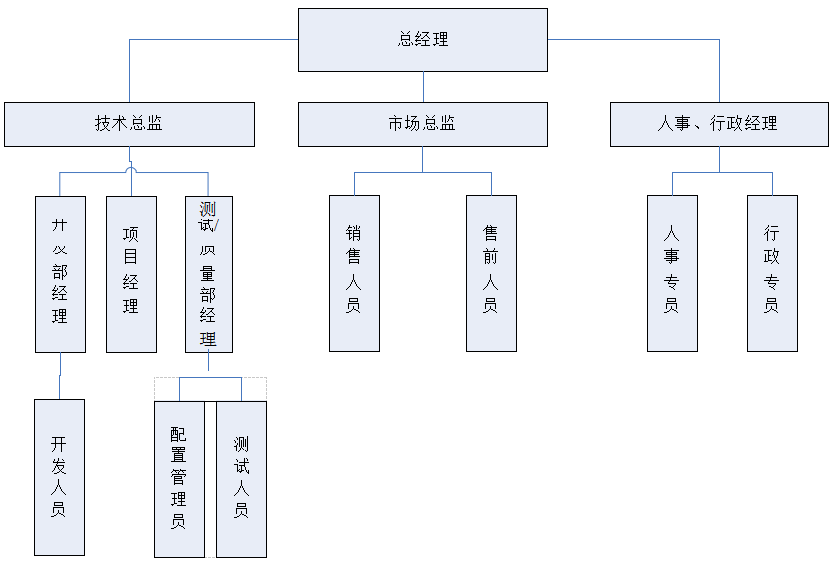
# **第一章 软件测试概述**

## **学习目标**

了解软件企业岗位

了解软件测试行业

## **软件公司的职位介绍**



## **软件的概念**

计算机软件，是指计算机程序以及解释和指导使用程序的文档的总和。

计算机程序包括源程序和目标程序。同一程序的源文本和目标文本应当视为同一作品。

源程序是指用高级语言或汇编语言编写的程序

目标程序是指源程序经编译或解释加工以后可以由计算机直接执行的程序。

## **软件的特点**

1. 计算机软件既是作品，又是工具，是作品性与工具性紧密结合的智力成果。
2. 计算机软件开发工作量最大、成本高，但复制容易、成本极低。
3. 计算机软件具有无形性，可以多次使用， 但商业寿命较短

## **软件危机**

软件危机(Software Crisis) 是计算机软件在它的开发和维护过程中所遇到的一系列严重问题。

主要包含两方面的问题：如何开发软件，怎样满足对软件日益增长的需求；如何维护数量不断膨胀的已有软件

在大型软件的开发过程中出现了复杂程度高、研制周期长、正确性难以保证的三大难题。遇 到的问题找不到解决办法，致使问题堆积起来， 形成了人们难以控制的局面，出现了所谓的 “软件危机”

### **软件危机的表现**

1. 对软件开发成本和进度的估计很不准确
2. 用户对“已完成的”软件系统不满意的现象经常发生
3. 软件产品的质量常常靠不住
4. 软件常常是不可维护的
5. 软件通常没有适当的文档资料
6. 软件成本在计算机系统总成本中所占比例逐年上升

### **软件危机出现的原因**

1. 软件生产本身存在着复杂性
2. 软件开发所使用的方法和技术有关。
3. 软件工程正是为克服软件危机而提出的一种概念，并在实践中不断地探索它的原理，技术和方法。

## **软件测试**

**软件测试的产生**：软件规模越来越大; 软件开发与用户之间的矛盾

**软件测试的定义**：为了发现错误而执行程序或者系统的过程；1983年，IEEE软件工程标准术语：使用人工或自动手段，来运行或测试某个系统的过程。其目的在于检验它是否满足规定的需求或弄清预期结果与实际结果之间的差别。

### **软件缺陷**

**软件缺陷定义**，满足5个条件之一（所有软件问题都称为缺陷）

1. 软件未达到产品说明书中已标明的功能
2. 软件出现了产品说明书中指明不会出现的错误
3. 软件功能超出了产品说明书指明的范围
4. 软件未达到产品说明书虽未指出但应达到的目标
5. 软件测试员认为软件难以理解，不易使用，运行速度缓慢，或者最终用户认为该软件使用效果不好。

### **怎样有效记录缺陷**

1. 分析故障——使用最少步骤重现缺陷
2. 减少开发人员重现缺陷的时间
3. 使开发人员更准确的定位缺陷
4. 包含所有重现缺陷的必要步骤
5. 测试人员假定常用的操作步骤开发人员不一定熟悉，省略了必要的步骤常常造成开发人员无法重现缺陷。
6. 注意自己的语气
7. 保证重现缺陷
8. 分析故障——使用最少步骤复现故障
9. 包含所有重现缺陷的必要步骤
10. 方便阅读
11. 尽量简单——一个缺陷一个报告
12. 值得注意的经验

### **软件测试的现状与前景**

国内：处于起步（发展）阶段（手工测试）， 有较好职业发展空间

国内外软件企业越来越重视软件测试

工作量、费用通常比例（发达国家）

– 设计：开发：测试—— 4：2：4

自动测试工具的出现提高了测试的效率

**软件测试行业发展的原因**

1. 市场竞争的压力
2. 不断提升的用户需求
3. 整个行业逐渐的规范
4. 用户技术水平的提升

### **软件测试人员的能力要求**

兴趣；灵感；经验；态度；沟通能力；开发能力；测试与调试的区别

**测试与调试的区别**

完成的任务不同：测试的任务是发现程序中的缺陷；调试的任务是定 位并且解决程序中的问题。

执行的人不同：测试主要是由测试人员和开发人员来执行，黑盒测 试主要由测试人员完成、单元/集成测试主要是由开发人员执行。

执行的周期不同：测试贯穿整个软件开发生命周期，调试一般在开发 阶段。

**开发VS测试**

难易程度；工作环境；薪水；发展前景；

# **测试分类**

## **按照测试方法分类**

**静态测试：**静态测试是不运行被测试本身而寻找程序代码中可能存在的错误或评估程序代码的过程。静态测试通 过分析或者监察源程序的语法、结构、过程、接口 等来检查程序的正确性，找出问题

**动态测试：**动态是运行被测试程序，输入相应的测试数据，检 查运行结果与预期结果的差异，判定执行结果是否 符合要求，从而检验程序的正确性、可靠性和有效 性，并分析系统运行效率和健壮性等性能。

**黑盒测试：**又称为功能测试、于规格说明书的测试。注重于测试软

**灰盒测试：**是介于黑盒测试的测试方法。在执行白盒测试的时的方法

**白盒测试**又称结构测试、逻辑驱动测试或基于程序本身的测试

**冒烟测试：**冒烟测试（Smoke testing）的对象是每一个新编译的需要正式测试的软件版本，目的是确认软件基本 功能正常，可以进行后续的正式测试工作。

**回归测试：**在发生修改之后重新测试先前的测试以保证修改的 正确性。理论上，对软件的任何新版本，都需要进 行回归测试，验证以前发现和修复的错误是否在新 软件版本上再现。

**功能测试：**根据产品特征、操作描述和用户方案，测试一个产品的特性和可操作行为以确定它们满足设计需求

**性能测试：**评价一个产品或组件与性能需求是否符合的测试。包括负载测试、强度测试、数据库容量测试、基准测试等类型

**压力测试：**获取系统正确运行的极限，检查系统在瞬间峰值负荷下正确执行的能力。

**负载测试：**用于检查系统在使用大量数据的时候正确工作的能力，即检验系统的能力最高能达到什么程度。

**易用性测试：**主要从使用的合理性和方便性等角度对软件系统进行检查，发现人为因素或使用上的问题。在保证足够详细的程度下，用户界面要便于使用，对输入的响应时间和响应方式合理，输出有意义、正确，出错信息能够引导用户去解决问题，文档全面、确切等。易用性测试多数情况下没有一个量化的指标，主观性较强。

**安装测试：**安装测试（Installing testing），确保该软件在正常情况和异常情况的不同条件下能够正确安装

**界面测试：**指测试用户界面的风格是否满足客户要求，文字是否正确，页面是否美观，文字，图片组合是否完美，操作是否友好等等

**配置测试：**配置测试(Configuration Testing)主要检查计算机系统内各个设备或各种资源之间的相互连接和功能分配中的错误

**文档测试：**主要检查文档的正确性、完备性和可理解性

**兼容性测试：**测试软件是否和系统的其它与之交互的元素之间兼容

**安全性测试：**检查系统对非法浸入的防范能力，检查系统中已经 存在的系统安全性、保密性措施是否发挥作用，有 无漏洞

**恢复测试：**主要检查系统的容错能力

**可移植性测试：**测试软件是否可以被成功移植到指定的硬件或软件 平台上。

**引导测试：**软件开发中，验证系统在真实硬件和客户基础上处 理典型操作的能力

**随机测试：**没有书面测试用例、记录期望结果、检查列表、脚本或指令的测试。主要是根据测试者的经验 对软件进行功能和性能抽查

**健全测试：**软件主要功能成分的简单测试以保证它是否能进行基本的测试

**手工测试：**是采用手工的方式执行测试；

**自动化测试：**使用自动化测试工具来进行测试，这类测试一般不需要人干预，通常在GUI、性能等测试中用得较多。

**国际化测试**：国际化测试的目的是测试软件的国际化支持能力，发现软件的 国际化的潜在问题，保证软件在世界不同区域中都 能正常运行。

**本地化能力测试** ：本地化能力是指不需要重新设计或修改代码，将程序的用户 界面翻译成任何目标语言的能力。为了降低本地化 能力测试的成本，提高测试效率，本地化能力侧是 通常在软件的伪本地化版本上进行

**本地化测试**：本地化测试的对象是软件的本地化版本。本地化测试的环境是在本地化的操作系统上安装本地化的软件。从测试方法上可以分为基本功能测试，安装/卸载测试，当地 区域的软硬件兼容性测试。测试的内容主要包括软件本地化后的界面布局和软件翻译的语言质量，包含软件、文档和联机帮助等部分

**通过测试：**是使用满足需求的数据测试系统的过程。

**失败测试：**是使用不满足需求的数据测试系统的过程。

**错误猜测：**错误猜测试是通过直觉发现程序中的错误和缺陷的能力。

## **按照测试阶段分类**

**单元测试：**单元测试（Unit testing）：最小模块的测试， 可以是一个函数或子程序，一般由开发者在系统开发过程中进行执行。单元测试针对每一个程序模块进行正确性检验，检查各个程序模块是否正确地实现了规定的功能。单元测试是测试的第一步，其依据是详细设计，单元测试应对模块内所有重要的控制路径设计测试用例， 以便发现模块内部的错误。

**集成测试：**集成测试（Integration testing），被测试系统的所有组件都集成在一起，找出被测试系统组件之间关系和接口中的错误。该测试一般在单元测试之后进行

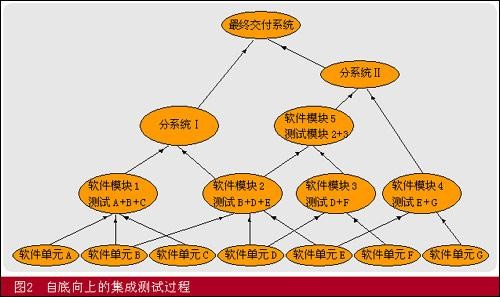
**集成测试：**

-非增式集成

-增式集成

自顶向下增式测试

自底向上增式测试（最常使用的方法）



**确认测试：**由集成测试进入系统测试之前，需要对软件是否可以进入系统测试进行评价，这个过程成为确认测试

确认测试需要做的工作包括：有效性测试、软件配置审查。

**系统测试：**是将通过确认测试的软件，作为整个基于计算机系统的一个元素，与计算机硬件、外设、某些支持软件、数据和人员等其他系统元素结合在一起，在实际运行环境下，对计算机系统进行全面的功能覆盖。

**验收测试：**

Alpha testing (α测试),是由一个用户在开发环境下进行的测试，也可以是公司内部的用户在模拟实际操作环境下进行的受控测试，Alpha测试不能由程序员或测试员完成。

Beta testing(β测试),测试是软件的多个用户在一个或多个用户的实际使用环境下进行的测试。开发者通常不在测试现场，Beta测试不能由程 序员或测试员完成。

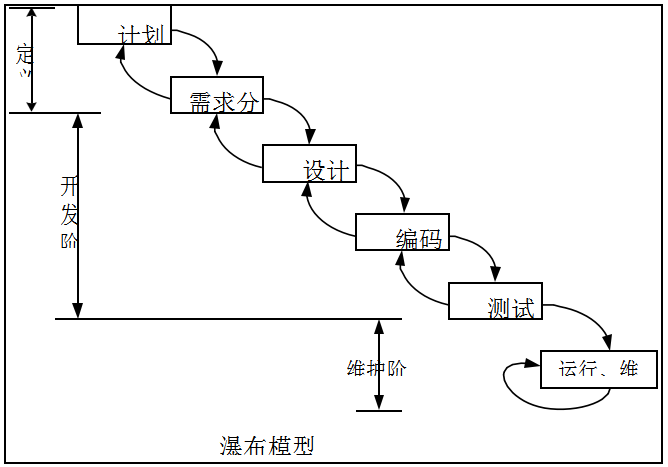
# **第三章 软件测试流程**

## **软件开发的生命周期**

软件生命周期是指从软件产品的设想开始到软件不再使用而结束的时间。

测试计划→ 测试设计、测试开发→ 测试执行→ 测试评估

## **瀑布模型 （Waterfall Model）**



### **瀑布模型的特点**

从上一项活动接受本项活动的工作对象， 作为输入；

利用这一输入实施本项活动应完成的内容；

给出本项活动的工作成果，作为输出传给下一项活动。

### **瀑布模型适用场合**

在需求不明确的情况下，就无法采用瀑布模型

**瀑布模型的优缺点**

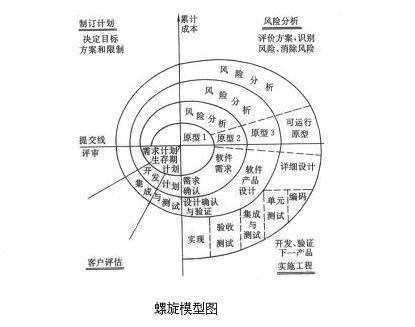
优点：

1. 强调开发的阶段性；
2. 强调早期计划及需求调查；
3. 强调产品测试。

缺点：

1. 依赖于早期进行的唯一一次需求调查，不能适应需 求的变化；
2. 由于是单一流程，开发中的经验教训不能反馈应用 于本产品的过程；
3. 风险往往迟至后期的开发阶段才显露，因而失去及 早纠正的机会。

## **螺旋模型 （Spiral Model）**



**螺旋模型的优缺点**

优点：

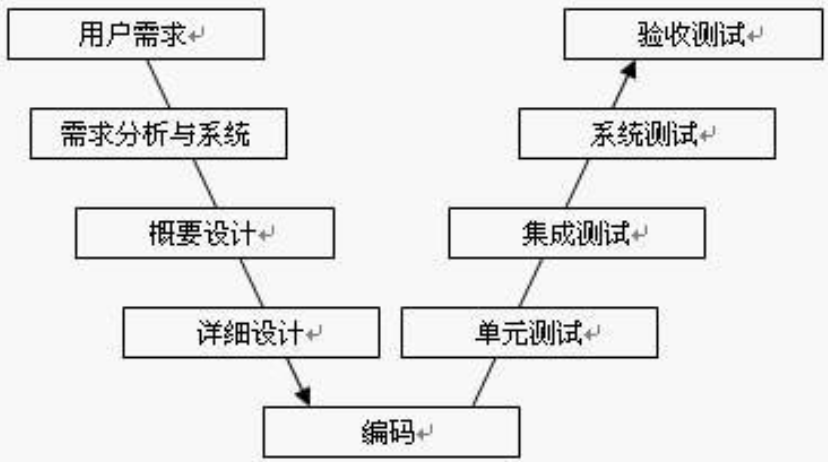
1. 强调严格的全过程风险管理。
2. 强调各开发阶段的质量。
3. 提供机会检讨项目是否有价值继续下去。

缺点：

引入非常严格的风险识别、风险分析和风险控制，这对风险管理的技能水平提出了很高的要求。这 需要人员、资金和时间的投入。

## **软件测试v模型**

**模型**



**V模型介绍**

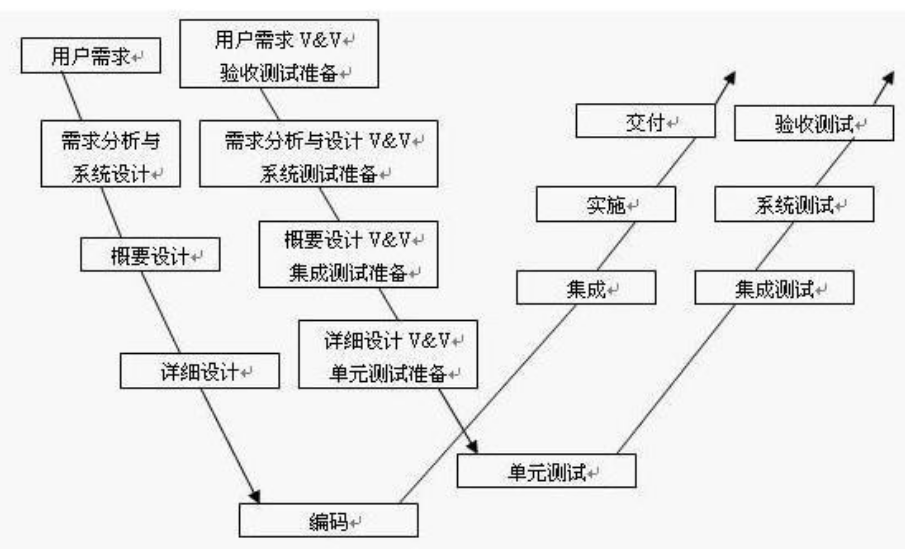
V模型最早是由Paul Rook在２０世纪８０年代后期提出的， 目的是改进软件开发的效率和效果。是瀑布模型的变种

明确的标注了测试过程中存在的不同类型的测试，并且清楚的描述了这些测试阶段和开发过程期间各阶段的对应关系

V模型指出，单元和集成测试应检测程序的执行是否满足软件设计的要求；系统测试应检测系统功能、性能的质量特性是否达到系统要求的指标；验收测试确定软件的实现是否满足用户需要或合同的要求

局限性:仅仅把测试作为在编码之后的一个阶段，未在需求阶段就进入测试

## **软件测试双V模型**



W模型增加了软件各开发阶段中应同步进行的验证和确认活动。W模型由两个V字型模型组成，分别代表测试与开发过程，图中明确表示出了测试与开发的并行关系。

W模型特点：测试的对象不仅是程序，需求、设计等同样要测试，测试与开发是同步进行的

W模型优点：有利于尽早地全面的发现问题。例如，需求 分析完成后，测试人员就应该参与到对需求的验证和确认 活动中，以尽早地找出缺陷所在。同时，对需求的测试也 有利于及时了解项目难度和测试风险，及早制定应对措施， 显著减少总体测试时间，加快项目进度。

局限性：需求、设计、编码等活动被视为串行的；测试和开发活动也保持着一种线性的前后关系，上一阶段完全结束，才可正式开始下一个阶段工作。无法支持迭代的开发模型。对于当前软件开发复杂多变的情况，W模型并不能解除测试管理面临着困惑。

**软件测试针对软件开发生命周期的流程**

**需求阶段 -** 测试人员了解需求、对需求进行分解，得出测试需求，编写测试计划/测试方案

**设计阶段 -** 测试人员适当的了解设计，对于设计测试用例是很有帮助的，测

试人员可以根据需求和设计编写一部分测试用例

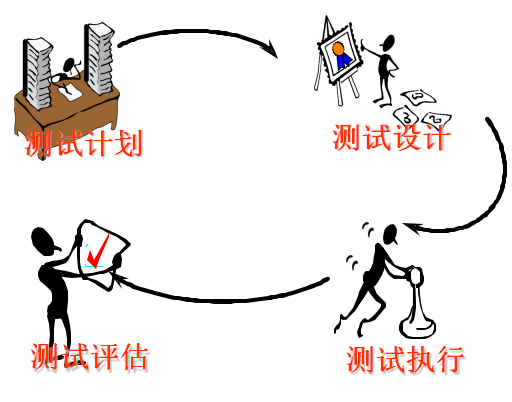
**编码阶段 -** 测试人员一般是不需要编码的，但已经编码的模块，专业的白盒

测试人员可以计划执行单元测试

**测试阶段 -** 测试阶段是软件测试人员最为重要的工作阶段，这时候测试人员需要把测试用例进一步细化，根据测试用例和计划执行测试，在执行的过程中记录缺陷，测试完成后整理测试报告。

**实施 -** 测试人员需要参与项目的实施工作。测试人员对项目产品的业务和操作非常了解，加上测试人员的沟通表达能力一般都比较强， 所以测试人员可以参与用户使用软件的培训，在试运行项目时收集问题并及时反馈给相关负责人。

**软件测试的活动**



# **第四章 软件测试原则**

## **软件测试原则**

1. 应当把“尽早和不断的测试”作为开发者的座右铭
2. 程序员应该避免检查自己的程序，测试工作应该由独立的专业的软件测试机构来完成
3. 设计测试用例时应该考虑到合法的输入和不合法的输入以及各种边界条件，特殊情况下要制造极端状态和意外状态，比如网络异常中断、电源断电等情况
4. 一定要注意测试中的错误集中发生现象， 这和程序员的编程水平和习惯有很大的关系。测试中存在群集现象，错误喜欢发现在相同的模块以及相关的开发人员编写的程序
5. 对测试错误结果一定要有一个确认的过程， 一般有A测试出来的错误，一定要有一个B 来确认，严重的错误可以召开评审会进行 讨论和分析
6. 制定严格的测试计划，并把测试时间安排的尽量宽松，不要希望在极短的时间内完成一个高水平的测试
7. 回归测试的关联性一定要引起充分的注意， 修改一个错误而引起更多的错误出现的现 象并不少见
8. 妥善保存一切测试过程文档，意义是不言而喻的，测试的重现性往往要靠测试文档
9. 测试的Good Enough原则：不要做不充分的测试，也不要做过多的测试；找到测试费用和测试量之间的平衡点，是最佳选择。
10. 制定测试内容和最低测试通过标准加以衡量

## **软件测试经验**

1. 测试的规律木桶原理和80－20原则
2. 测试人员永远不要保证什么
3. 测试人员的编写的文档是代表自己
4. 测试人员要学会逆向思维
5. 编写缺陷一定要保证重现
6. 测试一定要根据需求《规格需求说明书》
7. 关注对用户不利的缺陷
8. 站在用户的角度
9. 适当的引入测试工具提高测试效率
10. 测试人员是服务人员

# **第五章 软件测试计划**

**软件测试计划的概念**

一个叙述了预定的测试活动的范围、途径、资源及进度安排的文档。它确认了测试项、被测特征、测试任务、人员安排，以及任 何偶发事件的风险。

**制定软件测试计划的好处**

项目经理、高层经理等相关领导能够根据测试计划做宏观调空，进行相应资源配置等；

测试人员能够了解整个项目测试情况以及项目测试不同阶段的所要进行的工作等；

便于开发人员、市场人员、质量人员等了解测试人员的工作内容，进行有关配合工作。

**谁来负责制定软件测试计划**

具有丰富经验的项目测试负责人制定测试计划，需要对整个测试过程负责

**编写软件测试计划的时间**

软件测试计划越早编写越好，以便对整个项目有总体的测试规划。

**软件测试计划的要素**

why——为什么要进行这些测试，测试目的是什么；

what—测试哪些方面，确定测试的内容；

when—测试不同阶段的起止时间，确定各测试活动的时间；

where—相应文档，缺陷的存放位置，测试环境等；

who—谁来负责相应的工作；

how—如何去做，使用哪些测试工具以及测试方法、测试策略进行测试。

**软件测试计划模版**

测试目的

测试项目简介

测试参考文档

测试提交文档

术语和定义

测试策略

确定测试内容

资源

测试进度

测试人员的任务分配

风险和问题

**软件测试计划维护与评审**

随着项目计划、软件需求、软件开发以及人员等的变更，软件测试计划也是不断更新和维护的

# **测试用例**

## **软件测试用例的概念**

测试用例（Test Case）是为了实施测试而向被测试的系统提供的一组集合，这组集 合包含：测试环境、操作步骤、测试数据、预期结果等要素。测试用例一般可以简单划分为：场景测试用例（简称“测试用例”）和基本测试用例（或称为“公用测试用例”）

## **设计测试用例的优缺点**

优点

-有效性

-完整性

-组织性

缺点

-测试用例的设计是费时费力的工作，往往设计测试用例所花费的时间比执行所花费的时间还多

-测试用例的模版

## **测试用例模版**

Word

Excel

使用工具

根据公司具体情况制定模版

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 项目名称 |  | | | 程序版本 | |  |
| 功能特性 |  | | | | | |
| 预置条件 |  | | | | | |
| 测试环境 |  | | | | | |
| 参考信息 |  | | 特殊规程 | |  | |
| 设计人 |  | | 设计时间 | |  | |
| 功能模块 | 用例编号 | 操作步骤/测试数据 | 预期结果 | | 实际结果/判定 | |
|  |  |  |  | |  | |
|  |  |  | |  | |
|  |  |  | |  | |
|  |  |  | |  | |
|  |  |  | |  | |
|  |  |  | |  | |
|  |  |  |  | |  | |
|  |  |  | |  | |

## **设计测试用例所需要的素质**

测试用例的方法

考虑问题的全面性

业务知识的深刻性

逆向思维能力

丰富的测试经验

## **测试用例的更新与维护**

1. 需要更新和维护的原因

-需求变更，功能变化，测试用例也需要更新

-测试用例需要细化和不断完善，是个循序渐进的过程

-通过测试实践检验测试用例并添加、修改、删除测试用例

1. 测试用例要经过正式、有效的评审
2. 利用测试工具来管理测试用例

# **第七章 等价类和边界值**

**等价类**

**等价类的基本概念**

依据需求将输入（特殊情况下会考虑输出） 划分为若干个等价类，从等价类中选出一 个测试用例，如果这个测试用例测试通过， 则认为所代表的等价类测试通过，这样就 可以用较少的测试用例达到尽量多的功能 覆盖，解决了不能穷举测试的问题。

**等价类类型**

有效等价类：对于程序的规格说明书是合理的、有意义的输入数据构成的集合，利用有效等价类验证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：根据需求说明书，不满足需求的集合。

**等价类的原则**

如果规定了输入值的范围(闭区间)，可以分为一个有效等价类，两个无效的等价类；

如果输入是布尔表达式，可以分为一个有效等价类和一个无效等价类；

如果规定了输入数据的一组值，而且程序对不同输入值做不同的处理，则每个允许的输入值是一个有效的等价类，此外还有一个无效的等价类（任意一个不允许的输入值）；

如果规定了输入数据必须遵循的规则，可以划分出一个有效的等价类（符合规则）和若干个无效的等价类（从不同角度违反规则）。

**等价类划分的步骤**

1. 考虑输入数据的数据类型(输入类型)
2. 考虑数据范围（输入长度）
3. 画出示意图，区分等价类
4. 为每一个等价类进行编号
5. 从一个等价类中选举一个测试数据构造测试用例

**等价类划分步骤示例**

医院系统挂号时输入姓名：

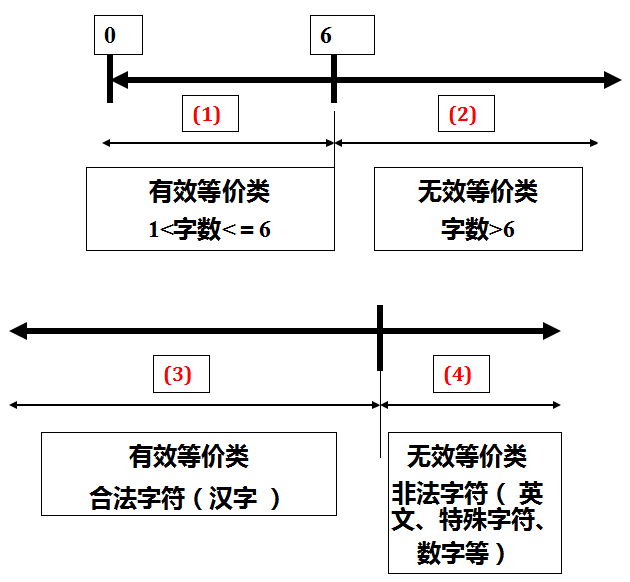
分析输入类型和输入长度

1、姓名的输入类型

2、姓名的输入长度

3、画出示意图

4、编号



等价类设计用例

合法、字数2-6：杨二车娜姆

非法、字数1-6：王@

合法、字数等于1：宝

合法、字数大于6：阿凡堤买买提阿吉米

ATM机的测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序**  **号** | **功能项** | **有效等价类** | **编号** | **无效等价类** | **编号** |
| **1** | **提取现金** | **50=<提取钱的数目**  **<=5000** | **2** | **提取钱的数目<50**  **提取钱的数目>5000** | **1**  **3** |
| **…..** | **…..** | **…..** | **…..** | **…..** | **…..** |

|  |  |  |  |
| --- | --- | --- | --- |
| **测试用例编号** | **输入数值** | **所属等价类** | **预期输出** |
| **1** | **20** | **1** | **错误信息** |
| **2** | **1000** | **2** | **正确输出** |
| **3** | **6000** | **3** | **错误信息** |

**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，而子集的并是整个集合；
3. 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）一个测试用例，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"。

**等价类的特点**

两块划成一块（等价类划分过粗），结果？

一块划成两块（等价类划分过细），结果？

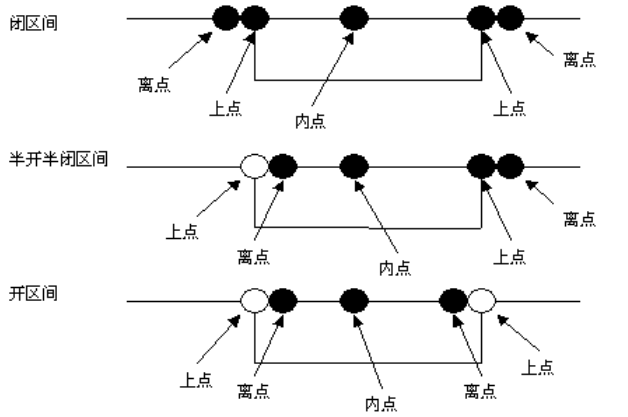
**边界值**

**边界值的基本概念**

上点：就是边界上的点，不管它是开区间 还是闭区间，就是说，如果该点是封闭的， 那上点就在域范围内，如果该点是开放的， 那上点就在域范围外；

内点：就是在域范围内的任意一个点；

离点：就是离上点最近的一个点，如果边界是封闭的，那离点就是域范围外离上点最近的点，如果边界是开放的，那离点就是域范围内离上点最近的点。



**边界条件类型**

1. 边界条件：可以在产品说明书中有定义或者在使用软件过程中确定
2. 次边界条件（内部边界条件）：

2的乘方

一个字节8位组成，一个字2个字节组成,16=24

例如：允许输入1-1000的数字，则14、15、16，

254、255、256

ASCII表

1. 其他边界条件：如输入信息为空、非法、错误、 不正确和垃圾数据

|  |  |
| --- | --- |
| 项 | 范围或值 |
| 位（bit） | 0 或 1 |
| 字节（byte） | 0 ~ 255 |
| 字（word） | 0~65535（单字）或 0~4294967295（双字） |
| 千（K） | 1024 |
| 兆（M） | 1048576 |
| 吉（G） | 1073741824 |

|  |  |  |  |
| --- | --- | --- | --- |
| 字符 | ASCII码值 | 字符 | ASCII码值 |
| 空 (null) | 0 | A | 65 |
| 空格 (space) | 32 | a | 97 |
| 斜杠 ( / ) | 47 | Z | 90 |
| 0 | 48 | z | 122 |
| 冒号 ( : ) | 58 | 单引号 ( ‘ ) | 96 |
| @ | 64 |  |  |

### **边界值的选择**

如果输入条件规定了值的范围，则应该取刚达到这个范围的边界值，以及刚刚超过这个范围边界的值作为测试输入数据；（例如：0-50，0、50、51、-1）

如果输入条件规定了值的个数，则用最大个数、最小个数、比最大个数多1个、比最小个数少1个的数做为测试数据；（例如：运动员的参赛项目为1-3项，则0项、1项、3项、4项）

根据规格说明的每一个输出条件，使用规则（1）

根据规格说明的每一个输出条件，使用规则（2）

如果程序的规格说明给出的输入域或输出域是有序集合（如有序表、顺 序文件等），则应选取集合的第一个和最后一个元素作为测试用例；例 如：输出的表最多有999行，每50行为一页，则：输出0行、1行、50行、51行、999行

如果程序用了一个内部结构，应该选取这个内部数据结构的边界值作为

测试用例；

分析规格说明，找出其他可能的边界条件。

**常见的边界值**

对16-bit 的整数而言 32767 和-32768 是边界

屏幕上光标在最左上、最右下位置

报表的第一行和最后一行

数组元素的第一个和最后一个

循环的第 0 次、第 1 次和倒数第 2 次、最后一次

**与等价划分的区别**

边界值分析不是从某等价类中随便挑一个作为代表，而是使这个等价类的每个边界都要作为测试条件。

边界值分析不仅考虑输入条件，还要考虑输出空间产生的测试情况。

## **练习**

1. 用等价类划分Windows文件名称，应该分成（）个等价区

A 、2 B、3 C、4 D、6

• 2. 用边界值分析法，假定X为整数，10〈=X〈=100，那么X在测试中应该取（）边界值

1. X=10，X=100
2. B、X=9，X=10，X=100，X=101

C、X=10，X=11，X=99，X=100

D、X=9，X=10，X=50，X=100

3. 在某大学学籍管理信息系统中，假设学生年龄的输入范围为16-40，则根据黑盒测试中的等价类划分技术，下面划分正确的是（）

A、可划分为2个有效等价类，2个无效等价类

B、可划分为1个有效等价类，2个无效等价类

C、可划分为2个有效等价类，1个无效等价类

D、可划分为1个有效等价类，1个无效等价类

# **第八章 因果图**

**因果图的基本概念**

因果图是一种利用图解法分析输入的各种组合情况，从而设计测试用例的方法，它适合于检查程序输入条件的各种组合情况

**因果图的需要掌握的知识**

**原因和结果之间的关系**

与：只有2个原因都为真，那么结果为真（例如：北京姑娘，必须有车且有房）

或：2个原因中有一个为真时， 结果就为真。（例如：长沙姑娘，你有车或者有房

）

非：只有原因为假，结果才为真。（例如：你不好好学习，找到好工 作）

与非：先与后非。只要有一个 为假，结果就为真，2个都为真， 结果为假。（例子：有爸爸，有妈妈，不是一个幸福的孩子）

或非：先或后非（只要有1个为真，结果为假；两个为假，结果为真）

**原因和原因之间的关系**

包含性约束：各个原因中总有一个为真。即可以同时为真，但不可以同时为假。

排他性约束：各个原因之间不能同时为真，但可以同时为假（例如：小明不是11班学生，也不是12班学生）

必要性约束：当原因a为真时，原因b必须同时为真； 但是原因b为真时，原因a既可以为真，也可以为假。（例如： 2个原因--你是妈妈， 你结婚了）

唯一性约束：有且只有原因a和原因b中的一个为真。非此即彼，不存在第三种情况。例如：男女

**结果与结果之间的关系**

掩码标记（结果约束）：如果结果b为真，那么结果a一定为假，如果结果b为假，则结果a的状态不定。例如：2个结果—你变得难看了、你变得漂亮了

**因果图法设计测试用例的步骤**

1. 分析软件规格说明描述中, 那些是原因(即输入条件或输入条件的等价类),那些是结果(即输出条件), 并给每个原因和结果赋予一个标识符。
2. 分析软件规格说明描述中的语义，找出原因与结果之间, 原因与原因之间对应的关系，根据这些关系,画出因果图。
3. 由于语法或环境限制, 有些原因与原因之间,原因与结果之间的组合情况不可能出现，为表明这些特殊情况, 在因果图上用一些记号表明约束或限制条件。
4. 把因果图转换为判定表。
5. 把判定表的每一列拿出来作为依据,设计测试用例。

**因果图适合场合**

因果图适用于检查程序输入条件的各种组 合情况。利用因果图设计测试用例需要考 虑输入条件与输出结果之间的因果关系， 而这些因果关系很难从需求规格说明书中 直接得出，一般软件系统的因果关系非常 的复杂，所以利用因果图设计用例时费时、费力。

# **第九章 正交排列**

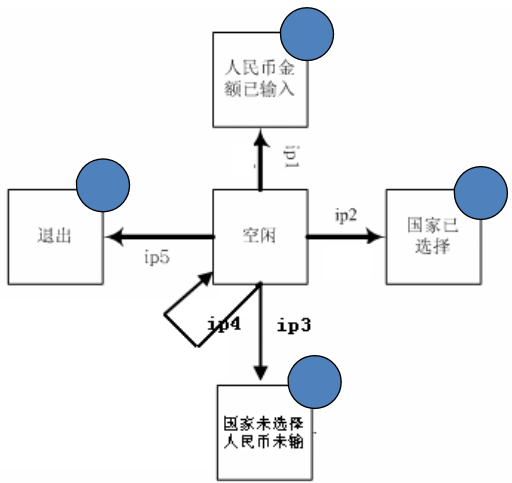
# **第十章 状态图法**

**如何画出状态图**

**第一步：**列出被测系统的输入事件



**第二步：**对空闲状态（程序刚启动时的状态）加所有可能的输入，判断产生哪些新状态。



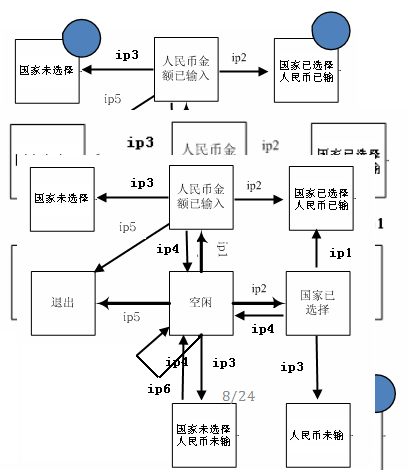
**第三步：**对第二步产生的每个新状态分别加所有可能的输入。

3.1 对“人民币金额已输入”加所有可能的输入。

3.2 对“国家已选择”再加所有可能的输入（图中加ip5输入的

3.3对“国家未选择、人民币未输”加所有可能的输入（ip6）

3.4对“退出”加所有可能的输入（没有）

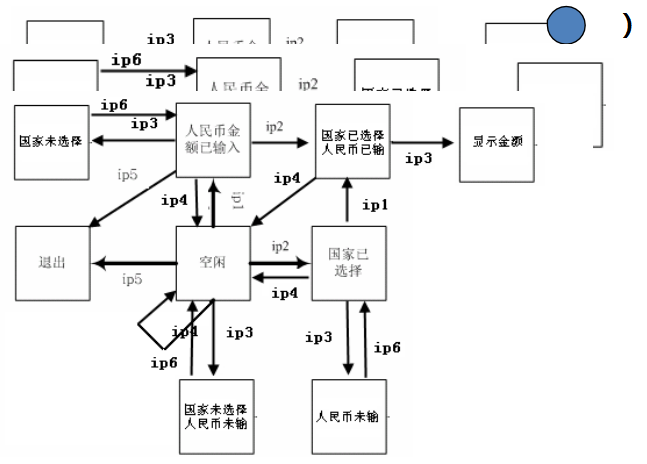


**第四步：**对第三步产生的每个新状态分别加所有可能的输入。

4.1 对“国家已选择、人民币已输”加所有可能的输入（省略了ip5）。

4.2对“国家未选择”加所有可能的输入（只有ip6

4.3对“人民币未输”加所有可能的输入（只有ip6)



**第五步：**对第四步产生的每个新状态分别加所有可能的输入。

5.1 对“显示金额”加所有可能的输入，经分析，不再有新的

状态产生，程序有如下9个状态：

空闲

遗漏国家和人民币

国家已选择

人民币已输入

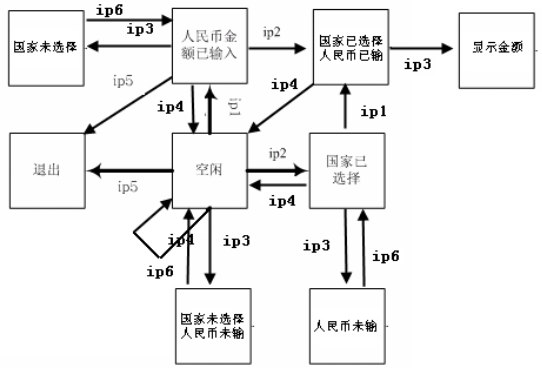
遗漏人民币信息

遗漏国家信息

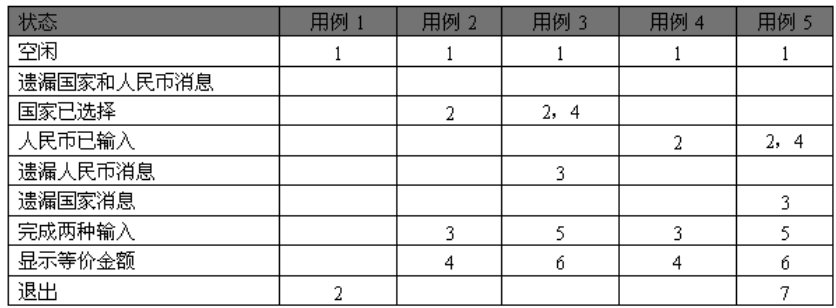
完成两种输入

显示等价金额

退出



**测试用例流程**



**减少测试用例的方法**

每种状态至少访问一次。

测试看起来最常见最普遍的状态转换。我们可以根据审查产品说明书时分析收集到的信息确定某些用户情况可能比其他更常见。

测试状态之间最不常用的分支。这些分支是最容易被产品设计者和程序员忽视的。

测试所有错误状态及其返回值。错误没有得到正确处理、错误提示信息不正确、修复错误时未正确恢复软件等情况是常有的。

利用工具自动执行状态转换测试。

# **第十一章 控件测试**

## **控件**

控件：是程序软件的组成部分，如软件中所经常涉及到的按钮、界面风格、表格、报表等，都可以直接使用控件。控件就像是一间房子里的桌椅家具一样，和整个房子的其它部份构成一套完整的住宅。

使用控件的必要性：

控件是程序开发过程中用于调用的部分，在整个程序中主 要用于方便调用，并且可以使用控件美化程序界面等，但 控件本身的开发强度较大，代码烦琐，如果将控件编写与 软件编写同时进行，则将导致程序及其代码显得比较臃肿， 并且大大地降低了工作效率，提高开发成本。因此，控件 的使用采用调用方式较为可取。而调用方式则是使用现有 的已经为程序准备好的控件。

**文本框测试**

文本框作用：接受用户输入的数据，或者显示数据

* + 是否允许输入
  + 输入汉字
  + 输入字符
  + 输入数字
  + 输入组合
  + 输入特殊字符
  + 输入保留字（9个字符），如果文本框中输入的是文件名
  + 输入空
  + 输入空格
  + 重名
  + 前含空格
  + 中含空格
  + 后含空格
  + 长度等于定义的最大长度
  + 长度刚好超过定义的最大长度：最大长度＋1
  + 长度小于定义的最大长度：最大长度－1
  + 长度远远超过定义的最大长度
  + 长度远远超过定义的最小长度
  + 输入默认值
  + 是否允许拷贝
  + 文本显示是否正确（业务和界面上是否正确）
  + 输入格式是否符合要求（日期，文件名等）
  + 是否适合用文本框的控件
  + 是否允许放大或者缩小

**按钮控件的测试**

按钮功能是否实现（添加，删除，修改，取消，保存等等）

对可能造成数据无法恢复的操作必须给出确认信息，给用户放弃选择的机会；（删除、关闭）

通过点击按钮有弹出框或者弹出窗体需要对弹出的窗体或弹出框进行测试，所弹出的窗体是否与按钮功能一致

按钮的可用与否逻辑上是否正确

对按钮测试需要考虑按钮对齐，字体大小，颜色，重复功能按钮等界面测试的要素

对非法的输入或操作给出足够的提示说明

**单选按钮的测试**

一组单选按钮不能同时选中，只能选中一个。

逐一执行每个单选按钮的功能。分别选择了“男”“女” 后，保存到数据库的数据应该相应的分别为“男”“女”；

一组执行同一功能的单选按钮在初始状态时必须有一个被默认选中，不能同时为空；

单选按钮一般被包含在GroupBox控件中，所以要明确Groupbox的名称是否与单选按钮所表达的含义一致

单选按钮是否可选（切换）

单选按钮有默认值和没有默认值

**up-down的测试**

直接输入数字或用上下箭头控制，如，在“数目”中直接输入10，或者单击向上的箭头，使数目变为10；

利用上下箭头控制数字的自动循环，如，当最多数字为

253时，单击向上箭头，数目自动变为1；反之亦适用；

直接输入超边界值，系统应该提示重新输入；

输入默认值，空白。如，“插入”数目为默认值，点击 “确定”；或，删除默认值，使内容为空，单击“确定” 进行测试；

输入字符。此时系统应提示输入有误。

**组合列表框的测试**

条目内容正确，其详细条目内容可以根据需求说明确定；

逐一执行列表框中每个条目的功能；

检查能否向组合列表框输入数据；

**复选框的测试**

复选框有且只有两种状态：选中或不被选中

每个选框能否被选中

能否被清空

能否同时选中

能否同时为空

每个选框的功能是否实现

需要考虑界面测试的要点：是否对齐，控件大小，字体等等

**列表框控件的测试**

条目内容正确；同组合列表框类似，根据需求说明书确定列表的各项内容正确，没有丢失或错误；

列表框的内容较多时要使用滚动条；

列表框允许多选时，要分别检查shift选中条目，按ctrl选中条目和直接用鼠标选中多项条目的情况；

**滚动条控件的测试**

滚动条的长度根据显示信息的长度或宽度及时变换，这样有利于用户了解显示信息的位置和百分比，如，word中浏览100页文档，浏览到50页时，滚动条位置应处于中间；

拖动滚动条，检查屏幕刷新情况，并查看是否有乱码；

单击滚动条；软件测试专业网站:

用滚轮控制滚动条；

滚动条的上下按钮。

**各种控件在窗体中混和使用时的测试**

控件间的相互作用；

tab键的顺序，一般是从上到下，从左到右；

热键的使用，逐一测试；

enter键和esc键的使用；

在测试中，应遵循由简入繁的原则，先进 行单个控件功能的测试，确保实现无误后， 再进行多个控件的的功能组合的测试

# **第十二章 界面测试**

界面测试要点

完成相同或相近功能的按钮用Frame框起来， 常用按钮要支持快捷方式。

完成同一功能或任务的元素放在集中位臵， 减少鼠标移动的距离。

按功能将界面划分区域块，用Frame框括起来,并要有功能说明或标题。

界面要支持键盘自动浏览按钮功能，即按Tab键、回车健的自动切换功能。

界面上首先要输入的和重要信息的控件在Tab顺序中应当靠前,位臵也应放在窗口上较醒目的位置。

同一界面上的控件数最好不要超过10个， 多于10个时可以考虑使用分页界面显示。

分页界面要支持在页面间的快捷切换，常用组合快捷键Ctrl+Tab

默认按钮要支持Enter操作，即按Enter后自动执行默认按钮对应操作。

界面空间较小时使用下拉框而不用选项框。

选项数較少时使用选项框，相反使用下拉列表框。

专业性强的软件要使用相关的专业术语， 通用性界面则提倡使用通用性词语。

常用菜单要有命令快捷方式。

完成相同或相近功能的菜单用横线隔开放在同一位置。

菜单前的图标能直观的代表要完成的操作。

菜单深度一般要求最多控制在三层以内。

工具栏要求可以根据用户的要求自己选择定制。

相同或相近功能的工具栏放在一起。

工具栏中的每一个按钮要有及时提示信息。

一条工具栏的长度最长不能超出屏幕宽度。

工具栏的图标能直观的代表要完成的操作。

工具栏太多时可以考虑使用工具箱。

工具箱要具有可增减性，由用户自己根据需求定制。

工具箱的默认总宽度不要超过屏幕宽度的1/5。

状态条要能显示用户切实需要的信息，常用的有：目前的操作、系统状态、用户位臵、用户信息、提示信息、错误信息等，如果某一操作需要的时间较长，还应该显示进度条和进程提示。

滚动条的长度要根据显示信息的长度或宽度能及时变换， 以利于用户了解显示信息的位置和百分比。

状态条的高度以放置五号字为宜，滚动条的宽度比状态条的略窄。

菜单和工具条要有清楚的界限;菜单要求凸出显示，这样在移走工具条时仍有立体感。

菜单和状态条中通常使用5号字体。工具条一般比菜单要宽，但不要宽的太多，否则看起来很不协调。

右键快捷菜单采用与菜单相同的准则。

帮助文档中的性能介绍与说明要与系统性能配套一致。(我们的系统帮助文档都是系统的祖先时期的说明，让人困惑)。

打包新系统时，对作了修改的地方在帮助文档中要做相应的修改。

操作时要提供及时调用系统帮助的功能。常用F1。

在界面上调用帮助时应该能够及时定位到与该操作相对的帮助位置。也就是说帮助要有即时针对性。

最好提供目前流行的联机帮助格式或HTML帮助格式。

用户可以用关键词在帮助索引中搜索所要的帮助，当然也应该 提供帮助主题词。

如果没有提供书面的帮助文档的话，最好有打印帮助的功能。

在帮助中应该提供我们的技术支持方式，一旦用户难以自己解决可以方便的寻求新的帮助方式。

父窗体或主窗体的中心位置应该在对角线焦点附近。

子窗体位置应该在主窗体的左上角或正中。

多个子窗体弹出时应该依次向右下方偏移，以显示窗体出标题为宜。

重要的命令按钮与使用较频繁的按钮要放在界面上注目的位置。

错误使用容易引起界面退出或关闭的按钮不应该放在易点击的位臵。横排开头或最后与竖排最后为易点位置。

与正在进行的操作无关的按钮应该加以屏蔽(Windows中用灰色显示，没法使用该按钮)。

对可能造成数据无法恢复的操作必须提供确认信息,给用户放弃选择的机会。

非法的输入或操作应有足够的提示说明。

对运行过程中出现问题而引起错误的地方要有提示，让用户明白错误出处，避免形成无限期的等待。

提示、警告、或错误说明应该清楚、明了、恰当。

长宽接近黄金点比例，切忌长宽比例失调、或宽度超过长度。

布局要合理,不宜过于密集，也不能过于空旷，合理的利用空间。

按钮大小基本相近，忌用太长的名称，免得占用过多的界面位臵。

按钮的大小要与界面的大小和空间要协调。

避免空旷的界面上放置很大的按钮。

放置完控件后界面不应有很大的空缺位置。

字体的大小要与界面的大小比例协调, 通常使用的字体中宋体9-12较为美观，很少使用超过12号的字体。

前景与背景色搭配合理协调，反差不宜太大，最好少用深色，如大红、大绿等。常用色考虑使用Windows界面色调。

如果使用其他颜色，主色调要柔和，具有亲和力与磁力， 坚决杜绝刺目的颜色。

大型系统常用的主色有"#E1E1E1"、"#EFEFEF"、"#C0C0C0"等。

界面风格要保持一致，字的大小、颜色、字体要相同，除非是需要艺术处理或有特殊要求的地方。

如果窗体支持最小化和最大化或放大时，窗体上的控件也要随着窗体而缩放；切忌只放大窗体而忽略控件的缩放。

对于含有按钮的界面一般不应该支持缩放，即右上角只有关闭功能。

通常父窗体支持缩放时，子窗体没有必要缩放。

如果能给用户提供自定义界面风格则更好，由用户自己选择颜色、字体等。

菜单通常采用“常用--主要--次要--工具--帮助”的位臵排列，符合流行的Windows风格。

常用的有“文件”、“編輯”，“查看”等，幾乎每個系統都有這些

選項，當然要根據不同的系統有所取捨。

下拉菜单要根据菜单选项的含义进行分组，並且按照一定的规则进行

排列，用横线隔开。

一组菜单的使用有先后要求或有向导作用时，应该按先后次序排列。

没有顺序要求的菜单项按使用频率和重要性排列，常用的放在开头， 不常用的靠后放臵；重要的放在开头，次要的放在后边。

如果菜单选项较多，应该采用加长菜单的长度而减少深度的原则排列。

菜单深度一般要求最多控制在三层以内。

对常用的菜单要有快捷命令方式

对与进行的操作无关的菜单要用屏蔽的方式加以处理，如果采用动态加载方式——即只有需要的菜单才显示——最好。

菜单前的图标不宜太大，与字高保持一直最好。

主菜单的宽度要接近，字数不应多于四个，每个菜单的字数能相同最好。

主菜单数目不应太多，最好为单排布臵。

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（如时钟显示）

下拉式操作能正确工作吗

菜单、调色板和工具条是否工作正确

是否适当地列出了所有的菜单功能和下拉式子功能

是否可能通过鼠标访问所有的菜单功能

相同功能按钮的图标和文字是否一致

是否能够用其他的文本命令激活每个菜单功能

菜单功能是否随当前的窗口操作加亮或变灰

菜单功能是否正确执行

菜单功能的名字是否具有自解释性

菜单项是否有帮助，是否语境相关

在整个交互式语境中，是否可以识别鼠标操作

如果要求多次点击鼠标，是否能够在语境正确识别

如果鼠标有多个按钮，是否能够在语境中正确识别

光标、处理指示器和识别指针是否随操作恰当地改变

安装界面上应有单位介绍或产品介绍，并有自己的图标。

主界面，最好是大多数界面上要有公司图标。

登录界面上要有本产品的标志，同时包含公司图标。

帮助菜单的“关于”中应有版权和产品信息。

公司的系列产品要保持一直的界面风格，如背景色、字体、菜单排列方式、图标、安装过程、按钮用语等应该大体一 致。

最重要的是排除可能会使应用非正常中止的错误。

应当注意尽可能避免用户无意录入无效的数据。

采用相关控件限制用户输入值的种类。

当用户作出选择的可能性只有两个时,可以采用单选框。

当选择的可能再多一些时,可以采用复选框,每一种选择都是有效的,用户不可能输入任何一种无效的选择。

当选项特别多时，可以采用列表框，下拉式列表框。

在一个应用系统中,开发者应当避免用户作出未经授权或没有意义的操作。

对可能引起致命错误或系统出错的输入字符或动作要加限制或屏蔽。

对可能发生严重后果的操作要有补救措施。通过补救措施用户可以回到原来的正确状态。

对一些特殊符号的输入、与系统使用的符号相冲突的字符等进行判断并阻止用户输入该字符。

对错误操作最好支持可逆性处理，如取消系列操作。

在输入有效性字符之前应该阻止用户进行只有输入之后才可进行的操作。

对可能造成等待时间较长的操作应该提供取消功能。

特殊字符常有；;’”><,｀‘：“［”｛、\|｝］+=)- (\_\*&&^%$#@!,.。?/还有空格。

与系统采用的保留字符冲突的要加以限制。

在读入用户所输入的信息时，根据需要选择是否去掉前后空格。

有些读入数据库的字段不支持中间有空格，但用户切实需要输入中间空格，这时要在程序中加以处理。

在多窗口系统中，有些界面要求必须保持在最顶层，避免用户在打开多个窗口时，不停的切换甚至最小化其他窗口来显示该窗口。

在主界面载入完毕后自动卸出内存，让出所占用的

WINDOWS系统资源。

关闭所有窗体，系统退出后要释放所占的所有系统资源， 除非是需要后台运行的系统。

尽量防止对系统的独占使用。

窗口能否基于相关的输入或菜单命令适当地打开

窗口能否改变大小、移动和滚动

窗口中的数据内容能否使用鼠标、功能键、方向箭头和键盘访问

当被覆盖并重调用后，窗口能否正确地再生

需要时能否使用所有窗口相关的功能

所有窗口相关的功能是可操作的吗

是否有相关的下拉式菜单、工具条、滚动 条、对话框、按钮、图标和其他控制可为 窗口可用，并适当地显示显示多个窗口时， 窗口的名称是否被适当地表示

活动窗口是否被适当地加亮

如果使用多任务，是否所有的窗口被实时更新

多次或不正确按鼠标是否会导致无法预料的副作用

窗口的声音和颜色提示和窗口的操作顺序是否符合需求

窗口是否正确地关闭

# **第十三章 文档测试**

### **哪些文档需要测试**

国家有关计算机软件产品开发文件编制指南中共有 14 种文件，可分为 3 大类。

开发文件：可行性研究报告、软件需求说明书、数据要求说明书、概要设计说明书、详细设计 说明书、数据库设计说明书、模块开发卷宗。

用户文件:用户手册、操作手册，用户文档的作用：改善易安装性；改善软件的易学性与易用性；改善软件可靠性；降低技术支持成本。

管理文件：项目开发计划、测试计划、测试分析报告、开发进度月报、项目开发总结报告。

**用户文档测试的方法**

技术校对

功能测试

其他辅助方式

### **文档测试要点**

文档的读者群；

文档的术语；

文档的正确性；

文档的完整性；

文档的一致性；

文档的易用性；

样例与示例；

文档的语言；

印刷与包装质量等等。

**文档测试的过程**

文档测试主要是查看文档并且进行相关的交流讨论。

### **文档测试与静态测试的关系**

静态测试只针对代码，

文档测试也用于设计文档。

静态测试与单元测试一样，是检查代码在功能上的正确性；针对代码的文档测试更注重代码与设计的一致，而代码在功能上的正确性则更多的由针对设计的文档测试来保证。

### **文档测试与评审的区别**

评审人的范围比较广，可以包括主管、项目经理、其他开发人员等

评审的时间相对较短

评审的随意性较大，系统性不强，评审人从各自的角度提出意见

文档质量较差时，评审人很难提出实质性

的意见

**总体设计的验证点**

检查需求中规定的功能点如何实现

需求中列出的所有功能点都能实现

检查需求中规定的性能指标如何保证

需求中列出的所有性能指标都能保证

检查普遍性的功能点（强壮性、容错性、安全性）如何实现

系统部分失效（断线重连、断线重启）

异常的输入数据

异常业务量（零负荷、超负荷）

非法入侵

检查普遍性的性能指标（可靠性、稳定性） 如何保证

业务处理能力

业务预期响应时间

最大支持用户数

检查模块定义是否正确

模块的功能描述明确

模块与模块的关系与现实关系一致

### **概要设计的验证点**

检查功能点（包括接口）如何实现

界面的输入项齐全

界面的输入项的数据类型、输入方式正确

界面的输出项齐全

界面的输出项的数据类型、输出方式正确

接口的输入参数齐全

接口的输入参数的数据类型正确

接口的输出参数齐全

接口的输出参数和返回值的数据类型正确

接口的输出参数和返回值能反映异常情况

本模块与其他模块的通讯与总体设计一致

算法和流程正确

检查性能指标如何保证

算法和流程高效

检查类定义是否正确

类的功能描述明确

类不可以再分为两个类

类与类的关系与现实关系一致

没有使用友类

检查类实现是否正确

输入正常的数据并经过正常的处理能得到正常的结果

输入异常的数据或处理过程中出现的异常能在输出参数或返回值中反映，或抛出异常

检查类实现是否容易理解

### **详细设计的验证点**

函数或方法用于正常处理的逻辑分支（循环和分支语句的个数）不会过多（最好5个以内，尽量不超过10个）

函数或方法用于正常处理的操作不会过多

（最好8个以内，尽量不超过15个）

函数或方法将产生的代码行不会过多（最好30行以内，尽量不超过60行）

代码的验证点

检查设计中规定的类和方法是否正确定义

类齐全

方法齐全

输入参数齐全

输入参数的数据类型正确

输出参数齐全

输出参数和返回值的数据类型正确

检查代码中的算法和流程是否与设计一致

设计中的流程分支有对应的代码分支

代码中不存在多余的分支

设计中描述的步骤有对应的代码

设计中描述的步骤有对应的注释

检查纠错机制是否完善

函数或方法的开始处有检查输入参数的合法性

调用函数或方法前有检查输入参数的合法性

调用函数或方法后有检查输出参数和返回值的合法性

指针操作前有检查指针是否为空

检查异常处理机制是否完善

函数或方法可能抛出的所有异常都有处理

未知的异常也有处理

检查数据初始化有否进行

局部变量有初始化

类属性有初始化

全局变量有初始化

未知的异常也有处理

检查数据初始化有否进行

局部变量有初始化

类属性有初始化

全局变量有初始化

内存申请后有初始化

检查代码的可读性、可修改性

文件名、类名、属性名、方法名、变量名、常量名等等命名符合规范

代码的缩进符合规范

代码的注释符合规范

类的声明符合规范

函数或方法用于正常处理的逻辑分支（循环和分支语句的个数）不会过多（最好5个以内，尽量不超过10个）

函数或方法的代码行不会过多（最好30行以内，尽量不超过60行）

各种特定的值被定义为常量

函数或方法的输入参数和输出参数没有被作为工作变量使用

## **用户文档的验证点**

读者对象——主要是文档的内容是否能让该级别的读者理解；

术语——主要是检查术语是否适合读者；

内容和主题——检查主题是否合适、是否丢失、格式是否规范等；

图标和屏幕抓图——检查图表的准确度和精确度；

样例和示例——是否与软件功能一致；

拼写和语法；

文档的关联性——是否与其它相关文档的内容一致，例如与广告信息是否一致；

### **第十四章**

### **兼容性测试概念**

兼容性主要是指协调性，硬件上是指计算机各个部件、CPU、显卡等等组装到一起以后的情况，会不会相互有影响，不能很好的运作。软件上是指软件之间能否很好的运做，会不会有影响、软件和硬件之间能否发挥很好的效率工作，会不会影响导致系统的崩溃。

### **兼容性测试与配置测试**

兼容性主要是指协调性，硬件上是指计算机各个部件、CPU、显卡等等组装到一起以后的情况，会不会相互有影响，不能很好的运作。软件上是指软件之间能否很好的运做，会不会有影响、软件和硬件之间能否发挥很好的效率工作，会不会影响导致系统的崩溃。

### **兼容性测试验证点**

平台测试

浏览器测试

软件本身能否向前或者向后兼容

测试软件能否与其它相关的软件兼容

数据兼容性测试

# **第十五章 易用性测试**

## **易用性测试的基本概念**

易用性（Useability）是交互的适应性、功能性和有效性的集中体现。易用性属于人体工程学的范畴，人体工程学

（ergonomics）是一门将日常使用的东西设计为易于使用和实用性强的学科。

### **易用性包括**

易理解性

易学习性

易操作性

吸引性

依从性

### **优秀UI具备的七个要素**

符合标准和规范

直观

一致

灵活

舒适

正确

实用

### **易用性测试与ＵＩ测试**

ＵＩ测试（又称“用户界面测试”）：用于与软件交互的方式称为用户界面或UI。是软件面向用户的主大门，直接影响到用户对软件系统的映像，及后期的使用等。但是，对其的测试仅仅是易用性测试的一个方面，是一个包含的关系。

分清了一件事物的这两个方面，在分析的时候会避免将所有的问题都归结于易用性问题。

### **易用性测试验证点**

控件名称应该易懂，用词准确，无歧异

常用按钮支持快捷方式

完成同一功能的元素放在一起

界面上重要信息放在前面

支持回车

界面空间小时使用下拉列表框而不使用单选框

专业性软件使用专业术语

对可能造成等待时间较长的操作应该提供取消

对用户可能带来破坏性的操作有回到上一步的机会

根据需要自动过滤空格

主菜单的宽度要接近

工具栏图标与完成的功能有关

快捷键参考微软标准

提供联机帮助

提供多种格式的帮助文件

提供软件的技术支持方式

# **第十六章 安装测试**

### **安装测试验证点**

安装程序名称为Setup

选择不同的安装模式安装

安装在不同的磁盘和路径

安装过程中有上一步、下一步的选项

在硬件资源不足下安装

C/S系统，安装服务端和客户端的顺序

在笔记本上安装

在安装前后检查注册表及安装文件或文件夹

安装时提供Key的输入

安装后再次安装

卸载后再次安装

### **运行测试验证点**

开始菜单、桌面快捷方式能够打开系统

软件能够使用

安装的目录和内容是否正确

### **卸载测试验证点**

通过添加/删除卸载

通过自带的卸载程序卸载

卸载前后检查注册表信息、文件、文件夹、图标

卸载正在使用的软件

卸载时突然中断

# **第十七章 WEB测试**

Web测试

### 上一章回顾

安装测试验证点

运行测试验证点

卸载测试验证点

### 课堂提问

安装测试要点

### 本章学习目标

能够从事Web测试

### 本章学习方法

准备Web项目进行测试

### 课前思考

如果需要买一辆新车，你要从哪些方面进行测试？

**美观测试**：外观造型、颜色、室内沙发、靠

1

背等配件位置和颜色搭配… …

**功能测试：**点火、启动、加速、行驶、转向 、

2

刹车、停车 … …

**性能测试:**启动加速到100千米/小时需要多少秒

3

、100公里耗油、最大时速… …

**兼容性测试：**配置是否为通用型、是否允许在高

4

速上行驶轮、胎是普通轮胎还是特制轮胎、… …

**安全测试：紧急刹车、抗撞击能力、安全气囊**

5

… …

### 本章主题

Web功能测试

Web性能测试

Web界面测试

Web兼容性测试

Web安全性测试

### 功能测试

链接测试

表单测试

Cookies测试

设计语言测试

数据库测试

### 链接测试

链接与页面对应

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

无空链接

测试所链接的页面是否存在

没有孤立的页面

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

### 链接测试方法

链接测试执行的时间一般在集成测试阶段， 即在整个Web应用系统的所有页面开发完成后进行链接测试。

链接测试可以运行手工测试；也可以通过自动化测试工具测试。

手工测试方法：点击任何一个可能有链接的地方，看页面和链接是否对应，看是否有空链接，看是否存在孤立的页面。

链接测试工具

Xenu Link Sleuth

HTML Link Validator

ACT

Rational sitecheck

Rational linkbot

演示Xenu Link Sleuth

### 链接测试案例

案例演示：点击链接无反应

错误现象：

原因：

超链接地址有误

解决办法

更正超链接地址

### 表单测试

当用户给Web应用系统管理员提交信息时， 就需要使用表单操作

必须测试提交操作的完整性，以校验提交给服务器的信息的正确性。

### 表单测试事例

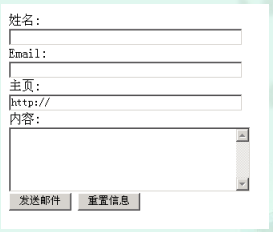
示例表单将检查如下功能：

姓名是否为空

Email地址是否为空，是否包含“@”和“.”

主页网址是否为空，是否包含“http://” “.”

内容是否为空



表单测试事例

案例演示：表单提交信息不完整

错误现象及重现步骤：

### 表单测试事例



链接测试案例

案例演示：链接页面不存在

错误现象及重现步骤：

原因：链接页面不存在

### Cookies测试

Cookies通常用来存储用户信息和用户在某应用系统的操作，当一个用户使用Cookies 访问了某一个应用系统时，Web服务器将发送关于用户的信息，把该信息以Cookies的形式存储在客户端计算机上，这可用来创建动态和自定义页面或者存储登陆等信息

测试的内容可包括Cookies是否起作用，是否按预定的时间进行保存，刷新对Cookies 有什么影响

### 设计语言测试

Java、JavaScript、 ActiveX、VBScript或Perl

等要进行验证

### 数据库测试

数据一致性错误

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

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

### 功能测试小结

性能测试

连接速度测试

负载测试

压力测试

### 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，他们或许是电话拨号，或是宽带上网

如果Web系统响应时间太长（例如超过5秒钟），用户就会因没有耐心等待而离开

有些页面有超时的限制，如果响应速度太慢， 用户可能还没来得及浏览内容，就需要重新登陆了

连接速度太慢，还可能引起数据丢失，使用户得不到真实的页面

### 负载测试

根据场景设计测试用例

100用户登录邮箱

200用户查看新闻

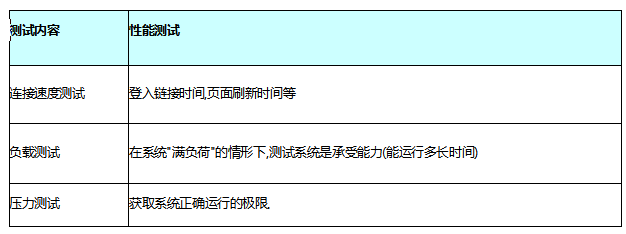
300用户查看科技

### 压力测试

并发测试

500人同时登录邮箱

性能测试小结



界面测试

导航测试

图形测试

内容测试

整体界面测试

### 导航测试

方便快捷的访问到用户需要的信息

在任何页面上都可以清楚地知道页面所处

Web应用系统中的位臵

页面逻辑结构清晰，层次分明

容易返回上一状态或主页面

### 图形测试

确保图形有明确的用途

图形无法显示

图形下载速度缓慢

放臵重要信息的图片丢失

所有页面字体的风格一致

背景颜色应该与字体颜色和前景颜色相搭配

图片的大小和质量影响性能

### 内容测试

检验Web应用系统提供信息的正确性、准确性和相关性

### 整体界面测试

设计风格一致性

### 界面测试小结



兼容性测试

平台（操作系统）测试

浏览器测试

分辨率测试

### 安全性测试

先注册，后登陆

登录密码控制

用户登陆后在一定时间内（例如15分钟） 没有点击任何页面，是否需要重新登陆才能正常使用

日志

能够在服务器端放臵和编辑脚本

**兼容性、安全性测试小结**



总结

Web功能测试

Web性能测试

Web界面测试

Web兼容性测试

Web安全性测试

请预习第十八章

### 扩展一

**Internet**

**界面测试**

**功能测试**

**性能测试**

**兼容性测试**

**安全测试**

### 安全性测试术语介绍

1、什么是“易受攻击性”？

这是网络应用程序的一个软肋。造成这个“软肋”的原因，可能是程序中的

bug，一种注入（SQL/脚本代码）或者已存在的病毒。

2、什么是“URL处理”？

一些网络应用程序通过URL在客户端（浏览器）和服务器端之间进行额外信

息的传递。有时在URL中改变信息可能会导致服务器不可预期的结果。

3、什么是“SQL注入”？

是指通过网络应用的用户接口插入SQL指令到服务器所执行的查询中去的过

程。

4、什么是“XSS（跨站式脚本攻击）”？

当一个用户通过Web应用接口插入HTML代码，而这种嵌入其中的代码对于其

他用户是可见的，被称做XSS。

5、什么是“欺骗”？

完全仿造网站或电子邮件的赝品被称为欺骗

### 2011十大安全漏洞

对比2004,2007年十大安全漏洞,几乎每年的漏洞情况都是变动，随着Web的发展,

都会出现新的安全漏洞，或者原来的安全漏洞优先级靠前.

A1 注入式漏洞

A2 跨站脚本

A3 无效的验证和会话管理 A4 对不安全对象的直接引用A5 伪造的跨站请求

A6 安全配置错误

A7 加密存储方面的不安全因素

A8 不限制访问者的URL

A9 传输层面的保护力度不足

A10 未经验证的重新指向及转发

### 扩展二：怎样开展Web测试

获取测试需求，功能列表或测试矩阵

执行功能测试，同时完成界面测试、易用性测试，并将安全测试思想加入到功能测试中，进行简单的安全测试

兼容性测试

如有要求，深入执行安全测试

性能测试

### 上机课作业

针对当前的Web系统编写Web方面的测试用例，并进行测试

### 家庭作业

将本PPT各小结中的测试项和测试内容手工输入到EXCEL中，做成一个通用测试用例

查找B/S与C/S结构的区别

# **第十八章**

数据库测试

### 上一章回顾

Web功能测试

Web性能测试

Web界面测试

Web兼容性测试

Web安全性测试

### 课堂提问

Web测试需要测试哪些内容

### 本章学习目标

数据库测试分类

数据库测试验证点

### 本章学习方法

运用

### 本章主题

数据库测试的执行者

数据库测试所需要的知识

数据库测试分类

数据库测试验证点

### 数据库执行者

开发人员

DBA

测试人员

### 数据库测试所需要的知识

了解数据库

了解SQL

能够使用SQL 操作数据库

具备优化数据库的能力

### 数据结构的基本知识

数据结构

定义一、数据元素集合（也可称数据对象）中各元素的关系。

定义二、相互之间存在特定关系的数据元素集合。

一个表( 数据库)，我们就称它为一个数据结构，它由很多 记录 ( 数据元素 )组成，每个元素又包括很多字段( 数据项)组成

### 数据库测试分类

系统测试

集成测试

单元测试

功能测试

性能测试

安全测试

### 系统测试

数据库在初期设计中需要进行分析测试.

存储过程，视图，触发器，约束，规则等我们都需要进行需求的验证确保这些功能设计是符合需求的.

确认数据库设计文档和最终的数据库相同， 当设计文档变化时我们同样要验证修改是 否落实到数据库上。

数据库设计评审来实现

### 集成测试

数据项的修改操作

数据项的增加操作

数据项的删除操作

数据表增加满

数据表删除空

删除空表中的记录

数据表的并发操作

针对存储过程的接口测试

结合业务逻辑做关联表的接口测试 ，需要对这些接口考虑采用等价类、边界值、错误猜测等方法进行测试

### 单元测试

单元测试侧重于逻辑覆盖，数据库开发的单元测试相对简单

语句覆盖

通过走读方式

### 功能测试

DBunit

一款开源的数据库功能测试框架，可以使用类似与Junit的方式对数据库的基本操作进行白盒的单元测试，对输入输出进行校验。

QTP

通过对对象的捕捉识别，我们可以通过QTP来模拟用户的操作流程，通过其中的校验方法或者结合数据库后台的监控对整个数据库中的数据进行测试。个人觉得比较偏向灰盒。

DataFactory

一款优秀的数据库数据自动生成工具，通过它你可以轻松的生成 任意结构数据库，对数据库进行填充，帮助你生成所需要的大量 数据从而验证我们数据库中的功能是否正确。这是属于黑盒测试。

### 数据库性能

性能优化分4部分

物理存储方面

逻辑设计方面

数据库的参数调整

SQL语句优化

### 数据库性能

数据库系统的SQL语句分析工具，分析得到数据库语句执行的瓶颈，从而优化SQL语句

Loadrunner

通过对协议的编程来对数据库做压力测试

Swingbench

专门针对oracle

oracle11g提供了real application test，提供数据库性能测试，分析系统的应用瓶颈

### 数据库安全测试

SQL 注入攻击、跨站点脚本攻击、未经授权的用户访问

所谓SQL注入（SQL Injection），就是利用程序员对用户输入数据的合法性检测不严或不检测的特点，故意从客户端提交特殊的代码，从而收集程序及服务器的信息，从而获取想得到的资料。通常别有用心者的目标是获取网站管理员的帐号和密码。

数据库强大的存储过程，黑客可以轻松的获得整个系统的权限

### 总结

数据库测试的执行者

数据库测试所需要的知识

数据库测试分类

数据库测试验证点

请预习第十九章

# **第十九章**

功能测试

### 上一章回顾

数据库测试分类

数据库测试验证点

### 课堂提问

数据库如何测试

### 本章学习目标

能够针对产品或者项目功能设计功能性测试用例

### 本章学习方法

准备项目，设计功能测试用例

### 本章主题

功能测试的内容

功能测试验证点

### 功能测试的内容

添加

删除

修改

查询/浏览

### 设计功能测试用例－添加[特]

只添加主表

只添加从表

能够先选择主表，再在主表的基础上添加从表数据

不影响主表原有数据

一个从表数据同一时间只能添加到一个主表数据中

同时添加主从表

首先添加主表，接着提供添加从表数据的界面

从表添加完成后，提交主表，主从表数据成功保存

只保存从表添加，主表添加失败，本次操作失败

### 添加[公]

必输项是否有必须输入的标记

能够成功添加

添加后每项数据核查正确

保存后跳转页面正确

如果是添加附件文件，能否正确上传附件文件；

添加后能够使用本次添加的数据

添加后在相关查询中可以查询到

### 修改

修改：修改与添加相关需要考虑到修改的编辑框中可以输入哪些类型的数据、数据长度等等，另外还要考虑到以下内容

能够修改哪些内容

不能修改的地方应该为只读

能够修改成功

修改主表数据不影响从表数据

修改从表数据不影响主表数据

修改主从表数据后不影响已经引用的地方

修改后在相关处查询为修改后的数据

### 设计功能测试用例－删除[特]

删除主表

主表未被其他地方引用，可以成功删除

主表被其他地方引用，不能删除，必须先删除引用，再删除主表

主表下有从表，删除主表，从表同时被删除或者主表下有从表，必须先删除从表才能删除主表

### 设计功能测试用例－删除[特]

删除从表

从表未被其他地方引用，可以成功删除

删除后，不能使用本从表数据，但不影响主表数据使用

从表被其他地方引用后，不能删除，必须先删除引用，再删除从表

### 设计功能测试用例－删除[特]

同时删除主从表

主表未被其他地方引用，可以同时成功删除主从表

### 删除[公]

必须有“确定删除”的提示信息，给用户放弃破坏性操作的机会

是一般删除还是破坏性删除（彻底删除，从数据库中删除）

是否有相关的数据删除，如果有要确认该相关的数据也已经删除，并且在同一事务中完成

是否有删除约束

只有拥有相关权限的用户才能删除，是否按照权限删除

是否支持Ctrl、Shift多条删除、全部删除

删除空记录

是否支持全部删除

### 设计功能测试用例－查询

查询：查询功能相对简单，但体现了数据的流向与正确性，可以从以

下方面考虑，注意可以使用正交排列法

支持全部查询

按照任意条件可以正确查询出数据

支持任意组合查询

因为各业务引起的数据变化，在查询中能够正确体现

查询结果准确

查询出的数据量大，有分页显示功能

下翻、上翻页正确

可以跳转到任意页

有查询结果说明，如本页多少条数据，共查询出多少条数据

分页的统计数字是否正确，共X页，第N页，共X条记录等；

对于主从表可以查询出主表数据和从表数据

支持模糊查询

支持精确查询

当查询的数据非常多的时候，性能有无问题

对于数据量比较大的表查询时，不容许无条件查询，避免性能问题的出现；

查询数据是否正确

### 设计测试用例的方法－正交排列

正交排列法

正交的由来

正交表测试策略：是一种成对测试交互的系统的统计的方法。它提供了一种能对所有变量对的组合进行典型覆盖（均匀分布）的方法。

运用的范围：

对软件组件的集成测试尤其有用（尤其对面向对象的系统，对客户端来说很多子类能够被代替做为服务器）；在配置选项组合的测试中也很有用（比如一个让用户选择字体、背景颜色及页面布局的网页）

使用最小的测试过程集合获得最大的测试覆盖率

当可能的输入数据或者这些输入数据的组合数量很大时，由于不可能为每个输入组合都创建测试用例，这种方法有效

### 正交排列法

如何查找正交表

Technical Support (support.sas.com)

<http://support.sas.com/techsup/technote/ts723_Designs.t> xt

查Dr. GenichiTaguchi设计的正交表

[www.york.ac.uk/depts/maths/](http://www.york.ac.uk/depts/maths/)

[http://tables/orthogonal.htm上面查询](http://tables/orthogonal.htm%E4%B8%8A%E9%9D%A2%E6%9F%A5%E8%AF%A2)

数理统计、试验设计等方面的书及附录中

### 正交排列法

案例分析

测试一个整容软件设置功能，测试过程中需要考虑一下4个方面的问题

鼻子：鹰勾鼻，蒜头鼻，高翘鼻

眼睛：丹凤眼，斗鸡眼，斜视眼

嘴巴：樱桃小口，血盆大口，中口

脸型：马脸，圆脸，瓜子脸

### 正交排列法

分析组合情况：3×3×3×3＝？

分析是否符合正交法？

当有大量变量，并且每个变量都有几个已定义状态的时候

4个变量：鼻子，眼睛，嘴巴，脸型，每个变量是3个状态

鼻子：鹰勾鼻，蒜头鼻，高翘鼻

眼睛：丹凤眼，斗鸡眼，斜视眼

嘴巴 ：樱桃小口，血盆大口，翘嘴

脸型：马脸，圆脸，瓜子脸

L9 （34），找到正交表

L9（34），A，B，C，D：是变量，123是变量的取值

### 查找正交表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **A** | **B** | **C** | **D** |
| **1** | **1** | **1** | **1** | **1** |
| **2** | **1** | **2** | **2** | **2** |
| **3** | **1** | **3** | **3** | **3** |
| **4** | **2** | **1** | **2** | **3** |
| **5** | **2** | **2** | **3** | **1** |
| **6** | **2** | **3** | **1** | **2** |
| **7** | **3** | **1** | **3** | **2** |
| **8** | **3** | **2** | **1** | **3** |
| **9** | **3** | **3** | **2** | **1** |

值替换、设计用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **A[鼻子]** | **B[眼睛]** | **C[嘴巴]** | **D[脸型]** |
| **1** | **1[鹰勾鼻]** | **1丹凤眼** | **1樱桃小口** | **1马脸** |
| **2** | **1[鹰勾鼻]** | **2斗鸡眼** | **2血盆大口** | **2 圆脸** |
| **3** | **1 [鹰勾鼻]** | **3斜视眼** | **3翘嘴** | **3 瓜子脸** |
| **4** | **2蒜头鼻** | **1丹凤眼** | **2血盆大口** | **3瓜子脸** |
| **5** | **2蒜头鼻** | **2斗鸡眼** | **3翘嘴** | **1马脸** |
| **6** | **2蒜头鼻** | **3斜视眼** | **1樱桃小口** | **2圆脸** |
| **7** | **3高翘鼻** | **1丹凤眼** | **3翘嘴** | **2圆脸** |
| **8** | **3高翘鼻** | **2斗鸡眼** | **1樱桃小口** | **3瓜子脸** |
| **9** | **3高翘鼻** | **3斜视眼** | **2血盆大口** | **1马脸** |

功能测试用例演示

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 操作步骤 | 测试要点与测试数据 | 预期输出 |
| YYGL01 | 选择客户管理-〉客户资料管  理  选中某级项目夹  单击‘增加组’按钮 | 新项目夹名称为：一般字符、数字、不同级项目夹相同名称；特殊字符、空、同级与其他项目夹相同字符  在根目录下、根目录下级目录下增加组  增加10层组 | 项目夹名称不能为空，可以为其它字符和数字；同级名称不能相同  可以在任意目录下新增组  可以无限增加组 |
| YYCZ02 | 选择客户管理-〉客户资料管理  选中某级项目夹  单击‘修改组’按钮 | 将项目夹名称修改为： 一般字符、数字、不同 级同名；特殊字符、空、同级同名字符  修改根目录  该组在贷款业务中被引  用 | 同级项目夹名称不能修改同名，不能修改为特殊字符、空；其他允许  根目录不能修改，  修改按钮为灰  在贷款业务中也被  修改 |
| YYCZ03 | 选择客户管理-〉客户资料管  理  选中某级项目夹  单击‘删除组’按钮 | 删除组下无客户的组  删除组下有客户的组  删除该组在贷款业务中  被引用的组  删除根目录 | 成功删除  组及组内客户都删  除  允许删除  删除按钮为灰 |

功能测试用例演示续

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **用例**  **编号** | **相关**  **用例** | **操作步骤**  **输入数据** | **预期结果** | **备注** |
| Gh00  1 |  | 输入正确的数据  单击“确定”按钮 | 挂号成功 |  |
| GH00  2 |  | 输入姓名，选择科别  选择类型  输入应收  单击“确定”按钮 | 应收中自动显示该类型  的应收金额  自动计算找零金额 |  |
| GH00  3 |  | 输入正确的数据  勾选“打印前预览” | 2、正确显示预览信息，显  示的数据项与金额正确 |  |
| GH00  4 |  | 姓名、科别、类型、预收金额任意  一项为空，其他正确输入  单击“确定”按钮 | 2、提示：“姓名/科别/类型/  预收不能为空” |  |

总结

功能测试的内容

功能测试验证点

请预习第二十章

# **第二十章**

业务测试

### 上一章回顾

功能测试的内容

功能测试验证点

### 课堂提问

添加如何测试

查询如何测试

### 本章学习目标

能够对业务系统设计测试用例

### 本章学习方法

运用

### 本章主题

业务测试的概念

业务测试的验证点

业务测试的概念

业务测试是测试人员把系统各个模块串接 起来运行、模拟真实用户实际的工作流程， 满足用户需求定义的功能来进行测试的过 程。

### 业务测试的测试方法

业务测试关注需求和用户

设计业务用例

测试执行

### 业务测试关注需求和用户

站在用户的角度

测试人员最好能够全程参与整个开发过程，尤其是需求解决要 及早介入到需求，多与客户沟通，真正理解用户手工的业务流 程，尽量减少业务理解的偏差。

重点关注整体业务和分业务

在进行业务测试时，是在功能测试成功实施的基础上进行的测试，业务测试的工作重点应该是放在尽可能全面的收集模块需 求、了解整体的业务流程、分析主体业务流程和重点业务流程 等工作上。

现场客户

现场客户随时提供对需求细节的指导。如果没有条件，可以定 期的邀请用户参加项目例会或安排和用户交流等。另外在需求 理解评审和测试设计评审会尽量邀请用户参与。

### 设计业务用例

业务流程测试用例同样可以采用边界值和等价类划分的方法。

对于业务系统的测试需要考虑基础数据、业务数据。基础数据一般采用客户真实的数据，业务数据要符合实际的业务流程。

一般情况下每一个典型的业务操作就是一个业务流程

业务流程可以用场景法写，针对一个业务流程设计一个或者多个场景

业务流程无需覆盖到所有的功能，只要覆盖到用户的典型业务。业务是贯穿多个功能模块，不受到业务属于哪个功能模块的限制。

在设计业务用例时，需要理清系统的业务流程，可以采用相关的辅助手段理清业务，例如画总体业务流程图以及分业务流程图等

### 测试执行

在系统测试每轮测试保持测试数据库都是完整的一套初始数据，在每次测试之前保证数据的原始状态

一般在版本比较稳定的情况下可以采用自动化工具录制业务流程测试脚本实现整个业务测试的多轮测试过程

### 业务测试用例演示

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编  号 | 业务描述 | 测试要点与测试数据 | 预期输出 |
| YW01 | 客户资料管理中维护客户  信息  选择信贷银行及贷款类别  ，新建个人住房按揭贷款  ，完整输入该客户信息  对该客户进行贷款回收 | 1. 能否正常完成信贷业  务完整流程 | 1. 能完成从客户资料管理到、新建贷款、到还款的完整流程 |
| YW02 | 选择信贷银行及贷款类别， 新建个人住房按揭贷款， 不对该新客户进行维护  （不输入任何信息）  对该新客户进行贷款回收 | 1. 对信息维护不完整的客户不能进行贷款回收 | 1. 提示该客户输入信息不完整，不能回收贷款 |
| YW03 | 新建贷款  贷款回收中对该贷款余额  全部回收  对已经还清的贷款进行贷款回收  对已经还清的贷款进行贷款展期 | 能否完成全额提前还  款  对已经还款后的贷款能够进行贷款回收与贷款展期 | 1. 本贷款结算状态改为已结，不能对该贷款进行贷款回收业不能进行贷款展期 |

业务测试用例演示

|  |  |  |
| --- | --- | --- |
| 用例编  号 | 预期输入及操作步骤 | 预期输出 |
| Zyyw1 | 业务描述：住院登记-〉交押金->录入医嘱-〉出院  选择住院管理-〉住院基本资料，正确输入相关信息后保 存  选择住院管理-〉住院预交款费单录入；  双击某住院病人信息  在预交款中输入预交款如:1000,点‘保存’ | 在数据列表中显示刚才添加  的住院病人基本信息  在数据列表中显示住院患者信息  该病人相关信息显示在相应  的文本框中  预交款信息显示在预交款数  据显示区域中  处方信息显示在费单数据显示区域中，在药房管理的药房库存查询种所开医嘱的药品数量不变化  在左边的数据列表中显示出  待发药的处方信息  摆药信息显示在数据列表中  发药成功，在药房库存查询中，该处方药品数据减少发药的数据量  病人信息显示在相应的数据  区中  显示押金信息和费用信息费  用计算结果正确  预览发票信息，相关信息显示正确  打印收据  提示病人出院 |
|  | 在费单中输入医嘱，并点‘保存’  药房管理-〉住院处方发药，选择当前处方日期，点查询  点击“摆药”  点击“发药”  住院管理-〉出院结算，首先查询出出院病人信息，双击  出院病人信息  点‘计算费用’  点‘出院结账’  回车  点‘出院’ |

业务测试用例续

，

|  |  |  |
| --- | --- | --- |
| Zyyw2 | 业务描述：已出院病人直接交押金  （1）住院管理-〉住院预交款费单录入-〉输入一个  已经结账、出院的病人住院号 | （1）不能查询出该病人信息  提示该病人已经出院 |
| Zyyw3 | 业务描述：已出院病人再次出院  （1）住院管理-〉出院结算，输入已经结账、出院  的病人住院号，点查询 | （1）无法查询出该病人信息不能再次进行出院结算 操作 |
| Zyyw4 | 业务描述：已出院病人到药房领药  （1）药房管理-〉住院处方发药，点查询 | （1）不能查询出已出院病人  的处方信息 |
| Zyyw5 | 业务描述：住院登记-〉不交押金-〉录入医嘱   1. 住院管理-〉住院基本资料录入并保存 2. 住院管理-〉住院预交款管理，不进行预交款操   作，直接录入费单的医嘱信息 | （2）提示没有交押金 |

，

### 总结

### 业务测试的概念

业务测试的验证点

请预习第二十一章

### 作业

上机作业

编写当前系统的分业务测试用例

由一人编写一个主线业务（整体业务）测试用例

家庭作业

编写商场退货业务测试用例

# **第二十一章**

软件测试故障模型

### 上一章回顾

业务测试的内容

业务测试验证点

### 课堂提问

业务测试的概念

业务测试需要关注什么

业务测试用例的注意事项

### 本章学习目标

熟练掌握二十一种故障模型

### 本章学习方法

运用

### 内容进度

故障模型

功能性测试的测试方法

用户接口输入测试

用户接口输出测试

### 故障模型

故障模型概念

设计测试用例时有太多的单个输入变量、 多个输入变量的组合，优秀的软件测试人 员不会依靠运气，他们有着丰富的经验和 直觉，可以从中找到哪些是要进行测试的， 哪些不需要测试，哪些操作可能会引起软 件失效。我们把这些测试人员的经验和直 觉尽量归纳和固化，形成一些故障模型

（Fault Model）。

为软件测试工程师敏锐发现缺陷提供帮助

### 内容进度

故障模型

功能性测试的测试方法

用户接口输入测试

用户接口输出测试

方法1：输入非法数据

案例演示

原理分析

处理非法输入的方法

输入时过滤非法数据，

程序内部捕获错误信息

如何发现这类错误

举例：假设“软件测试工个文件进行保存，保存文则保存的该工程师信息的意工程师姓名输入的隐含

输入非法类型：文件名中不能包括的9个非法字符，系统保留字等；

输入超长字符：255个字符；

注意

检查错误信息，保证正确、易懂！

举例： 错误信息：Error 5-unkown data！

### 方法1：输入非法数据

**实战演练**

### 方法2：输入默认值

案例演示

环境：Word2000（可

此类缺陷产生原因

定义变量时未赋初值

赋初值不正确

再次赋初值后对程序

如何发现这类错误

测试方法小结

全面理解需求求

同时深刻理解

实战演练

方法3：输入特殊字符集

**案例演示**

**环境：Win2000、IE5**

**此类缺陷产生原因**

**特殊字符处理问题，没有对特殊字符输入做程序处理**

**注意系统保留字符串**

**注意应用程序处理特殊字符**

**C语言中的“\n”、“++”、“&”等**

**如何发现这类错误?**

**测试方法小结**

**实战演练**

方法4：输入使缓冲区溢出的数据

案例演示

环境：Win2000、Word2000

此类缺陷产生原因

输入的数据未经检查，超过该值固定大小内存缓冲区，影响其他内存单元，严重的引起程序关闭。

如何发现这类错误

获得需求（包括详细设计说明），输入最大字符串和超过最大字符串要求的输入数据

测试方法小结

加强和开发人员沟通，了解没有写到需求或设计文档中的变量范围

实战演练

方法5：输入产生错误的合法数据组合

**案例演示**

**在Word中插入表格，需求规格说明书中规定：列容许的最大值为**

**63，行容许的最大值为32767**

**输入：列=55，行=32005，结果？**

**此类缺陷产生原因**

**测试多个输入值的组合，每个合法输入值单独测试通过不代表合**

**法输入值的组合测试也能通过。**

**不过此例应用程序只是挂起，等待一段时间后，Word还是可以产 生所需要的表格，所以此例是否确定为缺陷可以和需求或开发人 员沟通，建议的做法是界面给出产生产生表格进度条。**

**如何发现这类错误?**

**测试方法小结**

**尽可能多的了解程序内部数据结构，多与开发人员沟通。**

**实战演练**

用户接口输入测试小结

输入非法数据

输入默认值

输入特殊字符集

输入使缓冲区溢出的数据

输入产生错误的合法数据组合

### 内容进度

故障模型

功能性测试的测试方法

用户接口输入测试

用户接口输出测试

**案例分析**

**输入：一个电话打来**

**输出：**

**状态一：如果此电话正在使用，则打来电话的人听到的声音应该是**

**占线的提示音。**

**状态二：如果此时电话未使用，则打来电话的人听到的声音应该是等待接听的提示音。**

**缺陷产生原因**

**开发人员可能没有判断当前所处状态，就想当然的给出了输出。**

**如何发现这类错误**

**熟悉被测软件业务知识，阅读各种程序文档，明确输入可能产生**

**的输出。**

### 方法7：产生不符合业务规则的无效输出

**案例演示**

**缺陷产生原因**

**程序开发人员对业务了解不深刻**

### **如何发现这类错误?**

### 用户接口输出测试小结

产生同一输入的各种可能输出

强制产生不符合业务规则的无效输出

### 内容进度

用户接口输出测试

数据结构的测试

### 方法8：输出属性修改后的结果

案例演示

输出具有可修改的属性

本案例是否为缺陷可以根据需求做进一步判断

缺陷产生的原因

开发人员在创建对象时设立了初始值，但当用户修改输出对象属性，开发人员编写的对应代码没有考虑这些属性值的修改对其他变量的影响。

如何发现这类错误及测试方法小结?

### 方法9：检查屏幕刷新

案例演示

缺

慢；刷新频率

现的现象。

–

如

测试方法小结

注意增加、删除和移动屏幕上的对象能发现类似的缺陷

陷产生的原因

刷新频率快了，程序运行变慢了，则会出现案例演示出刷新范围控制

何发现这类错误?

### 用户接口输出测试小结

产生同一输入的各种可能输出

强制产生不符合业务规则的无效输出

强制通过输出修改属性

检查屏幕刷新

### 内容进度

用户接口输出测试

数据结构的测试

### 方法10：数据结构溢出

案例演示

缺陷产生的原因

数据结构限制

内存限制

硬盘限制

如何发现这类错误

上溢

下溢

测试方法小结

数组

### 方法11：数据结构不符合约束

案例演示

缺陷产生的原因

在建立数据项时对数据属性的约束进行了检查， 而修改数据项的代码未做约

如何发现这类错误

修改属性判断是否进行约

测试方法小结

了解内部数据结构约束，尝行测试。

### 方法12：操作数和操作符不符

案例演示

是否是缺陷

？

如果是缺陷，开发人员修改成什么样的结果你作为测

修复。

如何发现这

找到程序中容易引起操作数和操作符不符的计算、表达式等。

实战演练

方法13：函数递归调用

案例演示

Excel案例演示

缺陷产生的原因

函数递归调用，没有 能会导致系统

死机。•

教材中的例子需要的环境

Win2000，Word2000

### 方法14：计算结果溢出

案例分析

– 如果value[0]=32700， value[1]=70，则？

•

里值

– 输入非法值，强制数据产生溢出，观察程序的处理情况。

### 方法15：数据共享或关联功能出错

案例演示

时， 功能

和测试方法小结

### 数据结构的测试小结

数据结构溢出

数据结构不符合约束

操作数与操作符不符

递归调用自身

计算结果溢出

数据共享或关联功能计算出错

### 本章学习目标

文件系统的测试

软件的故障模型

### 内容进度

文件系统的测试

软件的故障模型

### 方法16：使文件系统超载

案例

假设“软件测试工程师管理系统”要保存10000个工程师信息，则保存时engineer.txt文件会有20M大小，如果此时磁盘只有10M可用空间了， “软件测试工程师管理系统”会如何动作呢？

此类缺陷产生的原因

开发人员忽略了CreateFile、WriteFile等与操作系统交互的API错误代码检查。

如何发现这类问题?

使用工具Canned Heat，模拟文件系统负载。

方法17：使介质忙或不可用

案例演示

此类缺陷产生的原因

开发人员没有考虑介质忙或者不可用的情况， 未对此种情况做出处理。

如何发现这类问题

使用工具Canned Heat，模拟介质忙或不可用的情况。

方法18：介质损坏

案例分析

缺陷产生的原因

损坏的介质可能会是操作系统传回错误代码，这些错误代码没有在应用程序中编程处理。

操作系统不能检测出所有的这些错误。

如何发现这类问题

一般软件，不必考虑介质损坏问题。一般用在操作系统、设备驱动程序/控制器以及以安全为主的应用程序才会考虑此类测试。

例如测试实现RAID5技术的软件，则需要模拟一块硬盘坏了之后，换一个硬盘，数据是否可以恢复。

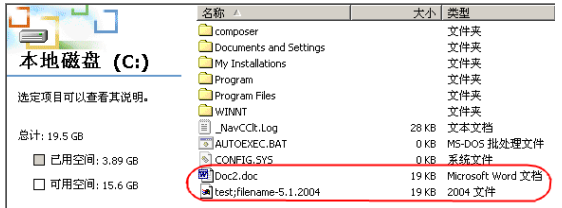
### 方法19：使用不合法的文件名

案例演示

– 环境：Win2000，Word2000

此类缺陷产生的原因

超 过 255 能被使用。



测试方法小结

熟记文件名命名规则

方法20：更改文件访问权限

案例演示

此类缺陷产生的原因

特别需要注意：不同的用户对相同文件具有不同的访问权限，需要考虑登录同一台机器的多个用户操作相同文件的权限问题。

如何发现这类问题?

方法21：文件内容受损

案例演示

缺

陷产生的原因

– 开发人员没有验证文件的格式和不通过的文件没有做出正确处理

### 文件系统的测试小结

使文件系统超载

使介质忙或不可用

介质损坏

使用不合法的文件名

更改文件访问权限

文件内容受损总结

二十种故障模型的来源

理解二十一种故障模型

请预习第二十一章

# **第二十二章**

软件测试执行

### 上一章回顾

二十一种故障模型的概念

二十一种故障模型

### 课堂提问

二十一种故障模型的概念

二十一种故障模型具体包括哪些

### 本章学习目标

掌握如何组织执行软件测试

掌握记录缺陷的方法

### 本章学习方法

运用

### 本章主题

组织测试团队

测试执行过程和方法

记录缺陷

### 组织测试团队

小型软件公司

开发团队在5-10左右，

没有专门的测试部门，

测试工作直接受项目经理和开发经理的直接领导，

优点：是能够节省人员的开销，测试活动周转和反馈较快。

缺点：测试工作不独立，不能全面的发现软件的缺陷，在保证软件质量上有一定限制。

### 中大型软件公司

有独立的测试组织结构

项目管理、测试管理和开发管理处于平等的地位，直接受控于技术总监

优点：能够保证测试的工作的独立开展，测试工作不会因为其他部门的影响还改变制定的测试规范

缺点：会给测试人员增加责任，尤其是测试部门经理的对整个测试结果和测试过程负责。信息流转相对较慢，需要人员开销大。

### 执行测试过程

按照测试计划由测试团队执行测试用例，一个优秀的测试用例处理能证明系统的功能和业务的正确性、合理性外， 还能够证明系统能够处理功能、业务等的异常情况。

执行测试时处理要做到测试用例和需求的覆盖外，还要有临时发挥的能力。根据自己的经验、对测试的感悟以及随机测试可以发现很多根据测试用例无法发现的缺陷。

不能拘泥于测试用例或者已经有的测试方法，在测试执行过程中要不断总结测试方法和测试故障模型。真正优秀的测试人员在执行测试时是想着做，做着想，这样的测试效果才好，尤其是在测试过程中，对程序的处理相当了解的情况下，测试的思路会更加清晰和全面。

### 缺陷报告

缺陷报告是描述软件缺陷现象和重现步骤地集合。

软件缺陷报告Software Bug Report （SBR） 或软件问题报告Software Problem Report

（SPR）

### 缺陷报告的作用

缺陷报告是软件测试人员的工作成果之一，体现软件测试的价值

缺陷报告可以把软件存在的缺陷准确的描述出来，便于开发人员修正

缺陷报告可以反映项目/产品当前的质量状态， 便于项目整体进度和质量控制

软件测试缺陷报告是软件测试的输出成果之一， 可以衡量测试人员的工作能力

#### 软件测试缺陷报告的“5C”原则

内容准确(Correct)

每个组成部分的描述准确，不会引起误解

步骤简洁(Concise)

只包含必不可少的信息，不包括任何多余的内容

内容清晰(Clear)

每个组成部分的描述清晰，易于理解

结构完整(Complete)

包含重现该缺陷的完整步骤和其他本质信息

风格一致(Consistent)

按照一致的格式书写全部缺陷报告

### 缺陷报告的内容

缺陷的标题；

缺陷的基本信息；

测试的软件和硬件环境；

测试的软件版本；

缺陷的类型；

缺陷的严重程度；

缺陷的处理优先级。

复现缺陷的操作步骤；

缺陷的实际结果描述；

期望的正确结果描述；

注释文字和截取的缺陷图像

### 缺陷的二八定理

在分析、设计、实现阶段的复审和测试工作能够发现和避免80%的缺陷，而系统测试又能找出其余缺陷中的80%，最后的4% 的缺陷可能只有在用户大范围、长时间使用后才会暴露出来。

### 记录缺陷与缺陷报告

使用较少的、必要的操作步骤确保缺陷能够重现

记录缺陷时要使用专业术语、注意书写格式

缺陷要言简意骇、尽量一个缺陷一个报告

对于实在不可重新的缺陷也需要报告 并且尽快报告

不能夸大缺陷的数量和缺陷的级别

及时记录缺陷

### 缺陷的分类

按照严重程度分类、缺陷的类型以及缺陷的优先级等进行分类

### 缺陷报告案例

演示如何编写缺陷报告

### 缺陷的分类

按照严重程度分类、缺陷的类型、缺陷的来源、功能以及缺陷的优先级等进行分类

### 按照严重程度

致命错误：如数据丢失、死机、系统崩溃

严重错误：如功能未完成，功能完成不正确

一般错误：如功能不完善，界面问题等

建议：测试人员认为怎么处理更好一些的问题。

### 按照缺陷类型

功能

压力/负载

界面

兼容

易用

安装/卸载

安全

### 按照修改优先级

立即修改

在本版本中修改

在产品发布前修改

在发布版本中可以存在的问题

思考：

修改优先级和缺陷严重程度的关系

严重程度高的修改优先级一定高？

一般情况下严重程度高的修改优先级就高，但不是绝对的。

### 按照功能模块

功能模块1

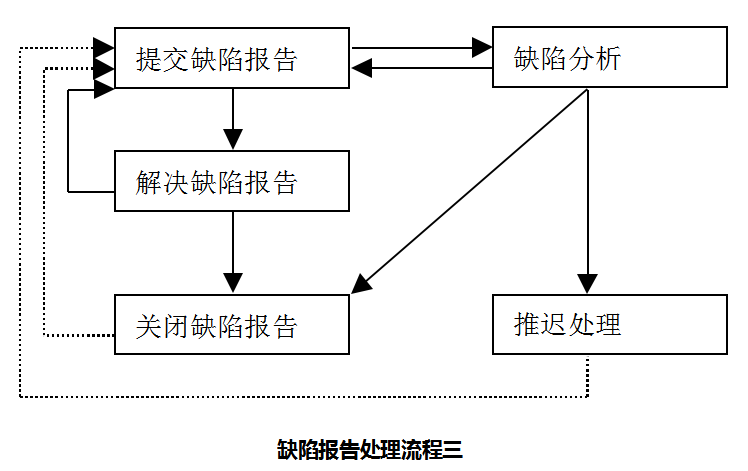
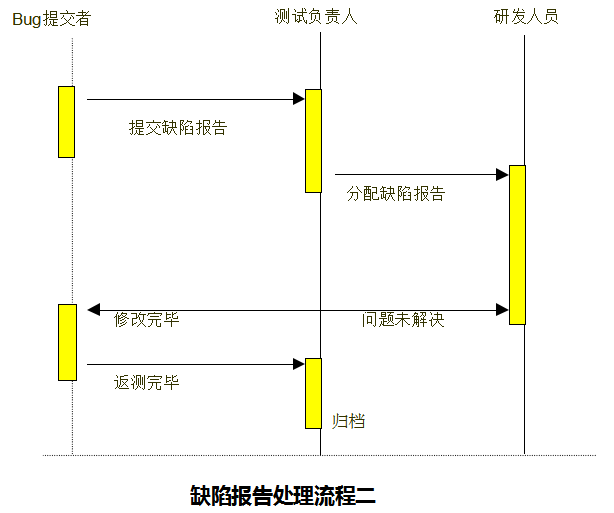
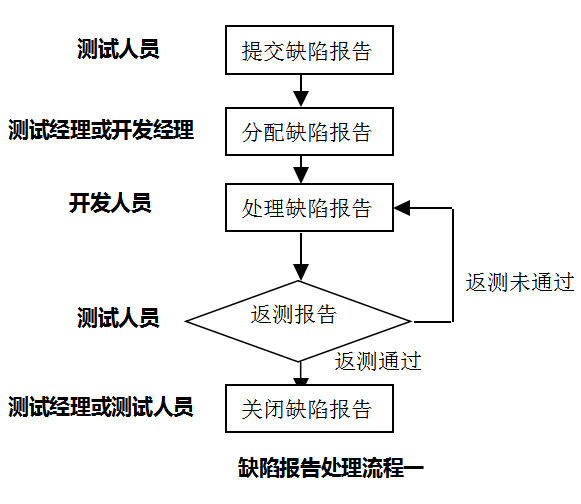
功能模块2

功能模块3

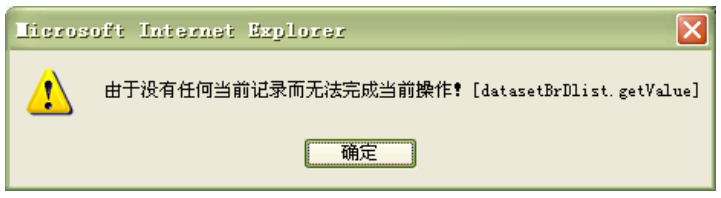
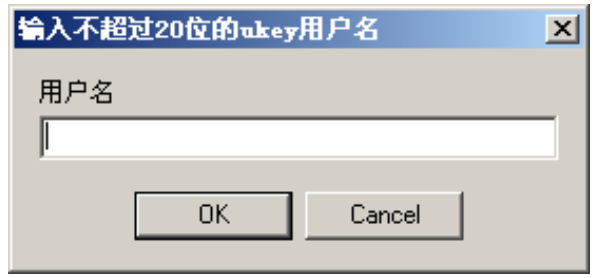
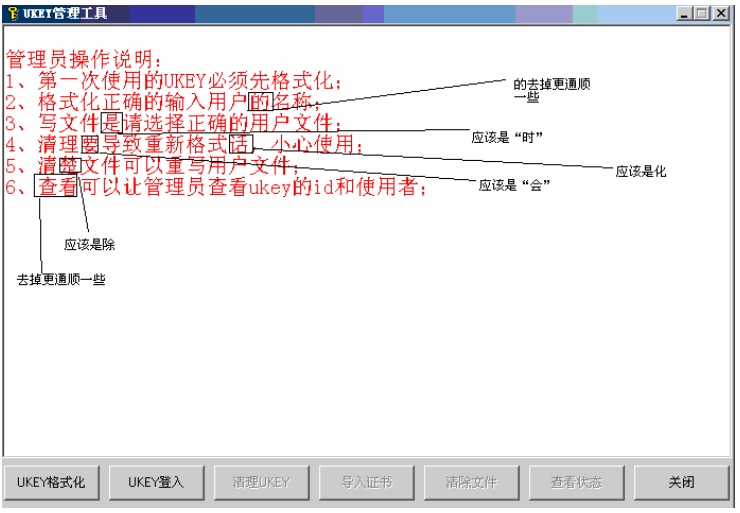
功能模块4

– ……

### 缺陷报告的处理流程



缺陷严重程度、是否记录练习



总结

组织测试团队

测试执行过程和方法

记录缺陷

请预习第二十三章

### 作业

课堂上机

将所负责的项目执行测试，并记录缺陷

课外作业

# **第二十三章**

软件测试总结

### 上一章回顾

组织测试团队

测试执行过程和方法

记录缺陷

### 提问

组织测试团队如何构成

测试执行过程和方法有哪些

记录缺陷的注意事项

### 本章学习目标

软件质量评估

### 本章学习方法

复习

记忆

### 软件质量评估

评估的方法

覆盖评测

基于需求的测试覆盖 (用例覆盖)

测试覆盖（已执行的）＝Tx/RfT

成功的测试覆盖（已执行的）＝Ts/RfT

基于代码的测试覆盖测试覆盖＝Tc/Tiic

软件质量评估

评估的方法

质量评测

缺陷报告

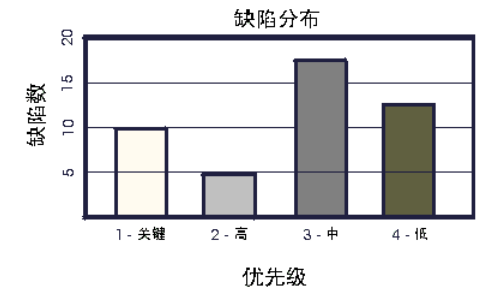
缺陷分布（密度）报告

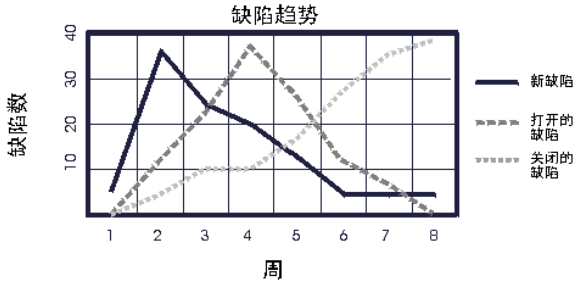
» 缺陷状态与优先级

» 缺陷状态与严重性

– 缺陷龄期报告

– 缺陷趋势报告





软件质量评估

### 评估的方法

质量评测

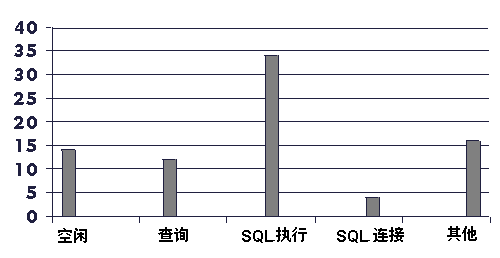
性能评测

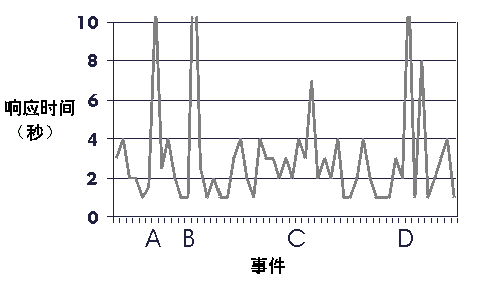
–动态监测

–响应时间/吞吐量

–百分位报告

–比较报告

–追踪报告



### 质内容进度

软件质量评估

测试总结报告

测试的文档

组织测试

测试总结报告

测试总结

测试总结的作用

测试总结的模板

测试总结报告

测试总结案例分析

概述

测试情况

测试统计

测试评价

测试总结报告小结

测试总结的作用

测试总结的格式

测试总结案例分析

内容进度

### 软件质量评估

测试总结报告

测试的文档

组织测试

测试的文档

文档的审核

需求说明书采用静态黑盒的审核方法

从客户的角度

系统是否运用了正确的标准，软件是否和现有的标准和规

范抵触

高质量的需求说明书的特点

完整

准确

精确，清晰

一致

贴切

合理

可测试

### 测试的文档

–高质量的需求说明书用语应避免使用

•总是，所有，每一种，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，以此类推

•良好，迅速，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）

### 文档的审核测试的文档

测试文档的管理和维护

放入项目的配置管理库管理，经过评审的文档进入受控库

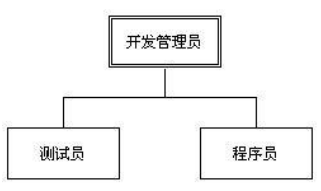
设置一位文档保管人员，负责保管项目的所有文档

测试小组的成员及时更新在配置管理库中的文档

测试的文档小结

文档的审核

测试文档的管理和维护



### 内容进度

软件质量评估

测试总结报告

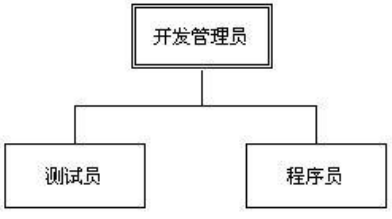
测试的文档

组织测试

组织测试

组织测试团队

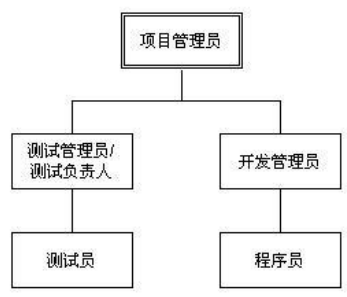
– 小型（少于10人）开发小组常用的组织结构



### 组织测试

组织测试团队

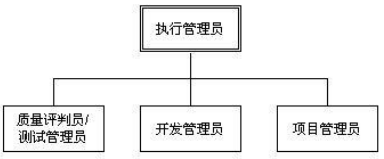
测试团队和开发团队都向项目管理员报告的组织结构



组织测试

组织测试团队

– 真正独立的测试组织结构



组织测试

激励机制

支持组员

工作时间

培训机会

构建学习型组织

在与其他公司合作中学习

向客户，同事学习

从自己过去的经验中学习

组织测试小结

组织测试团队

激励机制

构建学习型组织

本章内容总结

了解测试评估

了解测试总结报告

了解测试过程中文档的审核和管理

了解怎样组织测试

预习第二十四章

# **第二十四章**

软件测试过程改进

### 上一章回顾

软件测试的总结

软件测试的组织

### 本章学习目标

能够在企业中进行测试过程改进

### 本章学习方法

多讨论软件测试技术

### 本章主题

软件测试与软件质量

软件测试过程等级

软件测试过程改进

软件企业良好的软件测试过程

### 软件测试与软件质量

软件测试是提高软件质量的有效手段

良好的软件过程保证软件质量

软件测试属于软件过程的一个部分

#### 软件测试过程等级

(TMM/TCMM)

TCMM Level 1：Initial（初始级）

测试处于一个混乱的状态，测试与调试还没有分开，在编码完成后才进行测试工作，测试和调试交叉在一起，目的就是发现软件中的bug。 软件产品发布后没有质量保证。缺乏测试相应的测试资源、例如专职测试人员和测试工具， 测试人员没有经过培训。这种类型的公司属于这个阶段，处于这个阶段的公司在测试中缺乏成熟的测试目标，测试处于可无可有的地位。

### 软件测试过程等级

TCMM Level 2：Phase Definition（阶段定义级） 测试同调试分开且把测试作做为编码后的一个 阶段。尽管测试被看做是一个有计划的行为，

但是由于测试的不成熟仅在编码后制定测试计

划，因为测试完全是针对于源代码的。处于这个级别的公司测试的首要目的就是验证软件符合需求，会采用基本的测试技术和方法，由于测试处于软件生命周期的末尾环节，导致出现很多无法弥补的质量问题。另外，在需求和设计阶段产生的很多问题被引入到编码中，但基于源代码的测试导致产生了很多的问题无法解决。

### 软件过程等级

TCMM Level 3：Integration（集成级）

测试不再是编码后的一个阶段，而是把测试贯穿在整个软 件生命周期中。在需求阶段软件测试就介入了，测试是建 立在满足用户或客户的需求上，根据需求设计测试用例和 作为测试的依据。处于这个级别的公司测试工作由具有独 立的部门负责，测试部门与开发部门分开，独立开展工作。测试部门有自己的技术培训并且有测试工具辅助进行测试 工作。尽管处于这个阶段的公司认识到了评审在质量控制 中的重要性，但是并没有建立起有效的评审制度，还不能 在软件生命周期的各个阶段实施评审制度。没有建立起质 量控制和质量度量标准。

### 软件过程等级

TCMM Level 4：Management and Measurement（管理和度量级）

测试是一个度量和质量控制过程。在软件生命周期中评审作为测试和软件质量控制的一部分， 被测试的软件产品标准包括可靠性、可用性和可维护性等。在测试项目中设计的测试用例被 保存在测试用例数据库中便于重用和回归测试。 使用缺陷管理系统管理软件缺陷并划分缺陷的级别。但是处于这个阶段的公司还没有建立起缺陷预防机制，且缺乏自动地对测试中产生的数据进行收集和分析的手段。

### 软件过程等级

TCMM Level 5：Optimization（优化级）

具有缺陷预防和质量控制的能力。建立TCMM4基础上的测试公司已经建立起测试规范和流程，测试是受控的和被管理的。而达到TCMM5的公司，则坚决贯彻落实测试规范和流程且不断地进行测试过程改进，在实践中运用缺陷预防和质量控制措施。整个测试过程是被以往经验所驱动的，且是可信任和可靠的。选择和评估测试工具存在一个既定的流程。测试工具支持测试用例的运行和管理，辅助设计用例和维护测试相关资料，缺陷收集和分析，为缺陷预防和质量控制提供支持。

### 软件测试过程改进

调整测试活动的时序关系

优化测试活动资源配置

提高测试计划的指导性

确立合理的度量模型和标准

提高覆盖率

减少漏测

#### 软件企业良好的软件测试过程

测试流程与测试规范

测试尽早介入

自动化测试流程引入

质量控制机制

提高测试效率

引入白盒测试

测试数据记录与度量

### 总结

软件测试与软件质量

软件测试过程等级

软件测试过程改进

预习第二十五章

# **第二十五章 软件风险分析**

## **软件风险管理的概念**

软件项目风险是指在软件开发过程中遇到的预算和进度等方面的问题以及这些问题对软件项目的影响

软件风险管理：试图以一种可行的原则和实践，规范化地控制影响项目成功的风险

## **风险管理的重要性**

有效的风险管理增加项目成功的机会，减少项目失败的概率

风险管理可以增加团队的稳固性

风险管理可以帮助项目经理抓住工作重点， 将主要精力集中于重大风险，将工作方式 从被动救火转变为主动防范。

## **软件项目中的风险**

软件项目的风险主要来源于需求、技术、成本和进度。

### **需求风险**

已经纳入基线的需求在继续变更

需求定义不准确,进一步的定义会扩展项目范畴

增加额外的需求

产品定义含混的部分比预期需要更多的时间

在做需求中客户参与不够

缺少有效的需求变化管理过程

### **计划编制风险**

计划、资源和产品定义全凭客户或上层领导口头指令,并且不完全一致

计划是优化的,是"最佳状态",但计划不现实,只能算是"期望状态";

计划基于使用特定的小组成员,而那个特定的小组成员其实指望不上;

产品规模(代码行数、功能点、与前一产品规模的百分比) 比估计的要大;

完成目标日期提前,但没有相应地调整产品范围或可用资源;

涉足不熟悉的产品领域,花费在设计和实现上的时间比预期的要多。

### **组织和管理风险**

仅由管理层或市场人员进行技术决策,导致计划进度缓慢,计划时间延长;

低效的项目组结构降低生产率;

管理层审查决策的周期比预期的时间长;

预算削减,打乱项目计划;

管理层作出了打击项目组织积极性的决定;

缺乏必要的规范,导至工作失误与重复工作;

非技术的第三方的工作(预算批准、设备采购批准、法律方面的审查、安全保证等)时间比预期的延长。

### **人员风险**

作为先决条件的任务(如培训及其他项目)不能按时完成;

开发人员和管理层之间关系不佳,导致决策缓慢,影响全局;

缺乏激励措施,士气低下,降低了生产能力;

某些人员需要更多的时间适应还不熟悉的软件工具和环境;

项目后期加入新的开发人员,需进行培训并逐渐与现有成员沟通,从而使现有成员的工作效率降低;

由于项目组成员之间发生冲突,导致沟通不畅、设计欠佳、接口出现错误和额外的重复工作;

不适应工作的成员没有调离项目组,影响了项目组其他成员的积极性;

没有找到项目急需的具有特定技能的人。

### **开发环境风险**

设施未及时到位;

设施虽到位,但不配套,如没有电话、网线、办公用品等;

设施拥挤、杂乱或者破损;

开发工具未及时到位;

开发工具不如期望的那样有效,开发人员需要时间创建工作环境或者切换新的工具;

新的开发工具的学习期比预期的长,内容繁多。

### **客户风险**

客户对于最后交付的产品不满意,要求重新设计和重做;

客户的意见未被采纳,造成产品最终无法满足用户要求,因而必须重做;

客户对规划、原型和规格的审核决策周期比预期的要长;

客户没有或不能参与规划、原型和规格阶段的审核,导致需求不稳定和产品生产周期的变更;

客户答复的时间(如回答或澄清与需求相关问题的时间)比预期长;

客户提供的组件质量欠佳,导致额外的测试、设计和集成工作,以及额外的客户关系管理工作。

### **产品风险**

矫正质量低下的不可接受的产品,需要比预期更多的测试、设计和实现工作;

开发额外的不需要的功能(镀金),延长了计划进度;

严格要求与现有系统兼容,需要进行比预期更多的测试、设计和实现工作;

要求与其他系统或不受本项目组控制的系统相连,导致无法预料的设计、实现和测试工作;

在不熟悉或未经检验的软件和硬件环境中运行所产生的未预料到的问题;

开发一种全新的模块将比预期花费更长的时间;

依赖正在开发中的技术将延长计划进度。

### **设计和实现风险**

设计质量低下,导致重复设计;

一些必要的功能无法使用现有的代码和库实现, 开发人员必须使用新的库或者自行开发新的功能;

代码和库质量低下,导致需要进行额外的测试, 修正错误,或重新制作;

过高估计了增强型工具对计划进度的节省量;

分别开发的模块无法有效集成,需要重新设计或制作。

### **过程风险**

大量的纸面工作导致进程比预期的慢;

前期的质量保证行为不真实,导致后期的重复工作;

太不正规(缺乏对软件开发策略和标准的遵循), 导致沟通不足,质量欠佳,甚至需重新开发;

过于正规(教条地坚持软件开发策略和标准),导致过多耗时于无用的工作;

向管理层撰写进程报告占用开发人员的时间比预期的多;

风险管理粗心,导致未能发现重大的项目风险。

## **风险管理过程**

### **风险识别**

**风险识别的内容：**确定风险的来源；风险产生的条件；描述风险特征；影响本项目的风险。

风险识别的方法

**头脑风暴法(Brain-Storming)：**无限制的自由联想和讨论的代名词，其目的在于产生新观念或激发创新设想。最早是精神病理学上的用语，指精神病患者的精神错乱状态而言的。

头脑风暴法是由美国创造学家A·F·奥斯本于1939年首次提出、1953年正式发表的一种激发性思维的方法。

头脑风暴法有可分为直接头脑风暴法（通常简称为头脑风暴法）和质疑头脑风暴法（也称反头脑风暴法）。前者是在专家群体决策尽可能激发创造性，产生尽可能多的设想的方法，后者则是对前者提出的设想、方案逐一质疑，分析其现实可行性的方法。

**面谈**

**Delphi法：**系以一系列问卷向各类专家征询意见，依据所有专家对原问卷的答复再拟定下一份问卷，再次向各类专家征询意见，直到大多数专家的意见看法趋于一致才获至结论，此法预测通常较适合作长期预测。

**Delphi步骤**

成立一个团体委员确定问题及设计研究问卷。

选择专家匿名质问预测。

把质问的问题回答收回做成结果。

反复再预测，把问题重新修整再做预测， 如此反复预测至少2次以上

**核对表：**核对表是基于以前类比项目信息及其他相关信息编制的风险识别核对图表。核对表一般按照风险来源排列。利用核对表进行风险识别的主要优点是快而简单，缺点是受到项目可比性的限制

**SWOT技术：**主要是用来分析组织内部的优势与劣势以及外部环境的机会与威胁，在市场营销管理中经常使用的功能强大的分析工作。

SWOT 分析法的基本分析步骤

分析环境因素

构造SWOT 矩阵

制定行动计划

### **风险量化**

风险量化的内容：确定哪些事件需要制定应对措施

风险量化的方法：因子计算，PERT估计，决策树分析，风险模拟，

因子计算：因子量化，

PERT估计（Project Evaluation and Review Technique）

PERT对各个项目活动的完成时间按三种不同情况估计：一个产品的期望规模，一个最低可能估计，一个最高可能估计。用这三个估计用来得到一个产品期望规模和标准偏差的Pert 统计估计。Pert 估计可得到代码行的期望值E， 和标准偏差SD

### **风险应对计划制定**

制定风险应对策略主要考虑以下四个方面的因素：

可规避性:即通过变更项目计划消除风险或风险的触发条件， 使目标免受影响。

可转移性:即不消除风险，而是将项目风险的结果连同应对的权力转移给第三方(第三方应该知道这是风险并有承受能力)。

可缓解性:即将风险事件的概率或结果降低到一个可以接受的程度。

可接受性:即不改变项目计划(或没有合适的策略应付风险)， 而考虑发生后如何应对。

### **风险监控**

风险监控的方法：核对表；定期项目评估；挣值分析

挣值分析法又称偏差分析法，是一种分析目标实施与目标期望之间差异的方法。挣值法的优点是能同时判断项目预算和进度计划的执行情况，以预算和费用来衡量工程的进度

**经典风险管理理论**

### **Boehm模型**

两步骤：风险评估和风险控制

核心是维护和更新十大风险列表

包括人员短缺、不切实际的工期和预算、不合时宜的需求、开发了错误的软件功能、开发了错误的用户界面、过高的非实质性能要求、接连不断的需求改变、可外购部件不足、外部已完成任务不及时、实时性能过低和计算机能力有限

在软件项目开始时归纳出现在项目的十大风险列表，

在项目的生命周期中定期召开会议去对列表进行更新、评比。

十大风险列表是让高层经理的注意力集中在项目关键成功因素上的有效途径，可以有效地管理风险并由此减少高层的时间和精力

**CRM模型（持续风险管理模型）**

要不断地评估可能造成恶劣后果的因素

决定最迫切需要处理的风险

实现控制风险的策略

评测并确保风险策略实施的有效性

CRM模型要求在项目生命期的所有阶段都关注风险识别和管理，五个步骤：风险识别、分析、计划、跟踪、控制。强调对风险管理的各个组成部分的沟通

### **Leavitt模型**

将形成各种系统的组织划分为4个组成部分：任务、结构、角色和技术。

4个组成部分和软件开发的各因素很好地对应起来： 角色覆盖了所有的项目参与者，例如软件用户、

项目经理和设计人员等；结构表示项目组织和其他制度上的安排；技术则包括开发工具、方法、硬件软件平台；任务描述了项目的目标和预期结果。

Leavitt模型的关键思路是：模型的各个组成部分是密切相关的，一个组成部分的变化会影响其他的组成部分，如果一个组成部分的状态和其他的状态不一致，就会造成比较严重的后果，并可能降低整个系统的性能。