黑盒测试技术

中国软件评测中心
黄江平

黑盒测试技术

- 基本内容
- 测试用例的设计方法
- 测试用例的组织与编写
- 如何设计测试用例
- 一些测试方法和技巧
- 自动化测试工具

基本内容

- 软件测试种类
- 软件测试的共同特性
- 什么是软件缺陷
- 什么是黑盒测试
- 什么是通过测试
- 什么是失败测试
- 什么是测试用例

软件测试的种类

- 符合性测试
- 验收测试
- 易用性测试
- 兼容性测试
- 可靠性测试
- 安全性测试
- 性能测试
- ……

软件测试的共同特性

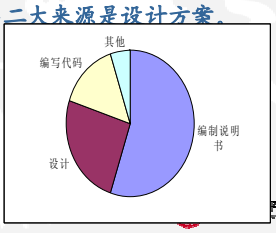
- 每种测试都需要测试员按照产品行为描述来实施。
- 每种测试都需要产品运行于真实或模拟环境下。
- 每种测试都要求以系统方法展示产品功能性，说明测试结果是肯定的还是否定的，以及是否可判断其中的区别。

什么是软件缺陷

- 软件未达到产品说明书标明的功能。
- 软件出现了产品说明书指明不会出现的错误。
- 软件功能超出产品说明书指明范围。
- 软件未达到产品说明书虽未指出但应达到的目标。
- 软件测试员认为软件难以理解、不易使用、运行速度缓慢，或者最终用户认为不好。

为什么会出现软件缺陷

- 导致软件缺陷最大的原因是产品说明书。
- 软件缺陷的第二大来源是设计方案。
- 编写代码
- 其他



软件缺陷的修复费用

- 从开始到计划、编制、测试、一直到公开使用的过程中，都有可能发现软件缺陷。
- 随着时间推移，修复软件缺陷的费用呈几何数级地增长。

什么是黑盒测试

- 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。
- 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规格说明书的规定正常使用。
- 黑盒测试又称功能测试。

黑盒测试

主要是为了发现以下几类错误:

①是否有不正确或遗漏了的功能?

②在接口上，输入能否正确地接受？能否输出正确的结果?

③是否有数据结构错误或外部信息（例如数据文件）访问错误?

④性能上是否能够满足要求?

⑤是否有初始化或终止性错误?

黑盒测试

黑盒测试又称功能测试、数据驱动测试或基于规格说明的测试，也可称为用户测试，主要应用于快速应用开发（RAD）环境

什么是通过测试

- 在设计和执行测试用例时，总是首先进行通过测试。在破坏性试验之前看看软件基本功能是否实现是很重要的，否则在正常使用软件时就会奇怪为什么有那么多软件缺陷。

什么是失败测试

- 确信软件在普通情况下正确运行之后，就可以采取各种手段通过搞垮软件来找出缺陷。纯粹为了破坏软件而设计和执行的测试用例称为失败测试或迫使出错测试

为什么做测试用例

完全测试是不可能的：

- 输入量太大；
- 输出结果太多；
- 软件实现途径太多；
- 软件说明书没有客观标准。从不同角度看，软件缺陷的标准不同。

什么是测试用例

- 为达到最佳的测试效果或高效的揭露隐藏的错误而精心设计的少量测试数据，称之为测试用例。
- 我们不可能进行穷举测试，为了节省时间和资源、提高测试效率，必须要从数量极大的可用测试数据中精心挑选出具有代表性或特殊性的测试数据来进行测试。
- 一个好的测试用例是在于它能发现至今未发现的错误。

使用测试用例的好处

- 在开始实施测试之前设计好测试用例，可以避免盲目测试并提高测试效率。
- 测试用例的使用令软件测试的实施重点突出、目的明确。
- 在软件版本更新后只需修正少部分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。
- 功能模块的通用化和复用化使软件易于开发，而相对于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精化其效率也不断攀升。

测试用例的设计过程

- 测试设计员（分析设计员）依据不同阶段的测试计划、设计模型和实施模型来设计该阶段测试用例。
- 测试设计员是具有丰富测试经验或具有软件分析设计能力的高级测试工程师。如果没有测试设计员，则可用分析设计员代替。
- 针对白盒，还应有驱动程序和桩模块。

测试点的确定

- 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。

理解上的误区

- 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。

中国软件评测中心 China Software Testing Center

19

测试用例的设计方法

- 等价类划分
- 边界值分析
- 因果图
- 比较法
- ……

中国软件评测中心 China Software Testing Center

20

等价类划分

- 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。每一类的代表性数据在测试中的作用等价于这一类中的其他值，也就是说，如果某一类中的一个例子发现了错误，这一等价类中的其他例子也能发现同样的错误；反之，如果某一类中的一个例子没有发现错误，则这一类中的其他例子也不会查出错误。

中国软件评测中心 China Software Testing Center

21

怎样划分等价类（一）

1) 如果输入条件规定了取值的范围或值的个数，则可确定一个有效等价类和两个无效等价类；
2) 如果一个输入条件说明了一个"必须成立"的情况，则可划分一个有效等价类和一个无效等价类；
3) 如果输入条件规定了输入数据的一组可能的值，而且程序是用不同的方式处理每一种值，则可为每一种值划分一个有效等价类，并划分一个无效等价类；

中国软件评测中心 China Software Testing Center

22

怎样划分等价类（二）

4) 如果我们确知，已划分的某等价类中的各元素（例子）在程序中的处理方式是不同的，则应据此将此等价类进一步划分成更小的等价类。
5) 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
…	…	…
…	…	…

中国软件评测中心 China Software Testing Center

23

确定等价类测试用例的步骤

- 为每个等价类规定一个惟一的编号；
- 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。

中国软件评测中心 China Software Testing Center

24

示例1：关于复制的等价划分

- 有5种执行方式：单击复制菜单命令，键入c或C，按Ctrl+c或Ctrl+Shi+c组合键；
- 可以把这5种输入途径划分减为3个，单击菜单命令，键入C和按Ctrl+c组合键；
- 对软件质量有了信心之后，知道无论以何种方式激活复制功能都工作正常，甚至可以进一步缩减为1个区间，例如按Ctrl+c组合键。

示例2：关于文件名的等价划分

- Windows文件名可以包含除了、/：·？"”<>和\之外的任意字符。文件名长度是1—255个字符；
- 等价区间有合法字符、非法字符、合法长度的名称、过长名称和过短名称 。

示例3：关于三角形的等价划分

- 问题：根据下面给出的规格说明，利用等价类划分的方法，给出足够的测试用例。“一个程序读入3个整数，把这三个数值看做一个三角形的3条边的长度值。这个程序要打印出信息，说明这个三角形是不等边的、是等腰的、还是等边的。”

示例3：分析三角形的特点

- 我们可以设三角形的3条边分别为A，B，C。如果它们能够构成三角形的3条边，必须满足：A>0，B>0，C>0，且A+B>C，B+C>A，A+C>B；
- 如果是等腰的，还要判断A=B，或B=C，或A=C ；
- 如果是等边的，则需判断是否A=B，且B=C，且A=C ；

示例3：列出三角形的等价类列表

输入条件	有效等价类	无效等价类
是否三角形的3条边	(A>0)，(1) (B>0)，(2) (C>0)，(3) (A+B>C)，(4) (B+C>A)，(5) (A+C>B) (6)	(A≤0)，(7) (B≤0)，(8) (C≤0)，(9) (A+B≤C)，(10) (B+C≤A)，(11) (A+C≤B) (12)
是否等腰三角形	(A=B)，(13) (B=C)，(14) (C=A) (15)	(A≠B) and (B≠C) and (C≠A) (16)
是否等边三角形	(A=B) and (B=C) and (C=A) (17)	(A≠B)，(18) (B≠C)，(19) (C≠A) (20)

示例3：设计三角形的测试用例

序号	(A，B，C)	覆盖等价类	输出
1	(3，4，5)	(1)，(2)，(3)，(4)，(5)，(6)	一般三角形
2	(0，1，2)	(7)	
3	(1，0，2)	(8)	
4	(1，2，0)	(9)	不能构成三角形
5	(1，2，3)	(10)	
6	(1，3，2)	(11)	
7	(3，1，2)	(12)	
8	(3，3，4)	(1)，(2)，(3)，(4)，(5)，(6)，(13)	
9	(3，4，4)	(1)，(2)，(3)，(4)，(5)，(6)，(14)	等腰三角形
10	(3，4，3)	(1)，(2)，(3)，(4)，(5)，(6)，(15)	
11	(3，4，5)	(1)，(2)，(3)，(4)，(5)，(6)，(16)	非等腰三角形
12	(3，3，3)	(1)，(2)，(3)，(4)，(5)，(6)，(17)	是等边三角形
13	(3，4，4)	(1)，(2)，(3)，(4)，(5)，(6)，(14)，(18)	
14	(3，4，3)	(1)，(2)，(3)，(4)，(5)，(6)，(15)，(19)	非等边三角形
15	(3，3，4)	(1)，(2)，(3)，(4)，(5)，(6)，(16)，(20)	

边界值分析

- 边界值分析法是一种补充等价划分的测试用例设计技术，它不是选择等价类的任意元素，而是选择等价类边界的测试用例。实践证明，在设计测试用例时，对边界附近的处理必须给予足够的重视，为检验边界附近的处理专门设计测试用例，常常取得良好的测试效果。边界值分析法不仅重视输入条件边界，而且也从输出域导出测试用例。

边界值设计遵守的几条原则（一）

1) 如果输入条件规定了取值范围，应以该范围的边界内及刚刚超范围的边界外的值作为测试用例。如以a和b为边界，测试用例应当包含a和b及略大于a和略小于b的值；
2) 若规定了值的个数，分别以最大、最小个数及稍小于最小、稍大于最大个数作为测试用例；

边界值设计遵守的几条原则（二）

3) 针对每个输出条件使用前面的第1）和2）条原则；
4) 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；
5) 分析规格说明，找出其他的可能边界条件。

一个演示边界条件缺陷的简单程序

```
1：Rem Create a 10 element integer array
2：Rem Initialize each element to -1
3：Dim data(10) As Integer
4：Dim I As Integer
5：For I=1 TO 10
6：data(i)=-1
7：Next i
8：End
```


边界问题会在哪儿呢？

```
data(1)=-1    data(2)=-1
data(3)=-1    data(4)=-1
data(5)=-1    data(6)=-1
data(7)=-1    data(8)=-1
data(9)=-1    data(10)=-1

          data(0)=0
```


边界条件类型

- 数值、速度、字符、地址、位置、尺寸、数量等等；
- 第一个／最后一个、最小值／最大值、开始／完成，超过／在内、空／满、最短／最长、最慢／最快、最早／最迟、最大／最小、最高／最低、相邻／最远等等；
- 越界测试通常是简单地加1或者很小的数（对于最大值）和减少1或者很小的数（对于最小值）

可能的边界条件（一）

- 如果文本输入域允许输入1~255个字符，就尝试输入1个字符和255 个字符作为合法区间。还可以输入254个字符作为合法测试。输入0个字符和256个字符作为非法区间 ；
- 如果程序读写软盘，就尝试保存一个尺寸极小，甚至只有一项的文件，然后保存一个很大的——刚好在软盘容量限制之内的文件。还要尝试保存空文件和尺寸大于软盘容量的文件。

可能的边界条件（二）

- 如果程序允许在一张纸上打印多个页面，就尝试只打印一页(标准情况)，并尝试打印所允许的最多页面，如果可能，还要尝试打印0页和多于所允许的页面;
- 如果测试飞行模拟程序，尝试控制飞机在地平线上和最大允许高度飞行。尝试在地平线和海平面之下飞行，以及太空飞行;
- ……

次边界条件

- 普通边界条件是最容易找到的，它们在产品说明书中有定义，或者在使用软件的过程中确定。
- 有些边界在软件内部，最终用户几乎看不到，但是软件测试仍有必要检查。

2的乘方

术语	范围或值
位	0 或 1
双位	0～15
字节	0～255
字	0～65,535 或 0～4,294,967,295
千	1,024
兆	1,048,576
亿	1,073,741,824
万亿	1,099,511,627,776

ASCII字符表

字符	ASCII 值	字符	ASCII 值
Null	0	B	66
Space	32	Y	89
/	47	Z	90
0	48	[	91
1	49	'	96
2	50	a	97
9	57	b	98
:	58	y	121
@	64	z	122
A	65	{	123

非法、错误、不正确和垃圾数据

- 从纯粹的软件测试观点来看，如果利用前述技术全面测试证明软件能够工作了，就不必再做破坏实验。然而，考虑到软件要应付用户千奇百怪的使用方式，这样做肯定没错。
- 非法、错误、不正确和垃圾数据测试是很有意思的。如果软件要求输入数字，就输入字母。如果软件只接受正数，就输入负数。如果软件对数据敏感，就看它在公元3000年是否还能正常工作。

什么是因果图法？

- 等价类划分方法和边界值分析法都是着重考虑输入条件，并没有考虑到输入情况的各种组合，也没考虑到各个输入情况之间的相互制约关系。
- 因果图方法的思路是：从用自然语言书写的程序规格说明的描述中找出因（输入条件）和果（输出或程序状态的改变），通过因果图转换为判定表。

因果图用法的几个步骤 （一）

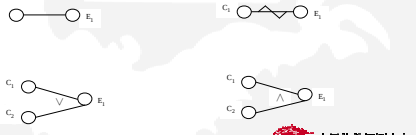
1) 分析程序规格说明的描述中，哪些是原因，哪些是结果。原因常常是输入条件或是输入条件的等价类，而结果是输出条件；
2) 分析程序规格说明的描述中语义的内容，并将其表示成连接各个原因与各个结果的"因果图"；
3) 由于语法或环境的限制，有些原因和结果的组合情况是不可能出现的。为表明这些特定的情况，在因果图上使用若干个特殊的符号标明约束条件；

因果图用法的几个步骤（二）

4) 把因果图转换成判定表；
5) 为判定表中每一列表示的情况设计测试用例。

因果图的基本图形符号

- 通常在因果图中，用C_i表示原因，E_i表示结果，各结点表示状态，可取值"0"或"1"。"0"表示某状态不出现，"1"表示某状态出现。



因果图的基本图形符号说明

- 恒等：若原因出现，则结果出现；若原因不出现，则结果也不出现。
- 非(～)：若原因出现，则结果不出现；若原因不出现，结果反而出现。
- 或(∨)：若几个原因中有1个出现，则结果出现；若几个原因都不出现，则结果不出现。
- 与(∧)：若几个原因都出现，结果才出现。若其中有1个原因不出现，则结果不出现。

因果图的约束符号

E（互斥） I（包含） O（唯一） R（要求） M（屏蔽）

因果图的约束符号说明

- E（互斥）：表示a、b两个原因不会同时成立，两个中最多有一个可能成立。
- I（包含）：表示a、b、c这3个原因中至少有一个必须成立。
- O（惟一）：表示a和b当中必须有一个，且仅有一个成立。
- R（要求）：表示当a出现时，b必须也出现。a出现时不可能b不出现。
- M（屏蔽）：表示当a是1时，b必须是0。而当a为0时，b的值不定。

因果图示例：自动售货机

- 产品说明书：有一个处理单价为1元5角钱的盒装饮料的自动售货机软件。若投入1元5角硬币，按下"可乐"、"雪碧"、或"红茶"按钮，相应的饮料就送出来。若投入的是2元硬币，在送出饮料的同时退还5角硬币。

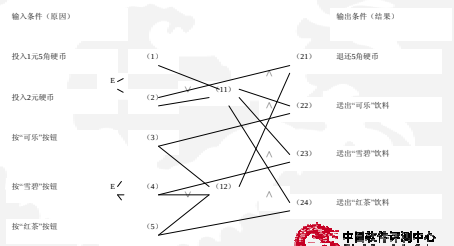
自动售货机：原因和结果

原因：
1. 投入1元5角硬币
2. 投入2元硬币
3. 按"可乐"按钮
4. 按"雪碧"按钮
5. 按"红茶"按钮

结果：
1. 退还5角硬币
2. 送出"可乐"饮料
3. 送出"雪碧"饮料
4. 送出"红茶"饮料

中间状态： 1、已投币　　2、已按钮

自动售货机：因果图



自动售货机：测试用例

			1	2	3	4	5	6	7	8	9	10	11
输入	投入1元5角硬币	(1)	1	1	1	1	0	0	0	0	0	0	0
	投入2元硬币	(2)	0	0	0	0	1	1	1	1	0	0	0
	按"可乐"按钮	(3)	1	0	0	0	1	0	0	0	1	0	0
	按"雪碧"按钮	(4)	0	1	0	0	0	1	0	0	0	1	0
	按"红茶"按钮	(5)	0	0	1	0	0	0	1	0	0	0	1
中间结点	已投币	(11)	1	1	1	1	1	1	1	1	0	0	0
	已按钮	(12)	1	1	1	0	1	1	1	0	1	1	1
输出	退还5角硬币	(21)	0	0	0	0	1	1	1	0	0	0	0
	送出"可乐"饮料	(22)	1	0	0	0	1	0	0	0	0	0	0
	送出"雪碧"饮料	(23)	0	1	0	0	0	1	0	0	0	0	0
	送出"红茶"饮料	(24)	0	0	1	0	0	0	1	0	0	0	0

比较测试

- 有时为保证系统的"绝对"可靠性，经常使用冗余的软件和硬件，以减少错误发生的可能性。这时根据同一的规格（需求）说明书由不同的开发小组开发出不同的软件版本，因此，可用相同的测试数据对它们进行测试以产生相同的输出，然后，执行所有版本并进行实时结果比较以保证一致性，这种测试就是比较测试（背靠背测试）。

测试用例设计方法选择的综合策略

- 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，则一开始就可选用因果图法。

中国软件评测中心 China Software Testing Center
55

GUI测试－窗口

- 窗口能否基于相关的输入或菜单命令适当的打开;
- 窗口能否改变大小、移动和滚动;
- 窗口中的数据能否有鼠标、功能键、方向箭头和键盘操作;
- 当被覆盖并重新调用后，窗口能否正确再生;
- 需要时能否使用所有窗口相关的功能，所有相关的功能是否可操作;
- 是否有相关的下拉式菜单、工具条、滚动条、对话框、按钮、图标和其他控制，既正确显示又完全可用;

中国软件评测中心 China Software Testing Center
56

GUI测试－下拉式菜单和鼠标操作

- 菜单条是否显示在合适的语境中;
- 应用程序的菜单条是否显示系统相关的特性;
- 下拉式操作能否正确进行，功能是否正确;
- 菜单、调色板和工具条能否正确地工作;
- 能否正确地列出所有的菜单功能和下拉式子功能;
- 能否通过鼠标操作所有的菜单功能，能否通过其他的文本命令激活每个菜单功能;
- 菜单功能能否随当前的窗口操作加亮或变灰;
- 如果要求多次点击鼠标，或鼠标有多个按钮，能否正确识别;

中国软件评测中心 China Software Testing Center
57

GUI测试－数据项

- 字母数字数据项能否正确回显，并输入到系统中;
- 图形方式的数据项（如滚动条）是否能正常工作;
- 数据输入消息是否可理解，能否识别非法数据;

中国软件评测中心 China Software Testing Center
58

客户/服务器体系结构的测试

- 从三个层次考虑
  - 各个客户端应按独立的模式进行测试，暂不考虑服务器和底层网络的运行;
  - 客户端软件和关联的服务器端应用一起测试，暂不多考虑网络运行;
  - 完整C/S体系结构被测试，包括网络运行和性能测试;

中国软件评测中心 China Software Testing Center
59

竞争条件和时序错乱

面临竞争条件的几个典型情形:

- 两个不同的程序同时保存或打开同一个文档。
- 共享同一台打印机、通信端口或者其他外围设备。
- 当软件处于读取或者修改状态时按键或者单击鼠标。
- 同时关闭或者启动软件的多个实例。
- 同时使用不同的程序访问一个共同数据库。

中国软件评测中心 China Software Testing Center
60

重复、压迫和重负

- 重复测试：不断执行同样的操作，这种反复测试的主要原因是看内存是否不足。
- 压迫测试：使软件在不够理想的条件下运行，观察软件对外部资源的要求和依赖的程度。
- 重负测试：提供条件任软件发挥，最大限度地发掘软件的能力。

实时系统的综合性测试步骤

- 任务测试：独立地测试各个任务，发现逻辑和功能错误；
- 行为测试：创建软件模型，用以仿真实时系统，并按照外部事件的序列检查其行为；
- 任务间测试：进行测试和时间有关的错误；
- 系统测试：发现软件和硬件接口间是否有问题。

编写测试用例

- 测试用例计划的目标
- 测试设计说明书是什么？
- 测试用例说明书是什么？
- 测试程序说明书是什么？

计划测试用例的重要性（一）

- 组织性：即使在小型软件项目上，也可能有数千个测试用例。建立案例可能需要一些测试员经过几个月甚至几年的时间。正确的计划会组织好案例，以便全体测试员和其他项目小组成员有效地审查和使用。
- 重复性：我们已经知道，在项目期间有必要多次执行同样的测试，以寻找新的软件缺陷，保证老的软件缺陷得以修复。假如没有正确的计划，就不可能知道最后执行哪个测试用例及其执行情况，以便重复原有的测试。

计划测试用例的重要性（二）

- 跟踪。计划执行多少个测试用例？在软件最终版本上执行多少个测试用例？多少个通过，多少个失败？有忽略的测试用例吗？等等。如果测试用例没有计划，就不能回答这些问题。
- 测试证实。在少数高风险行业中，软件测试小组必须证明确实执行了计划执行的测试。发布忽略某些测试用例的软件实际上是不合法和危险的。正确的测试用例计划和跟踪提供了一种证实测试的手段。

什么是测试设计说明

- 为单个软件特性定义测试方法的下一级细节是测试设计说明。
- ANSI／IEEE 829称测试设计说明为“提炼测试方法[在测试计划中定义]，明确指出设计包含的特性及其相关测试。如果要求完成测试还明确指出测试用例和测试程序，指定特性通过/失败的规则。”

测试设计说明的部分内容

- 标识符。用于引用和定位测试设计说明的唯一标识符。
- 要测试的特性。测试设计说明所包含的软件特性描述。
- 方法。用于测试特性的通用方法描述。
- 测试用例论证。对用于检查特性的具体测试用例的高级描述和引用。
- 通过/失败规则。描述是什么构成测试特性的通过和失败。

什么是测试用例说明

- ANSI / IEEE829标准称测试用例说明为——编写用于输入的实际数值和预期结果。测试用例还明确指出使用具体测试用例产生的测试程序的任何限制。

一个关于测试用例说明的表格

编号：

编制人		审定人	时间
软件名称		编号/版本	
测试过程			
用例编号			
参考信息（参考的文档及章节号或功能项）：			
输入说明（列出适用的输入项，覆盖正常、异常情况）：			
输出说明（逐条与输入项对应，列出预期输出）：			
环境要求（测试要求的软、硬件、网络要求）：			
特殊规程要求：			
用例间的依赖关系：			

测试用例说明的细节 （一）

- 标识符：由测试设计过程说明和测试程序说明引用的唯一标识符。
- 测试项：描述被测试的详细特性、代码模块等 。
- 输入说明：该说明列举送到软件执行测试用例的所有输入内容或者条件。
- 输出说明：描述进行测试用例预期的结果。

测试用例说明的细节（二）

- 环境要求：是指执行测试用例必要的硬件、软件、测试工具、实用工具、人员等等。
- 特殊要求：描述执行测试必须做到的特殊要求。
- 案例之间的依赖性：如果一个测试用例依赖于其他案例，或者受其他案例的影响，就应该在此注明。

打印机兼容性无限表的例子

测试案例序列号	型号	模式	黑白	选项
WP0001	Canon	BJC-7000	黑白	文字
WP0002	Canon	BJC-7000	黑白	超级照片
WP0003	Canon	BJC-7000	黑白	自动
WP0004	Canon	BJC-7000	黑白	草稿
WP0005	Canon	BJC-7000	彩色	文字
WP0006	Canon	BJC-7000	彩色	超级照片
WP0007	Canon	BJC-7000	彩色	自动
WP0008	Canon	BJC-7000	彩色	草稿
WP0009	HP	LaserJet IV	高	
WP0010	HP	LaserJet IV	中	
WP0011	HP	LaserJet IV	低	

什么是测试程序说明

- ANSI／IEEE829标准称测试程序说明为——明确指出为实现相关测试设计而操作软件系统和试验具体测试用例的全部步骤。

测试程序说明需定义的内容（一）

- 标识符：把测试程序与相关测试用例和测试设计捆绑在一起的唯一标识符。
- 目的：程序的目的以及将要执行的测试用例的引用信息。
- 特殊要求：执行程序所需的其他程序、特殊测试技术或者特殊设备。
- 程序步骤：执行测试的详细描述。
- 日志：指出用什么方式、方法记录结果和现象。

测试程序说明需定义的内容（二）

- 设置：说明如何准备测试。
- 启动：说明用于启动测试的步骤。
- 程序：描述用于运行测试的步骤。
- 衡量标准：描述如何判断结果。
- 关闭：说明由于意外原因推迟测试的步骤。
- 终止：描述测试正常停止的步骤。
- 重置：说明如何把环境恢复到测试前的状态。
- 偶然事件：说明如何处理计划之外的情况。

一个测试程序说明的例子

标识号：计算器程序
目的：这个程序描述执行加法测试用例必须的步骤
特殊要求：本次测试不需要特殊的硬件和软件
程序步骤：
日志：测试员按测试要求记录程序执行过程，所有必须填写的项都必须填写，包括问题的记录。
设置：测试者必须安装Windows98的干净副本，使用测试工具Tool-A和Tool-B等等。
启动：启动Windows98，点击开始按钮，选择程序，选择附件、选择计算器。
程序：用键盘输入每个测试用例并比较结果。
衡量标准：....

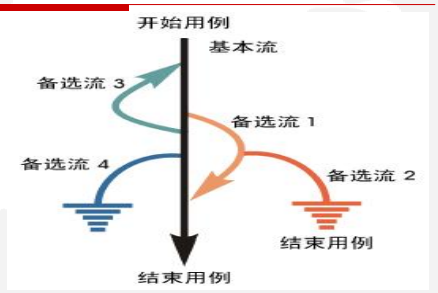
如何设计测试用例

- 用例场景
- 一个ATM的用例设计

测试用例场景

- 用例场景是通过描述流经用例的路径来确定的过程，这个流经过程要从用例开始到结束遍历其中所有基本流和备选流。
- 现在的软件几乎都是由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形成事件流。这种在软件设计方面的思想也可被引入到软件测试中，生动的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，同时测试用例也更容易的得到理解和执行。

用例场景描述



用例场景描述

- 场景1 基本流
- 场景2 基本流 备选流1
- 场景3 基本流 备选流1 备选流2
- 场景4 基本流 备选流3
- 场景5 基本流 备选流3 备选流1
- 场景6 基本流 备选流3 备选流1 备选流2
- 场景7 基本流 备选流4
- 场景8 基本流 备选流3 备选流4

中国软件评测中心 China Software Testing Center
80

测试用例例子

- 假定上图描述的用例对备选流3 规定如下："如果在上述步骤2'输入提款金额'中输入的美元量超出当前帐户余额，则出现此事件流。系统将显示一则警告消息，之后重新加入基本流，再次执行上述步骤2'输入提款金额'，此时银行客户可以输入新的提款金额。"

中国软件评测中心 China Software Testing Center
81

测试用例例子

- 据此，可以开始确定需要用来执行备选流3 的测试用例：

中国软件评测中心 China Software Testing Center
82

ATM（业务模型）

83

ATM（基本流）

- 步骤1：准备提款- 客户将银行卡插入ATM机的读卡机。
- 步骤2：验证银行卡- ATM 机从银行卡的磁条中读取帐户代码，并检查它是否属于可以接收的银行卡。
- 步骤3：输入PIN - ATM 要求客户输入PIN 码（4 位）。

中国软件评测中心 China Software Testing Center
84

ATM（基本流）

- 步骤4：验证帐户代码和PIN - 验证帐户代码和PIN 以确定该帐户是否有效以及所输入的PIN 对该帐户来说是否正确。对于此事件流，帐户是有效的而且PIN 对此帐户来说正确无误。
- 步骤5：ATM 选项- ATM 显示在本机上可用的各种选项。在此事件流中，银行客户通常选择"提款"。
- 步骤6：输入金额- 要从ATM 中提取的金额。对于此事件流，客户需选择预设的金额（10 美元、20 美元、50 美元或100 美元）。

中国软件评测中心 China Software Testing Center

85

ATM（基本流）

- 步骤7：授权-ATM 通过将卡ID、PIN、金额以及帐户信息作为一笔交易发送给银行系统来启动验证过程。对于此事件流，银行系统处于联机状态，而且对授权请求给予答复，批准完成提款过程，并且据此更新帐户余额。
- 步骤8：出钞- 提供现金。
- 步骤9：返回银行卡- 银行卡被返还。
- 步骤10：收据- 打印收据并提供给客户。ATM 还相应地更新内部记录。

中国软件评测中心 China Software Testing Center

86

ATM（备选流）

ATM（备选流）

ATM（说明）

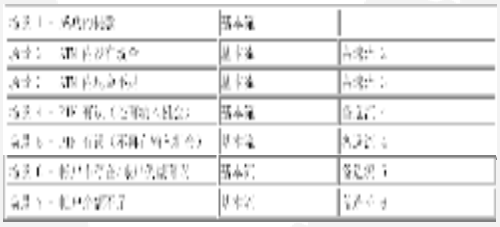
第一次迭代中，根据迭代计划，我们需要核实提款用例已经正确地实施。此时尚未实施整个用例，只实施了下面的事件流：

- 基本流- 提取预设金额（10 美元、20 美元、50 美元、100 美元）
- 备选流2 - ATM 内没有现金
- 备选流3 - ATM 内现金不足
- 备选流4 - PIN 有误
- 备选流5 - 帐户不存在/帐户类型有误
- 备选流6 - 帐面金额不足

中国软件评测中心 China Software Testing Center

89

ATM（场景）

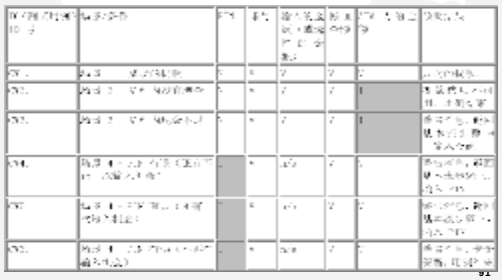


注：为方便起见，备选流3 和6（场景3 和7）内的循环以及循环组合未纳入上表。

中国软件评测中心 China Software Testing Center

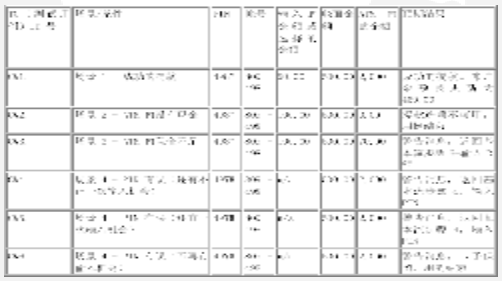
90

ATM（测试用例矩阵）



91

ATM（测试用例数据）



ATM（其他场景用例）

- 场景6 - 帐户不存在/帐户类型有误：未找到帐户或帐户不可用
- 场景6 - 帐户不存在/帐户类型有误：禁止从该帐户中提款
- 场景7 - 帐户余额不足：请求的金额超出帐面金额

93

ATM（其他测试用例）

- 无效卡（所持卡为挂失卡、被盗卡、非承兑银行发卡、磁条损坏等）
- 无法读卡（读卡机堵塞、脱机或出现故障）
- 帐户已消户、冻结或由于其他方面原因而无法使用
- ATM 内的现金不足或不能提供所请求的金额（与 CW3 不同，在CW3 中只是一种币值不足，而不是所有币值都不足）
- 无法联系银行系统以获得认可
- 银行网络离线或交易过程中断电

94

一些测试方法和技巧

- 关于易用性的分析
- 自由测试（理解软件四种能力）

95

优秀的用户界面具有的要素

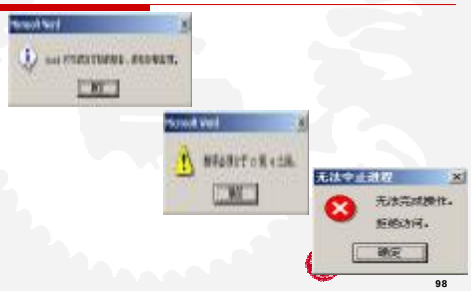
- 符合标准和规范
- 直观性
- 一致性
- 灵活性
- 舒适性
- 正确性
- 实用性

96

1、符合标准和规范

- 如果软件在Mac或者Windows等现有平台上运行，标准是已经确立的。
- 平台也可能没有标准，也许测试的软件就是平台本身。在这种情况下，设计小组可能成为软件易用性标准的创立者。

Windows界面标准中三种级别信息



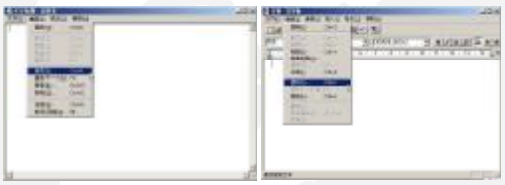
2、直观性

- 用户界面是否洁净、不唐突、不拥挤？UI不应该为用户制造障碍。所需功能或者期待的响应应该明显，并在预期出现的地方。
- UI的组织和布局合理吗？是否允许用户轻松地从一个功能转到另一个功能？下一步做什么明显吗？任何时刻都可以决定放弃或者退回、退出吗？输入得到承认了吗？菜单或者窗口是否深藏不露？
- 有多余功能吗？软件整体抑或局部是否做得太多？是否有太多特性把工作复杂化了？是否感到信息太庞杂？
- 如果其他所有努力失败，帮助系统真能帮忙吗？

3、一致性

- 快速键和菜单选项。在Windows中，按F1键总是得到帮助信息。
- 术语和命令。整个软件使用同样的术语吗？特性命名一致吗？例如，Find是否一直叫Find，而不是有时叫Search？
- 听众。软件是否一直面向同一听众级别？带有花哨用户界面的趣味贺卡程序不应该显示泄露技术机密的错误提示信息。
- 按钮位置和等价的按键。大家是否注意到对话框有OK按钮和Cancel按钮时，OK按钮总是在上方或者左方，而Cancel按钮总是在下方或者右方？同样的原因，Cancel按钮的等价按键通常是Esc，而选中按钮的等价按钮通常是Enter。保持一致。

Windows记事本和写字板



4、灵活性

- 状态终止和跳过。当软件具有用户非常熟悉的超级用户模式时，显然能够跳过众多提示或者窗口直接到达想去的地方。能够直接拨到公司电话分机的语音信箱就是一个例子。
- 数据输入和输出。用户希望有多种方法输入数据和查看结果。为了在写字板文档中插入文字，可以用键盘输入、粘贴、从6种文件格式读入、作为对象插入，或者用鼠标从其他程序拖动。

Windows计算器的两种选择

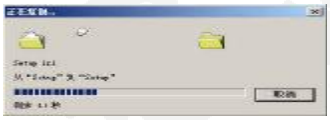
中国软件评测中心
103

5、舒适性

- 恰当。软件外观和感觉应该与所做的工作和使用者相符。金融商业应用程序不应该用绚丽的色彩和音效来表现狂放的风格。
- 错误处理。程序应该在用户执行严重错误的操作之前提出警告，并且允许用户恢复由于错误操作导致丢失的数据。
- 性能。快不见得是好事。不少程序的错误提示信息一闪而过，无法看清。如果操作缓慢，至少应该向用户反馈操作持续时间，并且显示它正在工作，没有停滞。

中国软件评测中心
104

Windows的进度条



中国软件评测中心
105

6、正确性

- 市场定位偏差。有没有多余的或者遗漏的功能，或者某些功能执行了与市场宣传材料不符的操作？
- 语言和拼写。程序员知道怎样只用计算机语言的关键字拼出句子，常常能够制造一些异想不到的用户信息。
- 不良媒体。媒体是软件UI包含的所有支持图标、图像、声音和视频。图标应该同样大，并且具有相同的调色板。声音应该都有相同的格式和采样率。正确的媒体从UI选择时应该显示出来。
- 所见即所得。保证UI所说的就是实际得到的。当单击Save按钮时，屏幕上的文档与存入磁盘的完全一样吗？从磁盘读出时，与原文档相同吗？

中国软件评测中心
106

7、实用性

- 不是指软件本身是否实用，而仅指具体特性是否实用。
- 在审查产品说明书、准备测试或者实际测试时，想一想看到的特性对软件是否具有实际价值。它们有助于用户执行软件设计的功能吗？如果认为它们没必要，就要研究一下找出它们存在于软件中的原因。

中国软件评测中心
107

辅助选项测试

- 视力损伤
- 听力损伤
- 运动损伤
- 认知和语言障碍

中国软件评测中心
108

软件的四种基本能力

- 输入：软件从其环境中接收输入；
- 输出：软件生成输出，并将输出提交给它的环境；
- 数据：软件内部以一种或多种数据结构形式存储数据；
- 计算：软件使用输入和存储的数据执行计算。

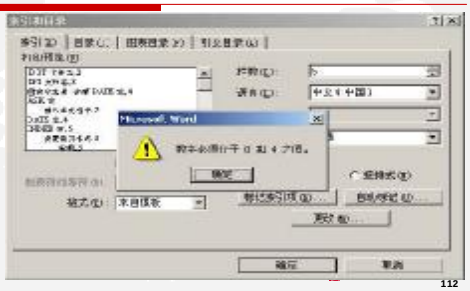
输入测试的建议（一）

1) 通过应用无效输入，确认所有错误信息至少能发现一次。考虑开发人员可能会错过的无效输入。
2) 强制给软件中可以通过用户接口设定的内部变量赋默认值。首先显示和接收已有值，然后赋给伪值，强制软件进行有效的计算。
3) 对每个输入域，键入类型错误的值以及表示字符串的值，该字符串可以以特殊方式进行处理，研究操作系统和程序设计语言，列出可能有问题的字符串。将其应用到每个输入域。

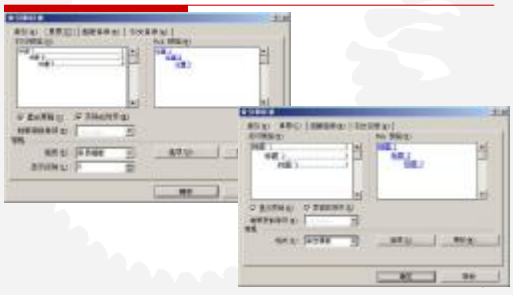
输入测试的建议（二）

4) 在每个输入域中键入允许的字符最大值。
5) 在按"OK"按钮之前，找到能够键入多个输入的输入面板。对每个输入域确定其合法值，并试图对输入的非法集合进行组合。
6) 找到接收输入的位置，并一次又一次地应用同一输入或输入系列。选择会产生一些基本计算或数据操作的输入，简单地显示在屏幕上。

示例1：输入强制产生所有错误信息



示例2：强制建立有默认值的输入



示例3：特殊的字符集合

- 字符集包括普通数字和特殊数字 。例如，ASC字符集有普通的字母字符和特殊字符，如控制字符和符号。
- 现应用程序的程序设计语言常有特定的处理一些字符和字符串的方法。例如，C语言使用像\n、++和&这样的字符用于特定的目的。
- 在操作系统中运行的应用程序也使用在设备名称、系统对象和程序的保留字符串集合。 如Windows的一些设备名：CON、NUL、AUX、CLOCK$、CONFIG$

例如：在IE中输入"file://c:\AUX "，就会使程序挂起。

示例4：输入缓冲区溢出

示例5：相互作用的输入组合

4 在Word里插入表格，我们可以通过试验或者从相关资料上看到列的最大值是63，行的最大值是32767。如果每个栏都键入小数值，Word会处理得很好。在一个栏中键入大数值，在另一个栏中键入小数值也没问题。但是，如果在列那一栏中键入50以上，在行那一栏中键入32000以上的值，应用程序就会挂起，因为这样就超过了机器的CPU周期。

中国软件评测中心
116

输出测试的建议

1) 挑选某输入，应用到受测软件中去，并记录输出。思考该输入在其他情况下应用会生成的输出，在这些其他情况中施加该输入，确保测试期间能观察到每个输出。
2) 思考软件不能或不应产生的输出，找出会产生这些非法输出的输入组合或序列。
3) 应用能产生某输出的输入，该输出具有一些可见性和可修改性（如大小）。强制修改其属性。
4) 确定受测软件何时刷新屏幕。搞清软件过于频繁地刷新屏幕或应刷新却没有刷新的情形。

中国软件评测中心
117

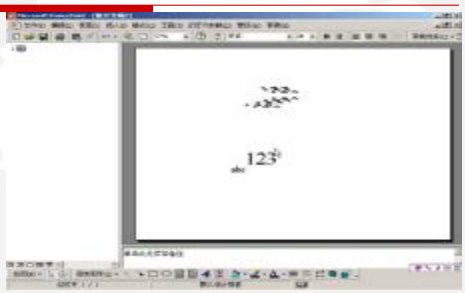
示例1：产生无效输出

4 例如，不熟悉闰年规则的测试员测试日历程序就会无效。对这样的测试员，日期2001年2月29日看起来是合理的。这样，理解问题域是测试的关键，对日历程序，并了解一些函数结果的受限范围，那么试图发现输入组合能强制这些函数产生无效结果，是很值得做的工作。

中国软件评测中心
118

示例2：改变输出属性

示例3：强制屏幕刷新



数据测试的建议

- 挑选一个输入，思考输入能应用的所有情况。通过在尽可能多的条件下应用输入来测试软件。思考在哪些条件下不应该施加输入。
- "看穿"界面，并标识数据结构，或评审源代码，如果源代码可获得的话。对任何给定的数据结构，在其中存储过多或过少的值。应用过多值意味着必须确定预置限值。存储过少值是指添加数据，并全部删除，然后再删除。
- 确定数据结构创建和使用的约束。找出修改数据的其他方法（使用软件的其他部分甚至是其他应用程序），使得能够违反这些约束。

121

示例1：强制数据结构溢出

- 在Word中插入表格，我们知道行数的上限是32767—两字节有符号整数的上限，这样可以通过增加行使该结构上溢。Word将会锁住。

122

计算测试的建议

- 找出计算产生的位置，并强制操作符与有冲突的操作数或对给定的操作符不能共存的多操作数匹配。
- 强制内部函数递归调用自身，包括包含自身的文档、指向自己的超链接指针等等。
- 强制进行计算，使得用于接收计算结果的存储空间上溢。
- 挑选某功能部件，思考介入它的其他功能部件。

123

示例1：无效操作数和操作符结合

124

示例2：递归调用

看一段C程序段：

```
long int test(int n)
{
  if (n<=1)
    return (1);
  else
    return (n*test(n-1));
}
```

修改后C程序段：

```
long int test(int n)
{
  return (n*test(n-1));
}
```

125

示例3：强制计算结果过大或过小

```
const count 2
main() {
  int sum,value[count];
  sum=0;
  for (i=0;i<count;++i) {
    sum=sum+value[i]
  }
}
```

输入：
value[0]=32700
value[1]=70

count=33000
value[0..32999]=1

126

自动化测试工具

- 自动化测试工具的特征
- WinRunner演示

自动化测试工具的特征

- 支持脚本化语言
- 对程序界面中对象的识别能力
- 支持函数的可重用
- 支持外部函数库
- 抽象层
- 分布式测试的支持

自动化测试工具的特征

- 支持数据驱动测试
- 错误处理
- 调试器
- 源代码管理
- 支持脚本的命令行方式执行

WinRunner

- 基于MS Windows的功能测试工具。
- WinRunner测试模式：

环境判断模式（Context Sensitive mode）

模拟模式(Analog mode)

WinRunner测试过程

- 创建GUI map
- 创建测试
- 调试测试
- 执行测试
- 查看测试结果
- 报告发现的错误