题目类型：一、 判断选择题（每题一分）二、简单论述题 三、应用题

第一章 软件测试概述

fault、error、failure给出一段代码，分析fault error failure的含义以及通过输入什么样测测试用例产生fault error failure的不同表现（4种表现）

* **fault、error、failure的含义**

fault: 软件中的静态缺陷

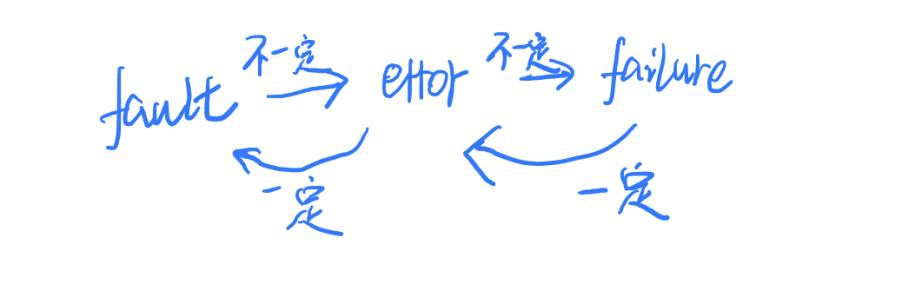
error: 错误的内部状态，是某种错误的表现，中间过程与理想的运行情况出现了偏差。

failure: 外部的，与要求或其他预期行为描述相关的不正确行为

测试人员找failure，开发人员找fault

* **4种表现：**

1. 未执行到falut
2. 执行到fault，没有产生error（没有error也就没有failure）
3. 执行到fault，产生了error，没有产生failure
4. 执行到了fault，产生了failure，产生了failure



* **PIE模型：**

**PIE模型是指执行，感染，传播三个必要条件，用于观测到failure。**

**要观测到一个failure要三个必要条件:**

**1、执行【Execution】:运行必须通过错误的代码**

**2、感染【Infection】:执行错误代码时候必须触发一个错误的中间状态**

**3、传播【Propagation】:这个错误的中间状态必须传播到最后输出，使得我们能观测到它跟预期的输出不一致即失效**

**总结：bug分为fault，error，failure，整个能够传播出去是靠PIE模型，即执行到fault，感染产生error，并且传播出去我们观测到失效的行为。**



软件测试的一些基本概念：

* 静态测试和动态测试的区别

静态测试主要包括：

1. 代码检查：代码会审、代码走查、桌面检查；
2. 静态结构分析；

（3）代码质量度量。

动态测试主要包括：

①黑盒测试：又称功能测试或数据驱动测试，是针对软件的功能需求/实现进行测试。穷举输入测试

②白盒测试：也称结构测试或逻辑驱动测试，必须知道软件内部工作过程，通过逻辑覆盖、路径覆盖等方式选择测试用例，可以用测试覆盖率评价测试用例。

注意：白盒测试是基于结构，黑盒测试是基于功能；动态测试需要执行，静态测试不需要执行代码。

* Validation& Verification

Validation（确认）是软件开发结束的时候

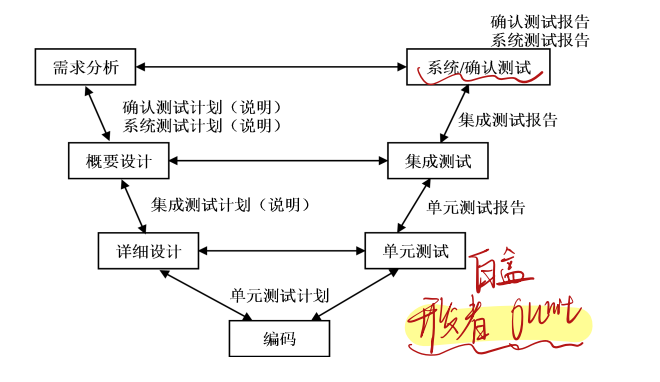
Verification（验证）是软件开发过程中

* Test & Debug

Test是测试人员找failure

debug是开发人员找fault

* V模型示意图



需求分析，概要设计，详细设计 自己描述

①单元测试：

1.完成对最小软件设计单元——模块的测试

2.能正确编码，通常是白盒测试

3.对一些功能进行的静态检查。

测试的内容：接口、数据结构、边界、语句覆盖、错误路径

②集成测试：

1.发现与模块接口有关的问题。

2.把模块取来，构造一个在设计中描述的程序结构。

3.应当避免一次性的集成（除非软件规模很小），而采用增量集成

③系统测试：

1.测功能/性能是否得到满足。

2.容错性、安全性等需求。

④验收/确认测试：

1. 配置审查: – 确保已开发软件的所有文件资料均已编写齐全，并分类编目
2. alpha审查:是由用户在开发者的场所来进行的，Alpha测试是在一个受控的环境中进行
3. beta测试:由软件的最终用户在一个或多个用户场所来进行的,开发者一般不在现场，用户记录问题报告给开发者，开发者修改后发布最终软件

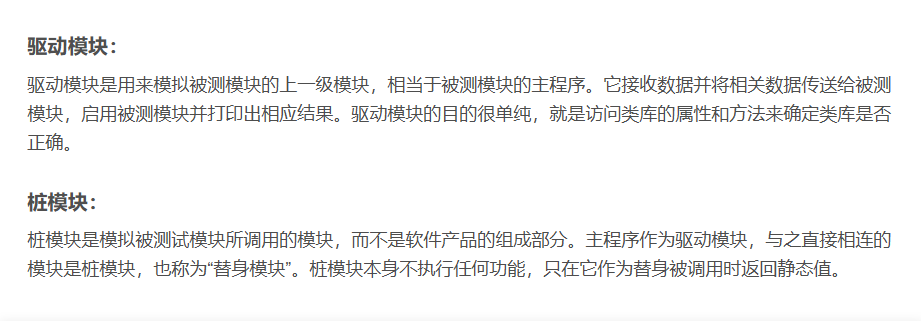
⑤回归测试：

1. 添加新功能后全重新测试，可以人工也可以自动化捕捉回放工具来测
2. 测试方法：全部重测或基于风险选择回归测试包

* 驱动模块与桩模块

什么是桩模块？什么是驱动模块？简单描述二者在集成测试中的作用

答：桩模块是替代被测模块的下层模块功能，为被测模块实现下层接口的功能；驱动模块是替代被测模块的上层模块功能，实现被测模块的上层调用功能。在自上而下的集成测试中需要桩模块实现逐步集成，自下而上的集成测试中需要驱动模块实现逐步集成。



第二章 需求、设计代码评审

静态测试，给一个例子，分析有什么问题，从什么角度去改（无论是需求，无论是设计还是代码，什么样的规则是合适的，有什么问题熟悉评测的要求）

静态测试 -代码运行前 -- 需求文档--设计文档--静态代码

需求，设计代码评审主要内容

1. 评审的方法和技术
2. 需求规格说明书的检查
3. 系统结构与业务逻辑检查
4. 代码风格和规则检查
5. 软件评审
6. 含义：软件评审是对软件元素或者项目状态的一种评估手段，以确定其是否与计划的结果保持一致，并使其得到改进。
7. 方法：走读，小组评审，审查 （正式和非正式两种方式）
8. 技术：检查表，场景分析，头脑风暴，辅助工具（统计，分析工具）
9. 评审会议角色以及职责

①作者：解释疑问、解决识别的缺陷

②主持人：从头到尾管理评审过程

③评审员：识别缺陷和问题

④记录员：记录识别的缺陷和问题并分发结果

⑤PM：确定评审计划、确认评审对象资格

⑥QA：评审过程支持、审计评审过程

⑦CM：工作产品配置管理

1. 产品需求评审
2. 需求评审重要性

v 发现需求定义中的问题，尽早发现缺陷，降低劣质成本。

v 保证软件需求的可测试性。

v 与市场、产品、开发等相关人员在需求理解上认识一致，以免后期的争吵。

v 更好的理解产品的功能性与非功能性需求，为制定测试计划打下基础。

v 确定测试目标与范围。虽然此后需求会发生变更，但能得到有效控制，降低测试风险。

1. 需求评审禁忌词

§ 某些、有时、常常、通常、贯常、经常、许多、大多、几乎、用户友好的、容易的、简单的、复杂的、健壮的、无缝的、透明的、优雅的、 最新技术等。这些词语太过模糊，所描述的功能无法验证

§ 等等、诸如此类、依此类推、包括但不限于等。不完整，无法验证

§ 良好、迅速、廉价、高效、灵活、稳定、显著、醒目等。这些是不确定的说法，不可验证

§ 可接受的、足够的、差不多的、可选择的、合理的、充分的、必要的、相关的等

§ 一般情况下、理想情况下、必要时、合适时

1. 需求常见问题汇总

目标性需求没有沟通好，后面的需求变成空中楼阁。

缺乏评审的可操作依据，遗漏评审内容。

没有作好前期准备工作，导致评审时间长，效率低。

没有选择合适的评审人员，无法获得有价值的反馈。

参加人员过多，容易陷入细枝末节的讨论，会议演变成一场人人自由的混战。

1. 设计评审

系统架构设计的评审：保证系统具有高性能、高可靠性、高安全性、高扩展性和可管理性

组件设计的审查：功能接口定义，算法的有效性，合理的数据结构、数据流和控制流

界面设计的评审

系统部署审查

1. 代码审查

代码风格与规则检查

程序设计和结构的检查

业务逻辑的检查：主要是检查单元功能设计是否满足概要设计要求、模块功能,模块层次关系，模块调用关系

第三章 测试用例设计

测试用例 的组成、结构、组织（基本的一些思想）

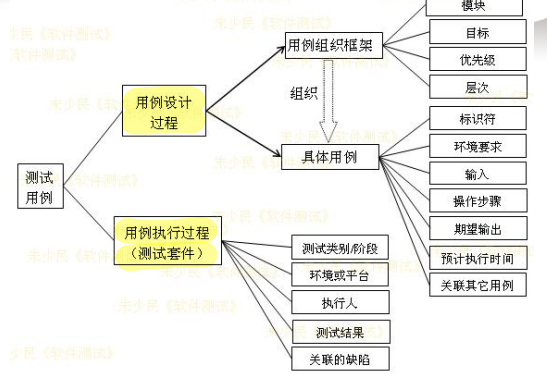
测试用例：独立执行的一个个体，最小测试实体，不能再分解。为了某个测试点而设计的测试操作过程序列、条件、期望结果及其相关数据的一个特定的集合。

测试用例的质量要求：要覆盖率好，易用，容易维护，力度适中。

测试的组织：

测试用例套件是由一系列测试用例并与之关联的测试环境组合而构成的集合，已满足测试执行的特定要求。通过测试套件，将服务于同一个测试目标、特定阶段性测试目标或某一运行环境下的一系列测试用例有机地组合起来。 按程序功能模块组织，按测试用例的类型组织 ，按测试用例的优先级组织

有效的组织测试用例：



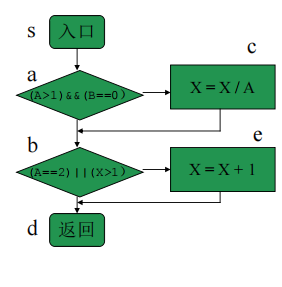
第四章 白盒测试

测试方法：白盒测试关于面向结构覆盖的（语句，判定，条件，组合条件，路径 MCDC 的思想，能发现什么样的缺陷，又不能发现什么，他们之间什么样的关系 考试也是给一个程序 你通过程序生成不同的测试用例来满足不同的覆盖）

注意：只有路径覆盖是用的控制流图进行分析，剩下的都是用的流程图来分析

大题、 判定、选择（给一个深度的思考，是解决什么问题的？）

概念：白盒测试把测试对象看做一个透明的盒子，它允许测试人员利用程序内部的逻辑结构及有关信息设计或选择测试用例，对程序所有逻辑路径进行测试。通过在不同点检查程序的状态，确定实际的状态是否与预期的状态一致。因此白盒测试又称为结构测试或逻辑驱动测试。 “白盒”法是穷举路径测试

面向单语句的测试准则 面向语句间控制的测试准则

**4.1语句级覆盖**

1.语句覆盖

每条语句执行一次、最弱覆盖，没考虑各种条件和分支

为使每个语句都执行一次，程序的执行路径应是sacbed

2.判定覆盖 （分支覆盖）

选取足够多的测试数据，使被测试程序中不仅每个语句

至少执行一次，而且每个判定都要走“true”和“false”

满足边覆盖，不一定满足路径覆盖

3.条件覆盖

每个判定里的每个条件都要取一遍，只考虑条件。

条件覆盖”通常比“判定覆盖”强，因为它使一个判定中的每一个条件都取到了两个不同的结果，而判定覆盖则不保证这一点。

条件覆盖并不包含判定覆盖

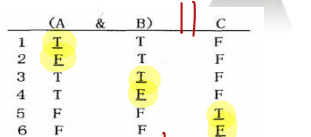
4.判定/条件覆盖

使得判定表达式中的每个条件都取到各种可能的结果，而且每个判定表达式也都取到各种可能的结果

某些条件掩盖了另一些条件，逻辑短路。

5.MC/DC覆盖

每个条件都要取一遍，而且是程序都执行到这里，考虑了逻辑短路的情况。

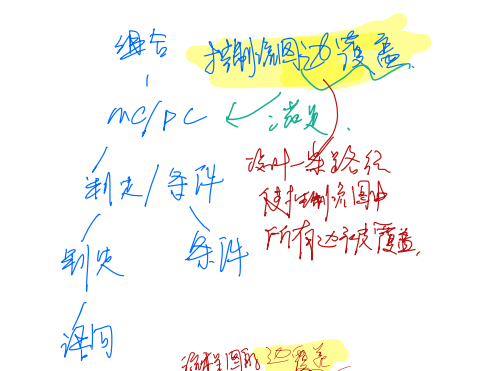
每个入口点和出口点至少被调用一次，判定条件每个条件的所有取值至少出现一次，每个判定的所有结果至少出现一次，每个条件都能独立影响判定结果。

6.条件组合覆盖

把条件的各种组合都走一遍

TT TF FT FF

关系图



**4.2 路径覆盖**

1. 路径覆盖:选取足够多的测试数据，使得程序的每条可能路径都至少执行一次

Sabd，sabed，sacbd，sacbed

1. 基本路径测试

在控制流图的基础上，通过分析控制结构的环路复杂性，导出基本可执行路径的集合，测试中语句覆盖100，条件覆盖100

一个图的基本路径集合数目是确定，图中的任意路径都可以表示为基本路径的线性组合

点是一个路径为0的路径

步骤

**1.画控制流图：**判定语句分解，顺序语句合并。

**2.计算圈复杂度**（基本路径数）

= 边-点+2

= 区域数 （边和节点封闭起来的区域+最外面的1）

= 流图的判定结点数+1

**3.写出基本路径**

一条独立路径是指，和其他的独立路径相比，至少引入一个新处理语句或一个新判断的程序通路

**4.设计测试用例**

**4.3 程序切片测试**

定义：给定一个程序P和P中的一个变量集合V，变量集合V在语句n上的一个片，记作S(V,n)，是P中对V中的变量值作出贡献的所有语句集合

**控制依赖:**假设C是包含结点n1和n2的控制流图，n1是一个判断结点，如果下面两个条件同时成立，则称结点n2控制依赖于n1:

1. 至少存在一条从n1到程序出口的路径，该路径包含n2
2. 至少存在一条从n1到程序出口的路径，该路径不包含n2.

**数据依赖:**假设D是包含结点n1和n2的数据依赖图，如果下面两个条件同时成立，

则称结点n2数据依赖于n1:

1.变量v在在n1处定义、n2处引用

2.存在一条从n1到n2的非空路径，不包含任何重定义v的结点

**4.4变异测试**

变异算子： 在符合语法规则前提下， 变异算子定义了从原有程序生成差别极小程序（即变异体）的转换规则。例如：将“＋” 操作符变异为 “-”操作符。

变异后不一定错误。

可杀除变异体:若存在测试用例t，在变异体p'和原有程序p上的执行结果不一致，则称该变异体 p'相对于测试用例集T是可杀除变异体。

可存活变异体:若不存在任何测试用例t在变异体 p'和原有程序p上的执行结果不一致，则称该变异体p'相对于测试用例集T是可存活变异体。

等价变异体:若变异体p'与原有程序p在语法上存在差异， 但在语义上与 p保持一致， 则称p'是p的等价变异体。

第五章 黑盒测试 功能测试 数据驱动测试

黑盒测试：给一个问题，设计测试用例

常见的一些方法，等价类，边界值（有时候会混合在一起考） 等价类，边界值也是一种组合思想

软件产品的功能就是为了满足用户的实际需求而设计的，所有的功能都需要得到验证，确认真正地满足子用户的需求--功能测试

5.1功能测试概述

定义：依据产品设计规格说明书完成对产品功能进行操作，以验证系统是否满足用户的功能性需求

功能测试又称为黑盒测试又叫数据驱动测试，它是把测试对象看做一个黑盒子，测试人员完全不考虑程序内部的逻辑结构和内部特性，只依据程序的需求规格说明书，检查程序的功能是否符合它的功能说明。

1. 等价类划分法
2. 划分有效无效等价类 并给等价类编号
3. 设计测试用例尽可能多的覆盖有效等价类。
4. 每次只能覆盖一个无效等价类。

3.要考虑取值的具体情况。

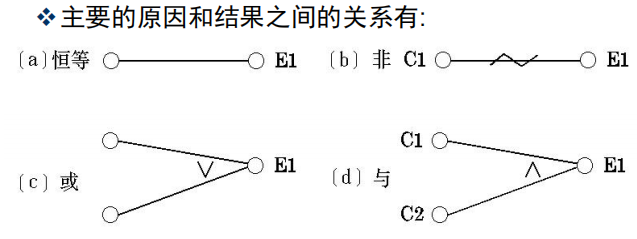
二、边界值分析法

边界值的设定：最小、次小、正常、次大、最大

n个变量的设定，让每一个变量分别取最小、次小、次大、最大，其余变量取正常值，这样共有4n种取值情况，最后所有变量取正常值，共有4n+1种取法。

三、因果图法

通过因果图，可以建立输入条件和输出之间的逻辑模型，从而比较容易确定输入条件组合和输出之间的逻辑关系，有利于设计全面的测试用例



**输入条件的四种约束关系：**

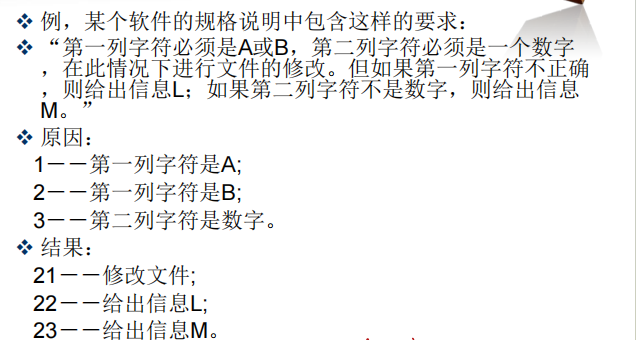
E异（至多1个为1） I或（至少1个为1） O唯一（有且仅有1个1） R要求（a是1时b也必须是1）

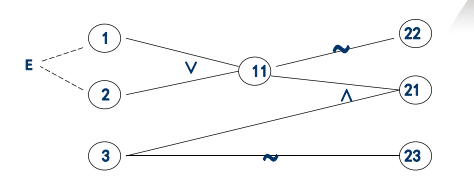
**输出条件约束：**M强制（a是1时b必须是0）

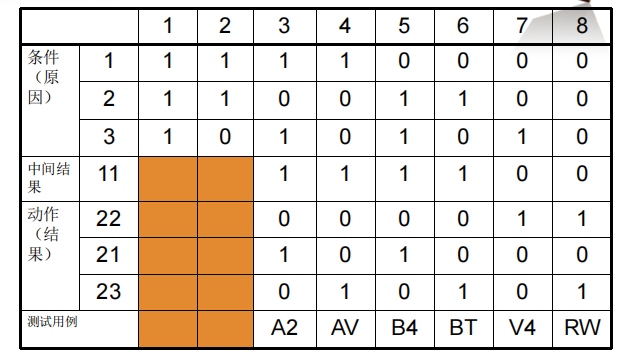
**因果图设计步骤：**设计用例图先写：原因，C1,C2,C3…这种，结果E1,E2,E3…

然后根据因果图设计判定表

把判定表的每一列拿出来，设计测试用例







四、决策表方法

先列条件的判断，取值为Y或者N，再列结果的判断。

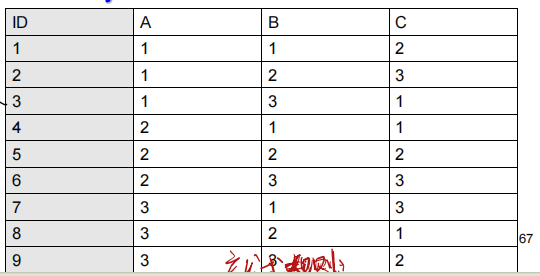
简化、合并相似规则或者相同动作。

五、功能图法

每个程序的功能通常由静态说明和动态说明组成，静态说明描述了输入条件和输出条件之间的对应关系，而动态说明描述了输入数据的次序或者转移的次序。

功能图法就是为了解决动态说明问题的一种测试用例的设计方法

功能图由状态迁移图和逻辑功能模型构成



六、组合测试设计方法

 pair-wise

组合覆盖准则：

Each Choice ：每个值用一次，不考虑输入变量之间的关系

Pair-Wise：任意两点满足组合，考虑任意两个变量之间的联系。

t-Wise : t个变量组合在一起

**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(条件组合)，从而合理地安排实验(测试)的一种科学实验设计方法

适用于**输入条件相互独立**，是研究多因素多水平的一种设计方法。

具备“**均匀分散，整齐可比**”的特点。

**正交表：**

行数：正交表行的个数，设计用例的个数。

因素数：正交表列的个数，要测试的功能点数。

水平数：任何单个因素都能取到值的最大个数。

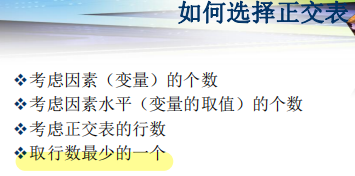
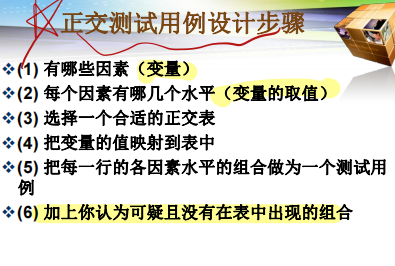


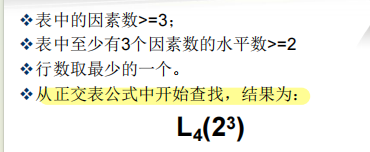
**整齐可比性**

在同一张正交表中，**每个因素的每个水平出现的次数是完全相同的**。由于在试验中每个因素的每个水平与其它因素的每个水平参与试验的机率是完全相同的，这就保证在各个水平中**最大程度的排除了其它因素水平的干扰。**

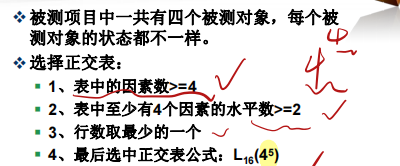
**均衡分散性**

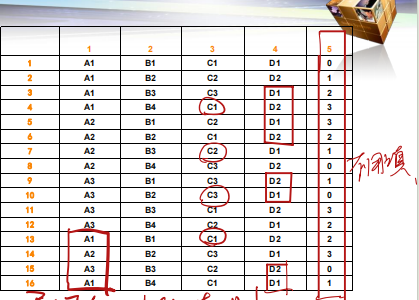
在同一张正交表中，任意两列（两个因素）的水平搭配（横向形成的数字对)是完全相同的。这样就保证了**试验条件均衡地分散在因素水平的完全组合之中**，因而具有很强的代表性，容易得到好的试验条件。 满足pair-wise



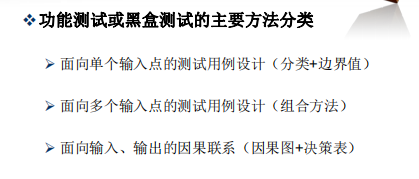


试验次数（行数）＝∑（每列水平数－1）＋1



多余的迭代填进去





第六章 web测试

Web测试，没有大题 选择，判断，简单描述

**6.1 Web网站的特点：**网络集约性，内容驱动性，持续演化性，即时性，安全性，美观性

1. Web测试分类
2. 系统架构：

①服务端测试

②客户端测试

1. 质量特性：

①功能测试

②性能测试

③易用性测试

④兼容性测试

⑤安全性测试



1. 系统职能：

①数据库服务测试

②安全系统测试

③应用功能测试

④应用服务测试

**6.2 Web测试用例设计**

1. Web功能性测试用例设计
2. Web性能测试用例设计
3. Web易用性测试用例设计
4. Web兼容性测试用例设计
5. Web安全性测试用例设计

**一. Web功能性测试**

1. 链接测试
2. 链接：Web应用系统的一个主要特征，它是在页面之间切换和指导用户去一些不知道地址的页面的主要手段。
3. 用例设计：

首先，测试所有链接是否按指示的那样确实链接到了该链接的页面；其次，测试所链接的页面是否存在; 最后，保证Web应用系统上没有孤立的页面，所谓孤立页面是指没有链接指向该页面，只有知道正确的URL地址才能访问。

§ 链接的显示

§ 链接跳转的结果

§ 链接访问的页面是否存在

§ 是否有孤立的页面存在

1. 表单与数据校验
2. 表单：在网页中经常使用，主要负责数据采集的功能（表单标签+表单域+表单按钮）
3. 用例设计：用边界值分析、等价类划分等方法

§ 显示：加载、缺省值、快捷方式、提示信息等

§ 单个控件：文本框、下拉列表、文件上传、日期框、检查框 、滚动条、网格、ActiveX控件等

§ 组合控件：几个下拉框、日期范围等

1. 状态保存测试

**Session测试**

1. 含义：指一类用来在客户端与服务器端之间保持状态的解决方案。Session (会话)
2. 用例设计思想：

§ 登录后的权限

§ 注销后的再次登录

§ Session超时

§ 一终端多用户和多终端一用户等

**Cookies测试**

1. 含义：一种能够让网站服务器把少量数据储存到客户端的硬盘或内存，或是从客户端的硬盘读取数据的一种技术。用于自动登录。
2. 用例设计

§ Cookies的加密

§ 自动登录

§ 失效时间

§ 更改密码等

**Cache测试**

1. 含义：用来保存浏览过页面的一种机制。作用就是缓存浏览过的页面,图片等。
2. 用例设计：

§ 缓存是否起作用

§ 数据修改后是否能及时刷新

§ 缓存失效时间是否正确

1. 数据库测试

（1）一般情况下，可能发生两种错误，分别是数据一致性错误和输出错误。

§ 数据一致性错误主要是由于用户提交的表单信息不正确而造成的，

§ 输出错误主要是由于网络速度或程序设计问题等引起的，针对这两种情况，可分别进行测试

（2）用例设计（最好使用真实数据测试）

v 增加、修改、查询数据，注意会造成字段约束、默认值、重复数据等问题

v 注意删除数据的关联情况

v 数据的并发访问冲突

v 数据库的压力测试

v 数据库的备份恢复问题

**二、web性能测试**

1. 连接速度测试

用户连接到Web应用系统的速度根据上网方式的变化而变化

1. 负载测试

（1）测量Web系统在某一负载级别上的性能，以保证Web系统在需求范围内能正常工作

（2）负载级别可以是某个时刻同时访问Web系统的用户数量，也可以是在线数据处理的数量。

1. 压力测试
2. 安排在Web系统发布以后，在实际的网络环境中进行测试.
3. 指实际破坏一个Web应用系统，测试系统的反映。压力测试是测试系统的限制和故障恢复能力，也就是测试Web应用系统会不会崩溃，在什么情况下会崩溃。黑客常常提供错误的数据负载，直到Web应用系统崩溃，接着当系统重新启动时获得存取权
4. 测试的区域包括表单、登陆和其他信息传输页面等
5. **Web易用性测试**

“易用性”是一个衡量标准，用来衡量使用一个产品完成指定任务的难易程度。

并规定用户应该是任何设计决定中最重要的因素。

1. **Web兼容性测试**

操作系统，浏览器，网络环境，分辨率，打印机

组合测试，v600x800 的分辨率在MAC 机上可能不错，但是在IBM 兼容机上却很难看....

1. **Web安全性测试**

就攻击技术本质而言，它利用的工具是SQL的语法，SQL Injection攻击是存在于常见的多连接的应用程序中的一种漏洞，攻击者通过在应用程序预先定义好的sql语句结尾加上额外的SQL语句元素，欺骗数据库服务器执行非授权的任意查询，篡改和命令执行。

就风险而言，SqL攻击也是位居前列，和缓冲区溢出漏洞相比，其优势在于能够轻易的绕过防火墙直接访问数据库，甚至能够获得数据库所在的服务器的系统权限。在Web应用漏洞中，SQLInjection 漏洞的风险要高过其他所有的漏洞。

常见的SQL注入

1. Get参数触发SQL注入
2. POST参数触发SQL注入
3. Cookie触发SQL注入
4. 其他参与sql执行的输入都有可能进行SQL注入

**因果图法 题目**

