密码输入。
2. 普通用户点击键盘“2”。
3. 普通用户点击键盘“3”。
4. 普通用户点击键盘“4”。
5. 普通用户点击键盘“5”。
6. 普通用户点击键盘“6”。
7. 普通用户点击键盘“7”。
8. 系统后台验证普通用户密码，错误，等待继续输入。

备选测试数据

序号	测试数据	期望值	实际值
01	123456	T	
02	234567	F	
03	00.564	E	

测试用例设计-登录（3）

- **特殊需求**
  特殊需求将在下次迭代中确定。

  前置测试条件
  1. 插卡
  在本用例开始前，普通用户要登录插卡。

  后置测试条件
  后置测试条件将在下次迭代中确定。

  扩展点
  用户密码输入错误三次，系统返回ATM取款机普通用户个人帐户界面。
-

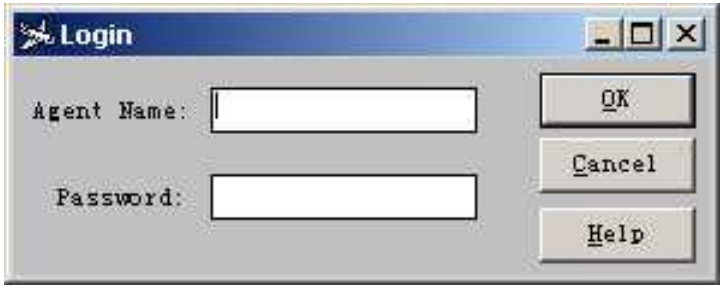
案例研究2：测试用户登录对话框的功能（1）

测试场景：

在各种输入条件下，测试程序的登录对话框功能。

用户名和密码的规则如下：

- 用户名长度为6至10位（含6位和10位）
- 用户名由字符（a-z、A-Z）和数字（0-9）组成
- 不能为空、空格和特殊字符
- 密码规则同用户名规则



案例研究2：测试用户登录对话框的功能（2）

- 确定输入数据的情形：

操作步骤	预期结果
输入正确的用户名和口令（均为6位），点击[OK]按钮	进入系统
输入正确的用户名和口令（均为10位），点击[OK]按钮	进入系统
输入正确的用户名和口令（均为6至8位之间），……	进入系统
用户名为空，……	提示输入用户名不能进入系统
用户名为空格，……	提示无效用户名不能进入系统
用户名小于6位，……	提示用户名太短不能进入系统
……	……

案例研究2：测试用户登录对话框的功能（3）

- 确定具体的输入数据：

“用户名”	“口令”	“预期结果”	说明
“user10”	“pass10”	进入系统	正确的用户名和口令(6位)
“user789”	“pass789”	进入系统	正确的用户名和口令(7-9位)
“user000010”	“pass000010”	进入系统	正确的用户名和口令(10位)
“”	“pass”	提示输入用户名 不能进入系统	用户名为空
“空格”	“pass”	提示无效用户名 不能进入系统	用户名为空格
“user”	“userpass”	提示用户名太短 不能进入系统	用户名小于6位
“user0000011”	“userpass”	提示用户名太长 不能进入系统	用户名大于10位
……	……	……	……

案例研究3：易用性（1）

- 控件

序号	测试内容	执行结果
1	按钮名称易懂，用词准确，与同一界面上的其他按钮易于区分	
2	常用按钮支持快捷方式	
3	相同或相近功能的按钮用frame框起来，并有功能说明或标题	
4	选项卡控件（tab）支持在页面间的快捷切换，常用快捷键ctrl+tab	
5	默认按钮要支持回车操作	
6	选择常用功能或数值作为默认值	
7	复选框或单选框有默认选项	
8	界面空间较小时采用下拉列表框而不采用单选框	
9	不同界面的通用按钮保持一致	
10	常用按钮的等价按键保持一致	
11	对可能给用户带来损失的操作最好支持可逆性处理	
12	对可能造成等待时间较长的操作应该提供取消功能，并显示操作的状态	
13	Tab键顺序要从上到下，从左到右	

案例研究3：易用性（2）

- 菜单

序号	测试内容	执行结果
1	常用菜单项要有快捷键	
2	菜单项前的图标能直观的代表要完成的操作	
3	一组菜单的使用有先后要求时，按照先后次序排列	
4	没有顺序要求的菜单按使用频率和重要性排列，常用的和重要的放前面	
5	主菜单的宽度要接近，字数不多于四个，每个菜单项的字数最好能相同	
6	相同或相近功能的工具栏放在一起	
7	工具栏的图标能直观的代表要完成的操作	
8	菜单和工具栏有清楚的界限	
9	菜单和状态条通常采用5号字体	

案例研究3：易用性（3）

- 快捷方式

序号	测试内容	执行结果
1	编辑：Ctrl+A全选；Ctrl+C 拷贝；Ctrl+V粘贴；Ctrl+X剪切；Ctrl+Z撤消操作；Ctrl+Y恢复操作；Ctrl+D删除；Ctrl+F寻找；Ctrl+H替换；Ctrl+I插入；Ctrl+TAB下一个TAB窗口	
2	文件操作：Ctrl+P打印；Ctrl+W关闭；Ctrl+N新建；Ctrl+S保存；Ctrl+O打开；	
3	主菜单：Alt+F文件；Alt+E编辑；Alt+T工具；Alt+W窗口；Alt+H帮助	
4	Windows保留键：Ctrl+Esc任务列表；Ctrl+F4关闭窗口；Alt+F4结束应用；Alt+Tab切换到下个应用；Enter 缺省按钮/确认操作；Esc 取消按钮/取消操作；Shift+F1 上下文相关帮助	
5	其他组合：Alt+N否；Alt+D删除；Alt+Q退出；Alt+A添加；Alt+E编辑；Alt+B浏览；Alt+R读；Alt+W写；	

案例研究3：易用性（4）

- 联机帮助

序号	测试内容	执行结果
1	操作时提供及时调用系统帮助的功能。常用F1来调用	
2	最好提供目前流行的联机帮助格式或HTML帮助格式	
3	如果没有提供书面的帮助文档的华，要有打印帮助的功能	
4	在帮助中提供软件的技术支持方式	

案例研究4：兼容性

- 与操作系统兼容

序号	操作系统
1	Windows 2000 s
2	Windows XP p
3	Windows 2003 s

- 与浏览器兼容

- 与整机兼容

案例研究6：文档测试

编号	检查项目	测试内容	测试结果
1	术语	用户能否理解术语，是否需要定义？术语是否标准？术语使用是否一致？	
2	标题	标题是否合适?有无丢失的标题？标题是否和实际产品一致	
3	内容	功能描述正确，清晰。 涉及的各个菜单、控件的名称与软件系统的名称一致。	
4	逐步执行	仔细阅读文字，完全按照提示操作，不要任何假设，将执行结果和文档描述进行比较。 确保所有信息真实正确和实际产品功能一致。 检查目录、索引和章节引用。 检查搜寻的正确性。 检查网站URL能否正确链接。	
5	图表和拷屏	检查图表的准确度和精确度。 确保拷屏和实际产品一致，不是来源于已修改过的版本。 图表的标题正确。	
6	示例	文档当中的示例，需要载入并使用。例如，示例为一段代码，就要输入或复制并执行它，保证示例可以执行。	
7	错别字	无错别字 标点符号正确	
8	排版	排版正确 风格一致	

案例研究7：安装测试

序号	测试内容	执行结果
1	执行典型安装：执行安装步骤，按功能测试方法确认功能正确，包括：各种控件、tab键、回车键、快捷键、错误提示消息等	
2	执行自定义安装：1）执行安装步骤，按功能测试方法确认功能正确，包括：各种控件、tab键、回车键、快捷键、错误提示消息等 2）选择与典型安装不同的安装路径和功能组件	
3	执行网络安装：执行安装步骤，按功能测试方法确认功能正确，包括：各种控件、tab键、回车键、快捷键、错误提示消息等	
4	取消或关闭安装过程，程序没有安装，检查注册表、安装路径中不存在程序的任何信息	
5	按界面和易用性测试规则，检查安装过程中的所有界面	
6	检查能否同时安装一个软件的多个版本	

案例研究8：运行测试

序号	测试内容	执行结果
1	确认安装的软件都可以正常的打开和关闭，功能可以使用	
2	确认软件的安装目录和安装内容正确，没有增加和遗漏	
3	安装软件之后的注册表信息正确	
4	对于多语言的软件要确认软件的字符编码	
5	确认产品信息与实际版本一致	
6	检查开始程序、桌面快捷方式、快速启动图标的名称正确，无错别字，可以正确打开相应程序	
7	对于正式版、升级版、试用版和限时版的软件，检查其时间锁或限制是否正确	

案例研究9：卸载测试

序号	测试内容	执行结果
1	安装完成后，先简单的使用一些功能，然后再执行卸载操作	
2	卸载完成后检查注册表中的相关注册信息都被删除	
3	卸载完成后检查系统是否把所有的文件全部删除，安装时创建的目录文件夹、开始菜单、桌面快捷方式、快速启动图标都被删除	
4	取消或关闭卸载过程，程序不被删除，仍然可以使用	
5	按界面和易用性测试规则，检查安装过程中的所有界面	
6	按照文档测试规则，检查安装过程中的所有文档（许可协议等）	
7	突然中断卸载过程	
8	卸载过程中介质处于忙碌状态	
9	卸载正在使用的程序	

本节课程总结

Soft Test 北京软测

Thank You !






软件测试教程