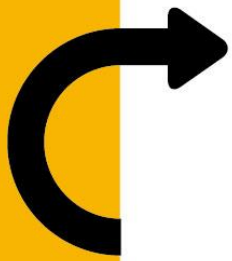
测试案例设计（PDF测试岗位课程）



第一部分 测试案例基础概念

第二部分 测试案例设计方法

第三部分 测试案例设计流程

什么是软件测试

- **软件测试的定义：**软件测试是在规定的条件下对程序进行操作，以发现错误，对软件质量进行评估
  - 即软件测试是为了发现错误而执行程序的过程。
  - 软件测试的目的：不仅是为了发现软件缺陷与错误，是软件质量保证的关键，是对软件质量进行度量与评估，判断风险，以提高软件质量

测试活动

- 所有的测试活动集中在以下四个方面
  - 测试需求分析（RTVM)
  - 测试案例设计（测试案例）
  - 测试案例执行（测试执行跟踪表）
  - 测试结果分析（测试报告）

- 测试案例设计和编写直接影响到测试的有效性和效率

- 本课程主要介绍黑盒测试案例设计方法，包括用例场景法、等价类划分法、边界值分析法、决策表分析法、正交阵列设计法等。其中重点介绍用例场景法。

- 这些设计方法是测试中比较常用的，但具体采用哪种方法，要针对开发项目的特点选择适当的方法。

为什么需要测试案例

因为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挑细选出具有代表性或特殊性的数据来进行测试。

使用测试案例的好处包括下面四个方面：

1.在开始实施测试之前设计好测试案例，可以避免盲目测试并提高测试效率。

2.测试案例的使用令软件测试的实施重点突出、目的明确。

3.在软件版本更新之后只需要修正部分测试案例便可展开测试工作，降低工作强度，缩短项目周期。

4.功能模块的通用化和复用化使软件易于开发，而测试案例的通用化和复用化则使软件测试易于开展，并随着测试案例的不断优化其效率也不断提升。

什么是测试案例

- **测试案例：**是为特定的目的而设计的一组测试输入、执行条件和预期结果。测试案例是执行的最小实体。简单地说，测试案例就是设计一个场景，使软件程序在这种场景下，必须能够正常运行并且达到程序所设计的执行结果。

编写测试案例的思路是什么

测试案例是为**特定的目的**而设计的一组**测试输入**、**执行条件**和**预期结果**。

测试案例是**执行的最小实体**。

简单地说，测试案例就是设计一个**场景**，使软件程序在这种场景下，必须能够正常运行并且**达到程序所设计的执行结果**。

- 测试目标
- 操作步骤和测试数据
- 前提条件
- 测试的期望结果

- 是测试执行的最小实体

- 设计测试场景以满足需求

设计测试案例的原则

- 测试案例的**代表性**：能够代表并覆盖各种合理的和不合理、合法的和非法的、边界的和越界的、以及极限的输入数据、操作和环境设置等

- 测试结果的**可判定性**：测试案例执行结果的正确性是可判定的，每一个测试案例都应有相应的期望结果

- 测试结果的**可再现性**：同样的测试案例，系统的执行结果应当是相同的

测试案例的特点

设计测试案例时要考虑：

- **简洁性：**清晰明了，即不拖沓冗长，也不过于简单难以执行
- **完整性：**编写的案例要对系统（产品）有足够的覆盖
- **有效性：**较少的案例覆盖较多的测试区域，对于发现缺陷最有作用
- **唯一性：**每条测试案例都有唯一的测试目的
- **可维护性：**测试案例便于更新和维护

测试案例的分级管理

为什么要进行测试案例的分级管理：

- 在项目的生命周期里，在每一个版本上都执行你全部的测试案例是很困难的。但是你如何知道哪个测试案例必须在每一个版本中执行，什么应该被执行，同时如果你有时间的话，什么又可以被执行？在**快速组织测试案例、安排测试进度和工作量、制订项目计划时需要完成哪些测试案例**等方面，测试案例的分级管理可以给你很多帮助。
- 怎样划分测试案例优先级和如何执行测试案例取决于你在你的项目周期的位置。当你在测试进行的过程中，通过分析和度量，发现风险和缺陷集中出现的地方时，你可能会调整你的测试案例优先级别。这样会让你的测试目标和测试重点变得更清晰。
- 拥有划分了优先级别的测试案例，也为你潜在的，待定的自动化项目给出了一个好的起点。比如，自动化BVT（版本确认测试，又叫冒烟测试）中的测试案例，用于持续集成的自动化验收；或者自动化高、中优先级的测试案例，用于全量的功能自动化测试。

测试案例的分级管理

IEEE定义的标准的测试案例优先级如下：

- 版本确认测试（Build Verification Tests (BVTs)：也叫做“冒烟测试”，一组你想优先运行以确定版本是否可以测试的测试案例。如果你不能访问每一个功能区域或执行其他测试案例依赖的基本操作，那么在执行这组测试案例之前，试图做其他任何的测试都是没有意义的，因为他们大多数肯定要失败。
- 高（Highs）：最常执行以保证功能稳定的，目标的行为和能力可以正常的工作，以及重要的错误和边界被测试的测试案例的集合。
- 中（Mediums）：这是使给出的功能区域或功能变得更详细，检查功能的多数方面包括边界值，等价类，错误和配置测试的测试案例
- 低（Lows）：这是通常最少被执行的测试案例。但这并不意味着这些测试都不重要，只是说他们在项目的生命期间里不是常常被运行，例如GUI、错误信息、可用性、稳定性、易用性、压力测试和性能测试。

测试案例的分级管理

我们公司对测试案例的分级管理：

- **Level 0：**基本的功能性验证（或基本路径（Basic Path））的测试案例，以及少量重要的错误、边界值或重要的路径分支。适用于做确认测试（BVT），比例为15-20%;
- **Level 1：**对基本功能或基本路径的扩充，包含错误、边界值、等价类、路径分支等测试案例。适用于全量覆盖完整的需求功能点，比例为60-70%;
- **Level 2：**非功能性的（例如错误信息、可用性、稳定性、易用性、压力测试和性能测试）测试案例。适用于辅助完善测试覆盖，比例为10-15%。

如何评价测试案例的好坏

- **易用性**。对于一个即不熟悉测试工作，又不熟悉被测应用的测试人员，只需花费很少的时间就可以理解测试案例中表达的测试思路，并可以很快的执行完成

- **易维护性**。当开发过程中的某些因素影响了测试需求，测试案例的作者或其他测试设计人员，应该可以花费很少的时间就完成定位并维护所有相关测试案例的工作

第一部分 测试案例基础概念

第二部分 测试案例设计方法

第三部分 测试案例设计流程

测试阶段

测试阶段划分如下：

- 单元测试（Unit Testing）
- 集成测试（Integration Testing）
- 系统测试（System Testing）
- 系统集成测试（System Integration Testing）
- 用户验收测试（User Accept Testing）

测试方法

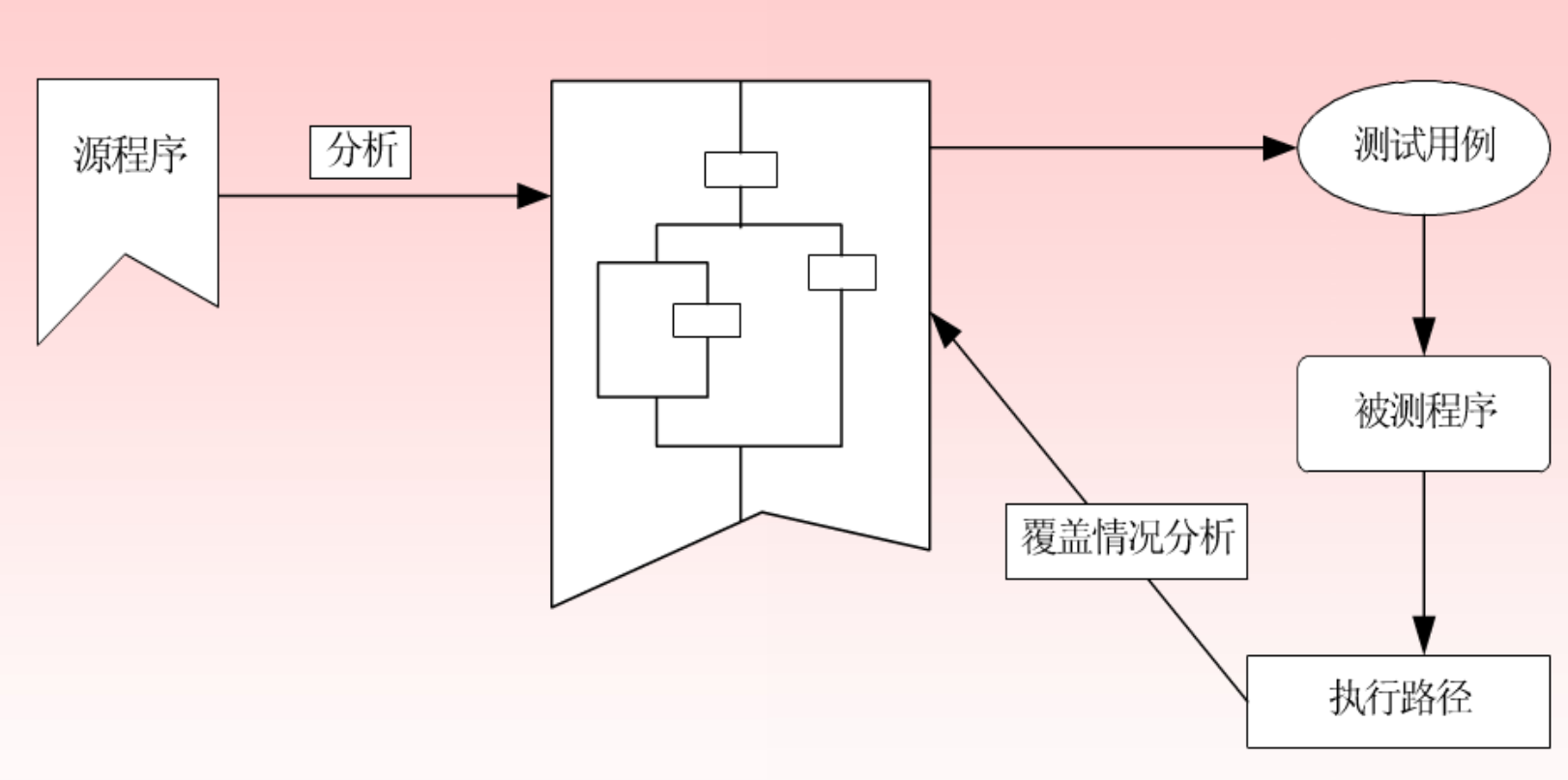
测试方法分类如下：

- **静态测试（不运行被测程序）**
  评审、走查、审查、桌面检查

- **动态测试（运行被测程序）**
  白盒测试、黑盒测试、错误猜测

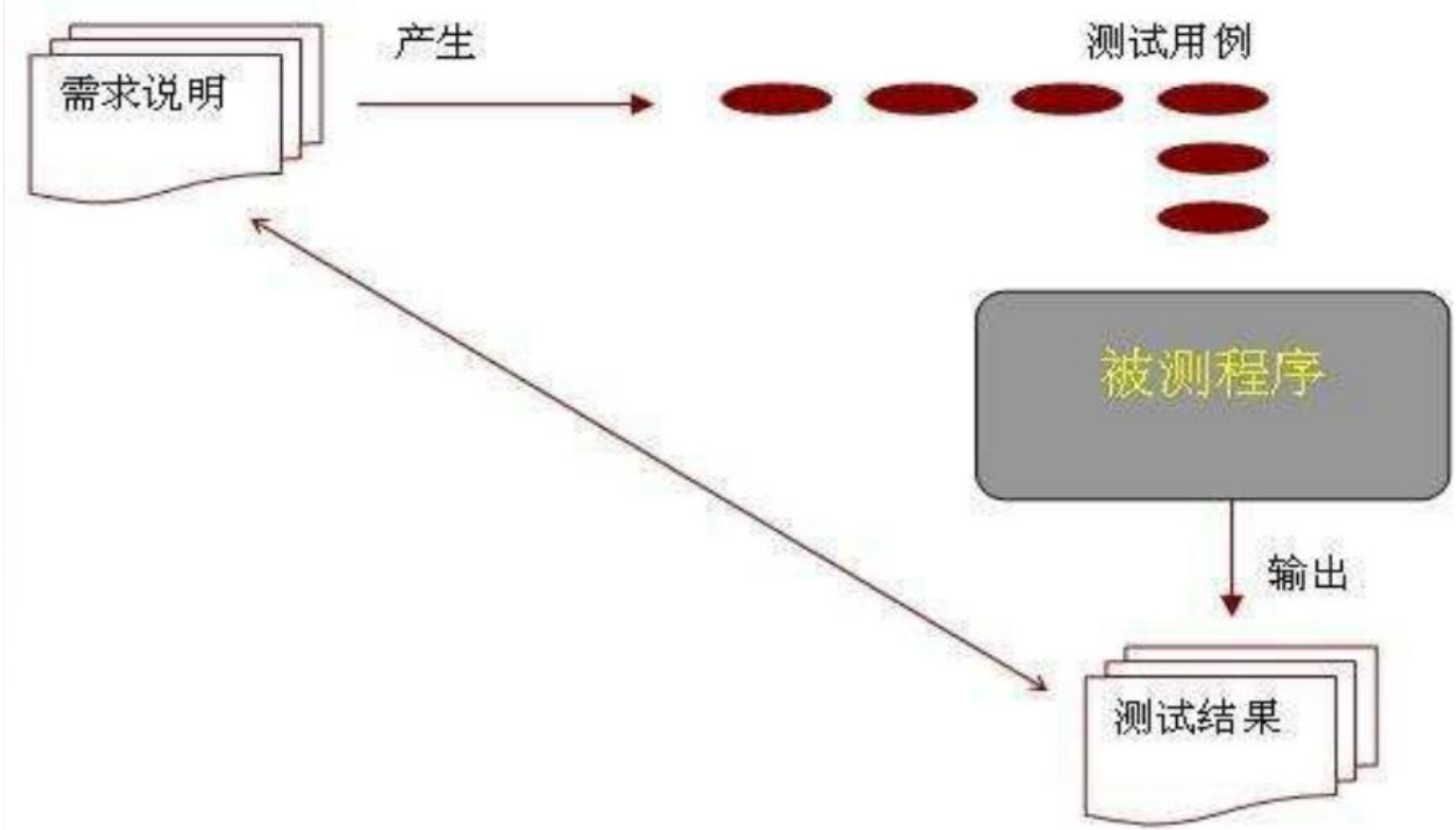
白盒测试介绍

白盒测试（是通过程序的源代码进行测试而不使用用户界面）



黑盒测试介绍

黑盒测试（已经实现的功能是否符合要求）



测试案例的常用编写方法

- 场景法
- 等价类
- 边界值
- 因果图
- 判定表
- 正交阵列
- 随机测试

场景法

- 场景法适用于什么样的项目？业务流程或事件比较复杂的程序，主要用来探索对于比较有经验的用户是怎么来使用软件的，并查找出更加有说服力的**bug**。不同的触发顺序和处理结果形成事务流，通过设计足够多的测试用例来覆盖基本流和各种备选流（如oa请假系统）
- 场景法一般包含基本流和备选流来完成整个场景，下图展示了场景法基本情况的一个实例图

  基本流：采用直黑线表示，是经过用例的最简单路径（开始直接执行到结束）

  备选流：采用不同颜色表示，一个备选流可能从基本流开始，在某个特定条件下执行，然后重新加入基本流；也可以起源于另一个备选流，或终止用例，不再加入到基本流中（各种错误情况）

场景法

场景法的基本设计步骤：

1. 根据说明，描述出程序的基本流及各项备选流
2. 根据基本流和各项备选流生成不同的场景
3. 对每一个场景生成相应的测试案例
4. 对生成的所有的测试案例重新复审，去掉多余的测试案例，测试案例确定后，对每个测试案例确定测试数据值

举例：用户进入一个网站进行手机充值，需要登录账号，登录成功后，进行付钱交易，交易成功后，完成充值

步骤1，确定基本流和备选流

基本流	打开该网站，登录账号，输入需要充值的手机号，付钱交易，充值成功
备选流1	登录账号或者密码输入错误
备选流2	登录账号状态异常
备选流3	需要充值的手机号码输入非法
备选流4	账号余额不足

场景法

步骤2，根据基本流和备选流来确定场景

场景1-成功完成充值	基本流	
场景2-登录账户或者密码输入错误	基本流	备选流1
场景3-登录账户状态异常	基本流	备选流2
场景4-充值手机号输入不正确	基本流	备选流3
场景5-账户余额不足	基本流	备选流4

场景法

步骤3，设计测试案例

案例ID	场景/条件	账号	密码	手机号码	账户余额	预期结果
1	场景1：成功完成手机充值	Y	Y	Y	Y	成功充值
2	场景2：登录账号错误	N	-	-	-	提示账号不存在
3	场景2：登录密码错误	Y	N	-	-	提示密码输入错误
4	场景3：登录账号状态异常	N	Y	-	-	提示账号状态异常
5	场景4：充值手机号输入不正确	Y	Y	N	-	提示手机号码输入不正确
6	场景5：账户余额不足	Y	Y	Y	N	提示账户余额不足

场景法

步骤4 设计数据，把数据填写入上面的案例表中

案例ID	场景/条件	账号	密码	手机号码	账户余额	预期结果
1	场景1：成功完成手机充值	Jim	123ok	13800000000	Y	成功充值
2	场景2：登录账号错误	Tom	-	-	-	提示账号不存在
3	场景2：登录密码错误	Jim	12	-	-	提示密码输入错误
4	场景3：登录账号状态异常	Lily	14q	-	-	提示账号状态异常
5	场景4：充值手机号输入不正确	Jim	123ok	123	-	提示手机号码输入不正确
6	场景5：账户余额不足	Sam	88u	13700000000	0	提示账户余额不足

测试案例的常用编写方法

- 场景法
- 等价类
- 边界值
- 因果图
- 判定表
- 正交阵列
- 随机测试

等价类

- 等价类划分是一种典型的黑盒测试方法，用这一方法设计测试案例完全不考虑程序的内部结构，只根据对程序的需求和说明，即需求规格说明书。
- 由于穷举测试工作量太大，以致于无法实际完成，促使我们在大量的可能数据中选取其中的一部分作为测试案例。

- **定义：等价类比法**是把所有可能的输入数据,即程序的输入域划分成若干部分（子集），然后从每一个子集中选取少数具有代表性的数据作为测试案例。

- 划分等价类：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，并合理地假定：测试某等价类的代表值就等于对这一类其它值的测试。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。

  - **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。**利用有效等价类可检验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：**与有效等价类的定义恰巧相反。

等价类

输入条件	有效等价类	无效等价类
…	…	…
…	…	…

- 划分等价类的标准：
  1)完备测试、避免冗余
  2)划分等价类重要的是：集合的划分，划分为互不相交的一组子集
  3)子集的并是整个集合：完备性
  4)子集互不相交：保证一种形式的无冗余性
  5)同一类中标识（选择）一个测试案例，同一等价类中，往往处理相同，相同处理映射到“相同的执行路径”

- 设计测试案例
  在确立等价类后,可建立**等价类表**

等价类

- 每个等价类中只测试一个值
- 如果由于某种原因需要重复测试，可以选取测试等价类中不同的值来测试

- 根据等价类表，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，然后从划分出的等价类中按以下三个原则设计测试案例：
  1)为每一个等价类规定一个唯一的编号；
  2)设计一个新的测试案例,使其尽可能多地覆盖尚未被覆盖地有效等价类,重复这一步，直到所有的有效等价类都被覆盖为止；
  3)设计一个新的测试案例,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。

等价类

例1.输入在0到40之间整数.

有效等价类

0<= X <= 40

无效等价类

X < 0

X > 40

0到40之间的非整数

字母、汉字、特殊字符

例 2. 第一个字符是字母.

有效等价类

26个字母中的任何一个

无效等价类

特殊字符

数字字符

汉字字符

等价类-练习

根据下面给出的规格说明，利用等价类划分的方法，给出足够的测试案例。

“一个程序读入3个整数，把这三个数值看作一个三角形的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。”

- 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。

请画出等价类表

等价类-练习

输入条件	有效等价类	无效等价类
是否三角形的三条边	（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）

等价类-练习

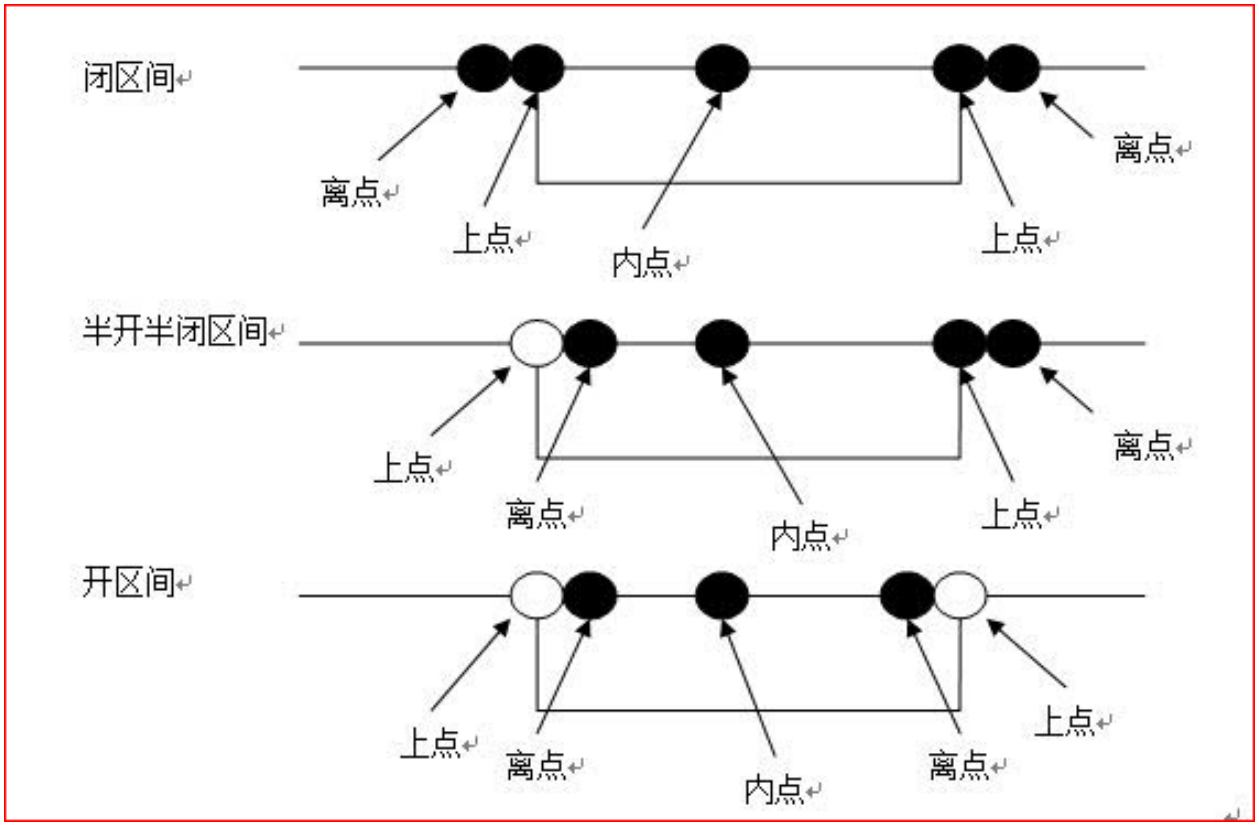
序号	【A，B，C】	覆盖等价类	输出
1	【3，4，5】	（1），（2），（3），（4），（5），（6）	一般三角形
2	【0，1，2】	（7）	不能构成三角形
3	【1，0，2】	（8）	
4	【1，2，0】	（9）	
5	【1，2，3】	（10）	
6	【1，3，2】	（11）	
7	【3，1，2】	（12）	
8	【3，3，4】	（1），（2），（3），（4），（5），（6），（13）	等腰三角形
9	【3，4，4】	（1），（2），（3），（4），（5），（6），（14）	
10	【3，4，3】	（1），（2），（3），（4），（5），（6），（15）	
11	【3，4，5】	（1），（2），（3），（4），（5），（6），（16）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

测试案例的常用编写方法

- 场景法
- 等价类
- 边界值
- 因果图
- 判定表
- 正交阵列
- 随机测试

边界值分析

定义：边界值分析法就是对输入或输出的边界值进行测试的一种黑盒测试方法。通常边界值分析法是作为对等价类划分法的补充，这种情况下，其测试案例来自等价类的边界。



边界值分析

- 与等价划分的区别
  1)边界值分析不是从某等价类中随便挑一个作为代表，而是使用这个等价类的每个边界都要作为测试条件。
  2)边界值分析不仅考虑输入条件，还要考虑输出结果产生的测试情况。

使用边界值分析方法的原因：

- 长期的测试工作经验告诉我们，大量的错误是发生在输入或输出范围的边界上，而不是发生在输入输出范围的内部。因此针对各种边界情况设计测试案例，可以查出更多的错误
- 使用边界值分析方法设计测试案例，首先应确定边界情况。通常输入和输出等价类的边界，就是应着重测试的边界情况。应当选取正好等于，刚刚大于或刚刚小于边界的值作为测试数据，而不是选取等价类中的典型值或任意值作为测试数据
  - 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。
  - 正确的软件通常应该将输入内容默认为合法边界内的最小值或者合法区间内某个合理值，否则返回错误提示信息。

边界值分析-练习

举例: 测试密码是6位到16位之间的阿拉伯数字

- 等价类法: x<6;6 ≤x ≤16;x>16
- 边界值法: 5,6,11,16,17

5 个测试案例：

- 6 位数字 (Success)
- 5位数字 (Error)
- 11位数字 (Success)
- 16位数字 (Success)
- 17位数字 (Error)

边界值分析-练习

举例: 1<X<10,1<=Y<=10,X与Y都是整数

- X 边界值法: ? -Y 边界值法: ?

-X 5 个测试案例：

- 2 (Success)
- 1 (Error)
- 6 (Success)
- 9 (Success)
- 10 (Error)

-Y 5 个测试案例：

- 1 (Success)
- 0 (Error)
- 6 (Success)
- 10 (Success)
- 11 (Error)

测试案例的常用编写方法

- 场景法
- 等价类
- 边界值
- 因果图
- 判定表
- 正交阵列
- 随机测试

因果图

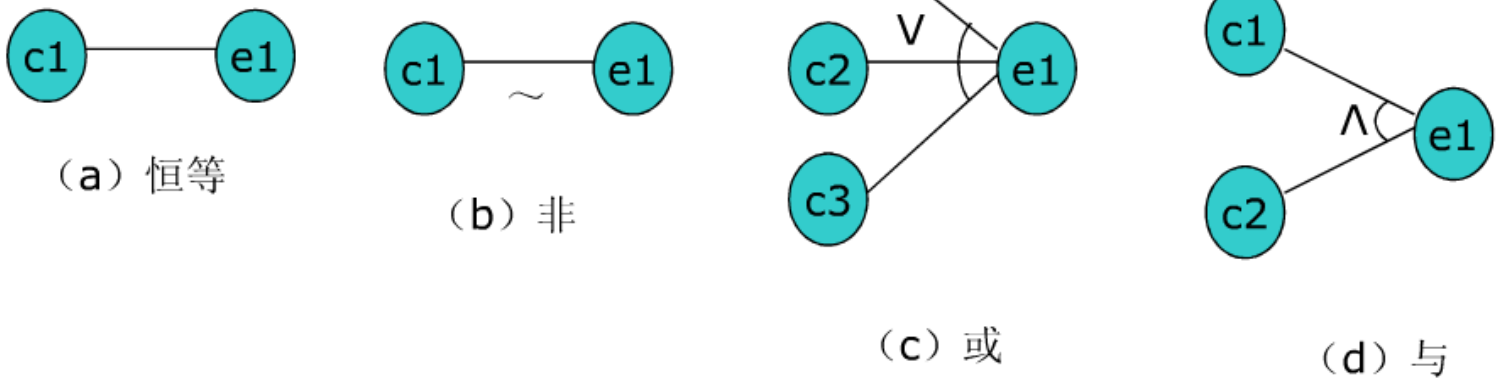
- 等价类划分法和边界值分析方法都是着重考虑输入条件，但是没有考虑输入条件的各种组合、输入条件的各种制约关系。这样虽然各种输入条件出错的情况已经测试到了，但是多个输入条件组合起来可能出错的情况却被忽视了。

- **定义：因果图法**是一种利用图解法分析输入的各种组合情况，从而设计测试案例的方法，它使用于检查程序输入条件的各种组合情况。因果图法一般和判定表法结合在一起使用
- 因果图中出现的基本符号

通常在因果图中用**Ci**表示原因，用**Ei**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

- 恒等：若**c1**是**1**，则**e1**也为**1**，否则**e1**为**0**；
- 非：若**c1**是**1**，则**e1**为**0**，否则**e1**为**1**；
- 或：若**c1**或**c2**或**c3**是**1**，则**e1**是**1**，否则**e1**为**0**，“或”可有任意个输入；
- 与：若**c1**和**c2**都是**1**，则**e1**为**1**，否则**e1**为**0**，“与”也可有任意个输入。



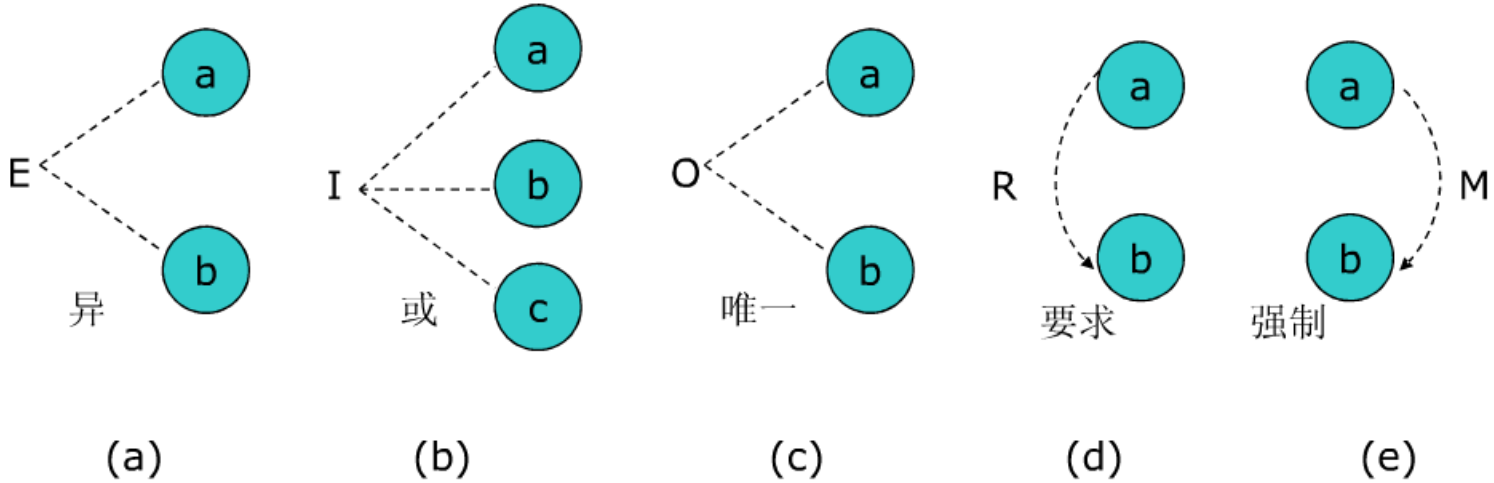
因果图

对于输入条件的约束有**4**种：

- **E**约束（异）：**a**和**b**中最多有一个可能为**1**，即**a**和**b**不能同时为**1**；
- **I**约束（或）：**a**、**b**、**c**中至少有一个必须是**1**，即**a**、**b**、**c**不能同时为**0**；
- **O**约束（唯一）：**a**和**b**必须有一个且仅有一个为**1**；
- **R**约束（要求）：**a**是**1**时，**b**必须是**1**；

对于输出条件的约束只有**M**约束

- **M**约束（强制）：若结果**a**是**1**，则结果**b**强制为**0**。

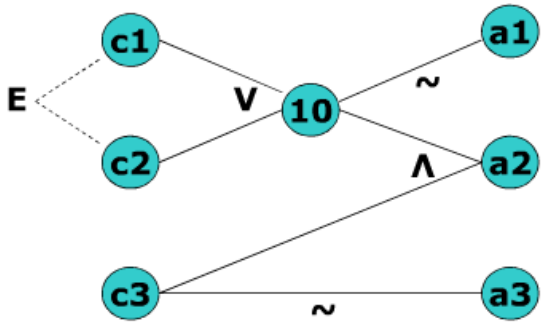


因果图

- 程序的规格说明需求，输入的第一个字符必须是“#”或者“*”，第二个字符必须是一个阿拉伯数字，在此情况下进行文件的修改。如果第一个字符不是“#”或者“*”，则给出信息N；如果第二个字符不是数字，则给出信息M
- **步骤1**： 分析程序的规格说明，列出原因和结果：

  原因：c1-第一个字符是“#”，c2-第一个字符是“*”，c3-第二个字符是一个阿拉伯数字

  结果：a1-给出信息N，a2-修改文件，a3-给出信息M
- **步骤2**：找出原因与结果之间的因果关系、原因与原因之间的约束关系，画出因果图



10是中间状态，输入字符

因果图

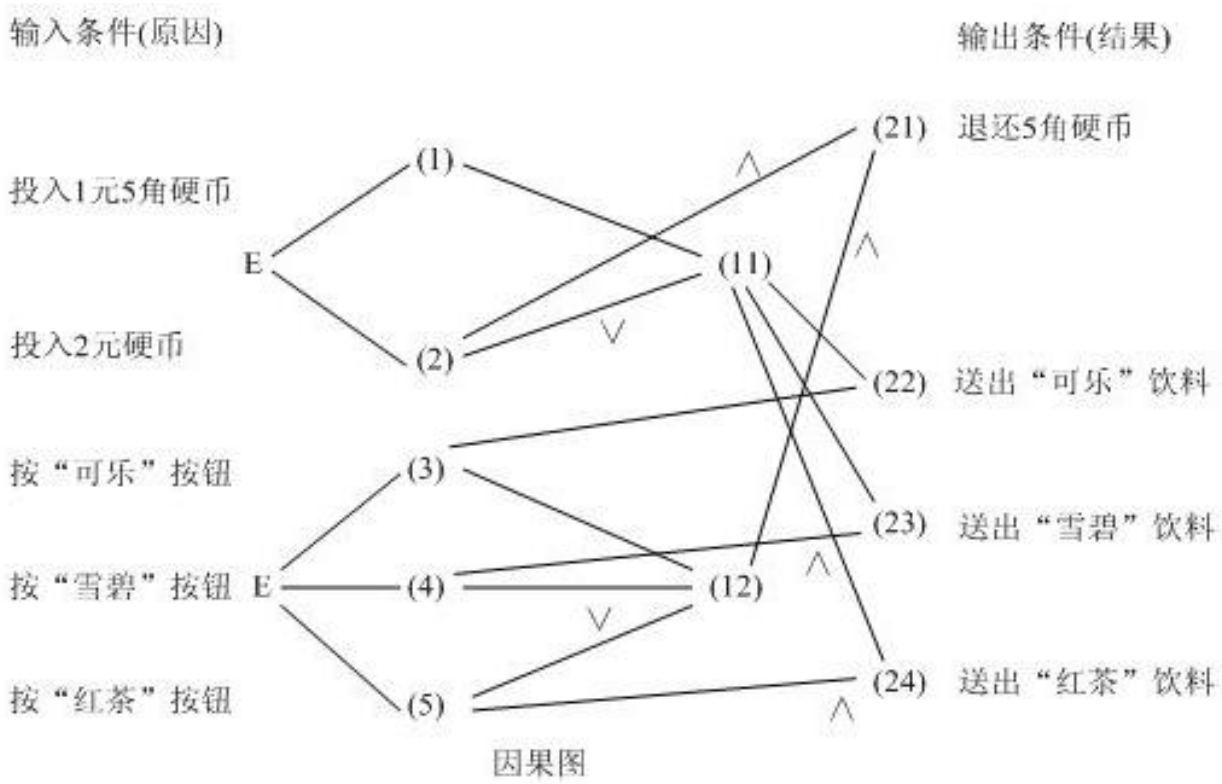
- **步骤3**：将因果图转换成判定表

	1	2	3	4	5	6	7	8
C1	1	1	1	1	0	0	0	0
C2	1	1	0	0	1	1	0	0
C3	1	0	1	0	1	0	1	0
10			1	1	1	1	0	0
a1							√	√
a2			√		√			
a3				√		√		√
不可能	√	√						
测试用例			#3	#B	*7	*M	C2	CM

- **步骤4**：设计测试案例
  - **Test1**：输入数据－**#3**　预期输出－－修改文件
  - **Test2**：输入数据－**#B**　预期输出－－给出信息**M**
  - **Test3**：输入数据－***7**　预期输出－－修改文件
  - **Test4**：输入数据－***M**　预期输出－－给出信息**M**
  - **Test5**：输入数据－**C2**　预期输出－－给出信息**N**
  - **Test6**：输入数据－**CM**　预期输出－－给出信息**M**和**N**

因果图-练习

有一个处理单价为1元5角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、雪碧或者红茶按钮，相应的饮料就送出来。若投入的是两元硬币，在送出饮料的同时退5角硬币



因果图

因果图-练习

将因果图转换成判定表

			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	1	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

测试案例的常用编写方法

- 场景法
- 等价类
- 边界值
- 因果图
- 判定表
- 正交阵列
- 随机测试

判定表

- 定义：判定表（也叫决策表）是分析和表达多逻辑条件下执行不同操作的情况的工具。

- 判定表的优点
  - 能够将复杂的问题按照各种可能的情况全部列举出来，简明并避免遗漏。因此，利用判定表能够设计出完整的测试案例集合。
  - 在一些数据处理问题当中，某些操作的实施依赖于多个逻辑条件的组合，即：针对不同逻辑条件的组合值，分别执行不同的操作。判定表很适合于处理这类问题。

判定表

适合使用判定表设计测试案例的条件：

- 规格说明以判定表形式给出，或很容易转换成判定表
- 条件的排列顺序不影响执行操作
- 规则的排列顺序不影响执行操作
- 每当某一规则的条件已经满足，并确定要执行的操作后，不必检验别的规则
- 如果某一规则得到满足要执行多个操作，这些操作的执行顺序无关紧要

判定表

- 问题要求：”……对功率大于50马力的机器、维修记录不全或已运行10年以上的机器，应给予优先的维修处理，否则做其他处理……” 。这里假定，“维修记录不全”和“优先维修处理”均已在别处有更严格的定义 。请建立判定表。

  - 1）确定规则的个数：这里有3个条件，每个条件有两个取值，故应有2*2*2=8种规则。
  - 2）列出所有的条件茬和动作茬：

条件	功率大于50马吗？
	维修记录不全吗？
	运行超过10年了吗？
动作	进行优先处理
	作其它处理

判定表

- 3）填入条件项。（二进制的真值表）。
- 4）填入动作桩和动作顶。这样便得到形如图的初始判定表。

		1	2	3	4	5	6	7	8
条件	功率大于 50 马力吗？	Y	Y	Y	Y	N	N	N	N
	维修记录不全吗？	Y	Y	N	N	Y	Y	N	N
	运行超过 10 年吗？	Y	N	Y	N	Y	N	Y	N
动作	进行优先处理	x	x	X		X		X	
	作其他处理				X		x		x

初始判定表

判定表

		1	2	3	4	5	6	7	8
条件	功率大于 50 马力吗?	Y	Y	Y	Y	N	N	N	N
	维修记录不全吗?	Y	Y	N	N	Y	Y	N	N
	运行超过 10 年吗?	Y	N	Y	N	Y	N	Y	N
动作	进行优先处理	x	x	X		X		X	
	作其他处理				X		x		x

初始判定表

		1	2	3	4	5
条件	功率大于 50 马力吗?	Y	Y	Y	N	N
	维修记录不全吗?	Y	N	N	-	-
	运行超过 10 年吗?	-	Y	N	Y	N
动作	进行优先处理	x	x		X	
	作其他处理			x		x

化减后的判定表

测试案例的常用编写方法

- 场景法
- 等价类
- 边界值
- 因果图
- 判定表
- 正交阵列
- 随机测试

正交阵列

- 方法简介

利用因果图来设计测试案例时, 作为输入条件的原因与输出结果之间的因果关系, 有时很难从软件需求规格说明中得到。往往因果关系非常庞大,以至于据此因果图而得到的测试案例数目多的惊人，给软件测试带来沉重的负担，为了有效地, 合理地减少测试的工时与费用,可利用正交实验设计方法进行测试案例的设计。

- 正交实验设计方法: 依据Galois理论,从大量的（实验）数据（测试例）中挑选适量的,有代表性的点（例）,从而合理地安排实验（测试）的一种科学实验设计方法.类似的方法有:聚类分析方法,因子方法方法等.

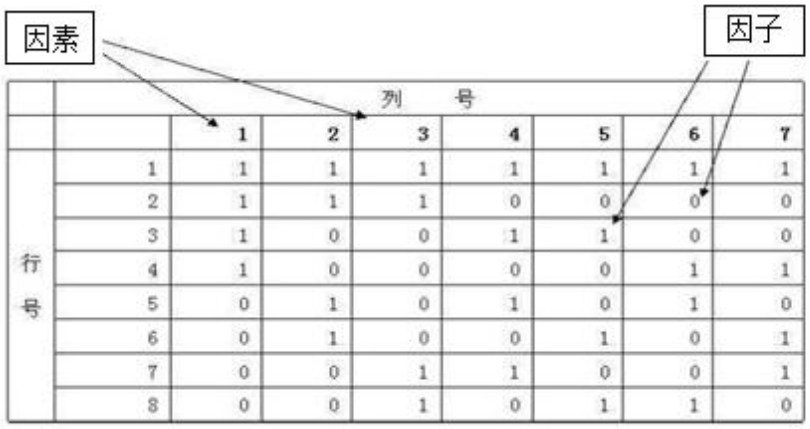
正交阵列

- 行数（Runs）：正交表中的行的个数，即实验的 次数，也就是我们通过正交实验法设计的测试案例的个数
- 因素数（Factors）：正交表中列的个数，即我们要测试的功能点
- 因子数（Levels）：任何单个因素能取得的值的最大个数。

正交表的形式

- $L_{行数}$(因子数 因素数)
- 如：$L_8(2^7)$

行数 = 因素数 *（ 因子数 － 1 ）+ 1



		1	2	3	4	5	6	7
行号	1	1	1	1	1	1	1	1
	2	1	1	1	0	0	0	0
	3	1	0	0	1	1	0	0
	4	1	0	0	0	0	1	1
	5	0	1	0	1	0	1	0
	6	0	1	0	0	1	0	1
	7	0	0	1	1	0	0	1
	8	0	0	1	0	1	1	0

测试案例的常用编写方法

- 场景法
- 等价类
- 边界值
- 因果图
- 判定表
- 正交阵列
- 随机测试

随机测试

经验 + 技术 = 专家

- 在软件测试中除了根据测试样例和测试说明书进行测试外，还需要进行随机测试(Ad-hoc testing)，主要是根据测试者的经验对软件进行功能和性能抽查。随机测试是对案例测试的重要补充手段，是保证测试覆盖完整性的有效方式和过程。
- 随机测试主要是对被测软件的一些重要功能进行复测，也包括当前的测试案例(TestCase)没有覆盖到的部分。对于软件更新和新增的功能要重点测试。尤其对一些特殊情况、特殊使用环境、并发性，进行检查。对以前测试发现的重大Bug，进行再次测试，可以结合回归测试(Regressivetesting)一起进行。

第一部分 测试案例基础概念

第二部分 测试案例设计方法

第三部分 测试案例设计流程

编写测试案例的流程

STEP BY STEP 方法

- Step1 需求分析与确认
- Step2 根据需求规格说明书列出所有功能点
- Step3 矩阵列表形式列出需求功能点和案例的对应关系
- Step4 利用一系列输入值增加测试类型
- Step5 编写或者修改测试案例
- Step6 测试案例评审

需求分析与确认

- 测试需求的依据与收集
  - 与待测软件相关的各种文档资料
  - 与客户或系统分析员的沟通
  - 业务背景资料
  - 正式与非正式培训
  - 其他
- 测试需求分析时需要列出以下类别
  - 常用的或规定的业务流程
  - 各业务流程分支的遍历
  - 明确规定不可使用的业务流程
  - 没有明确规定但是不应该执行的业务流程
  - 其他异常或不符合规定的操作

根据需求列出所有功能点

- 下面是一个业务需求模版和需求样例，我们可以对其中需求样例进行功能点罗列：

业务需求模版和需求样例

FMS_BR_DGFK_01.01	对公付款_制单-单笔付款	高
FMS_BR_DGFK_01.02	对公付款_制单-批量付款	高
FMS_BR_DGFK_01.03	对公付款_制单-汇总付款	高
FMS_BR_DGFK_01.04	对公付款_制单-合并付款	中
FMS_BR_DGFK_01.05	对公付款_制单-普通代付	高
FMS_BR_DGFK_01.06	对公付款_制单-批量导入	中

列出功能点与案例的对应关系

需求跟踪矩阵（RTVM）

需求跟踪矩阵(RTVM)

需求		是否高风险需求	系统测试	
需求编号	需求描述		测试案例编号	测试案例名称
FMS_BR_DGFK_01.01	对公付款_制单-单笔付款	Y	FMS_TC_DGFK_001	(+)单笔制单选择全额付款；
FMS_BR_DGFK_01.01	对公付款_制单-单笔付款	Y	FMS_TC_DGFK_002	(+)单笔制单选择部分付款；
FMS_BR_DGFK_01.01	对公付款_制单-单笔付款	Y	FMS_TC_DGFK_003	(-)单笔付款中选择多笔明细进行制单，报错“此界面只能单笔制单”；
FMS_BR_DGFK_01.01	对公付款_制单-单笔付款	Y	FMS_TC_DGFK_004	(-)在单笔付款中未选择支付单据，系统报错“需选择一笔业务单据方可制单”；
FMS_BR_DGFK_01.01	对公付款_制单-单笔付款	Y		
FMS_BR_DGFK_01.02	对公付款_制单-批量付款	Y		
FMS_BR_DGFK_01.03	对公付款_制单-汇总付款	Y		
FMS_BR_DGFK_01.04	对公付款_制单-合并付款	N		
FMS_BR_DGFK_01.05	对公付款_制单-普通代付	Y		
FMS_BR_DGFK_01.06	对公付款_制单-批量导入	N		

利用一系列输入值增加测试类型

- 构造测试数据，或者其他输入条件，丰富和完善测试类型。以前面的需求为例，我们在构造测试数据等输入条件的时候，可以考虑以下情况：

  1、构造一笔全额付款的数据；

  2、构造一笔部分付款的数据；

  3、构造多笔付款数据；

  4、构造一笔超额的付款数据；

  5、制单过程中死机；

  6、制单过程中网络断连；

  。 。 。 。 。

编写或修改测试案例

编写测试案例有两种情况：

- 如果需求是新功能，则新增测试案例
- 如果需求是老功能，只是做优化，则在案例库中寻找之前该需求对应的最新的案例，做修改或增补

测试案例模版

测试案例评审

测试人员编写完一个完整的功能模块的测试案例后，需要对测试案例进行评审，评审一般分两步：

- 同行评审：把测试案例发给参与评审的人员进行线下评审，评审完成后收集汇总评审意见，进行测试案例的修改；
- 评审会议：召开测试案例评审会议，讨论评审结果和建议，评审会议结束后，根据评审意见修改测试案例，最后把修改完成的测试案例发给参与评审的人员复核。

测试案例评审表

评审参考案例评审检查表

课程回顾

- 测试案例的定义、原则、意义、特点
- 常用的测试案例设计方法：等价类、边界值、因果图、判定表 ...
- 测试案例的设计流程，编写规范是什么 ...

Thanks!