测试需求分析与测试用例设计

一、界面中的控件知识

1文本框和密码框





文本框





长度要求；

输入内容限制。

密码框











长度要求；

不允许明文显示；

禁止复制粘贴；

输入内容限制；

两次密码要一致。

2单选按钮、组合列表框、数码框



单选按钮













框架标题/提示文本不缺失且正确；

各个选项正确；

执行同一功能的多个单选按钮只能选中一个；

要有默认选中项；

一般不能取消选中；

存入后台的数据正确。





组合列表框/下拉列表









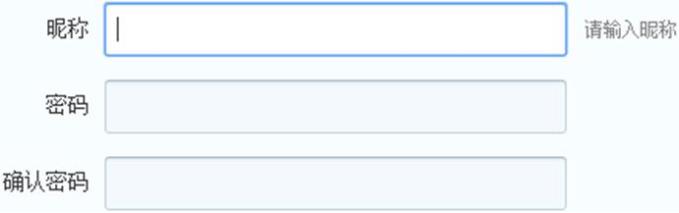
通常单选，条目内容要正确（没有多余/错放项、缺少项）；

横向显示要完整；

条目功能要正确实现；

组合列表框中可能允许输入数据。

数码框（up-down控件）











能使用上下箭头控制数字变动；

数字有范围限制；

数字自动循环或者到达边界时停止；

可以直接输入数字。

3复选框



复选框







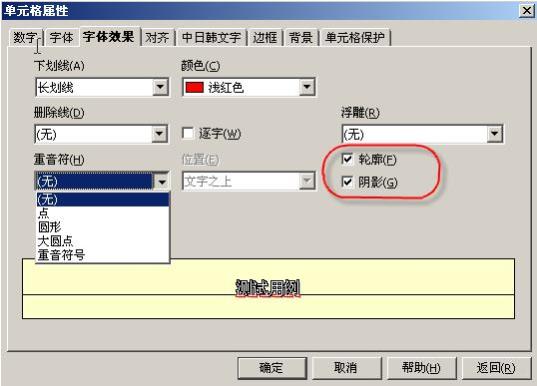
选项正确；

可以不选、任意选一个、任意选多个、全选；

可以取消选中；



每一个复选框功能都正确实现。



4列表框









通常多选；

条目内容要正确（多余/错放项、缺少项）；

横向显示完整，纵向显示完整；

条目功能要正确实现。

5命令按钮





实现所需的功能；

出现错误时，给出切当明确的提示信息。

6其他界面元素





窗口标题





不缺失；

显示正确。

选项卡



ctrl+tab切换







默认焦点

Tab顺序

快捷键/热键





使用 ctrl或 alt+其它字母

同一窗口、界面或菜单中不能重复



二、大纲法分解功能

1大纲法

大纲法主要用于对软件进行功能拆分。













模块



包含多个功能操作的对象或功能集合，如文件（菜单）等。

功能点/功能



能独立完成一件事或一个业务。如新建、打开等。

业务流程



软件为了完成业务或完成核心功能所经历的步骤。

业务逻辑



是对业务的不同处理方式。

业务规则



如要求用户名只能用英文，5-11个字符等。

案例



即时贴程序部分需求说明









便签的数量最多为 50个

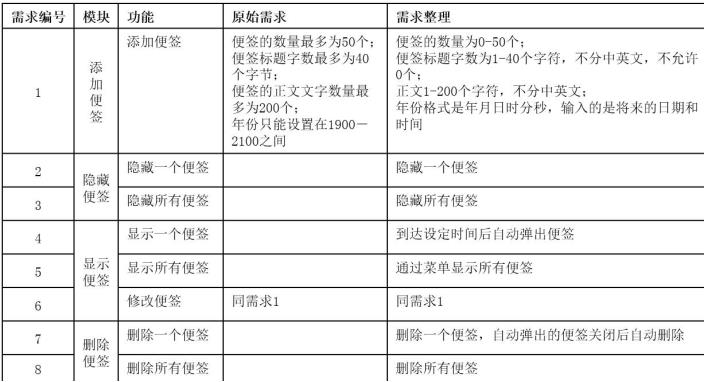
便签标题字数最多为 40个字节

便签的正文文字数量最多为 200个

年份只能设置在 1900－2100之间

2开始编写测试需求分析

将功能拆分与整理的需求信息写入测试需求分析



三、测试需求分析与测试用例设计方法

1场景法

1.1测试点/检查点



测试时应该考虑可以测试的诸多方面。

1.2场景法概述





场景法模拟用户操作软件时的情景，主要用于测试系统的业务流程。

当拿到一个测试任务时，我们先要关注它的主要功能和业务流程是否正确实现，这

就需要使用场景法来完成测试。

1.3场景的定义





场景用来描述软件操作的路径。

基本流



按照正确的业务流程来实现的一条操作路径（模拟正确的操作流程）。



备选流



导致程序出现错误的操作流程（模拟错误的操作流程）。

1.4场景法的分析步骤







分析软件需求

从用户使用情景角度，写出业务流程和业务规则

写出基本流场景和备选流场景

1.5场景法案例：ATM机取款



步骤一：分析业务流程（可以绘制流程图）





步骤二：描述程序的基本流及备选流



1、基本流（正确的流程）





（1）插入银行卡：客户将银行卡插入 ATM机的读卡器

（2）验证银行卡：ATM机从银行卡的磁条中读取账户代码，并检查它是

否属于可以接受的银行卡











（3）输入密码：ATM机要求客户输入密码

（4）验证密码：确定该密码是否正确

（5）进入 ATM主界面：ATM显示在本机中可用的各种选项

（6）选择取款并输入金额：客户选择“取款”，并选择取款金额

（7）ATM机验证：ATM机进行验证账户余额是否满足以及总取款金额

是否满足要求，验证 ATM机内现金是否够用

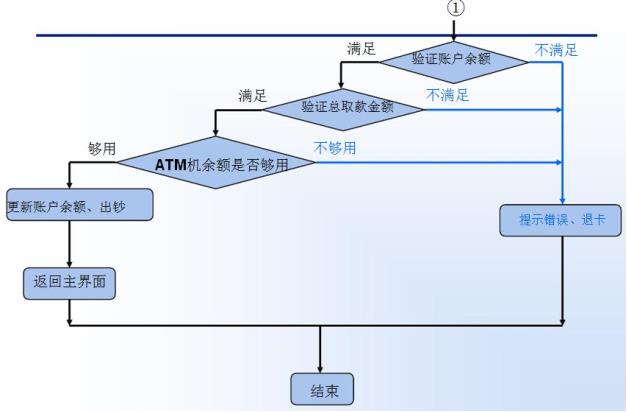
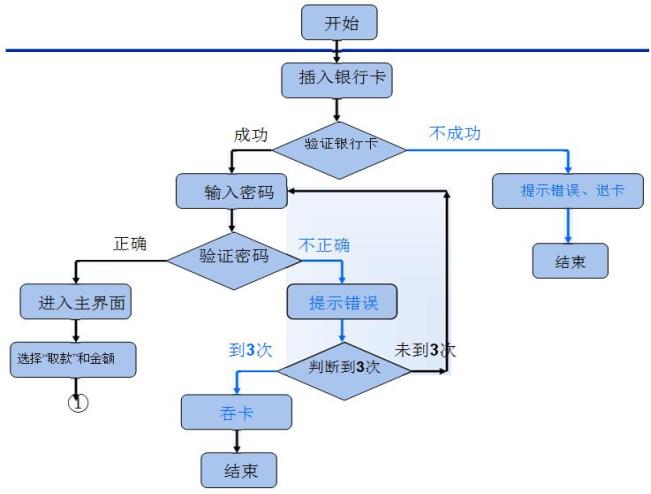
（8）更新账户余额、出钞：验证成功，更新账户余额，输出现金，提示

用户收取现金





（9）返回主界面





2、备选流（各种错误情况）













（1）银行卡无效：提示错误并退卡

（2）密码错误：提示错误，并判断是否 3次错误

（3）密码 3次错误：吞卡

（4）账户余额不足：提示错误并退卡

（5）总取款金额超出当日可取限额：提示错误并退卡

（6）ATM机余额不足：提示错误并退卡



步骤三：根据基本流和备选流生成不同的场景

1.6场景法练习

2等价类划分

2.1案例引入

测试两位数加法器（学生思考、讨论、陈述）



2.2等价类划分核心思想







通过需求分析，找出程序的输入域。

将输入域划分成若干类。

每一类中选取代表性数据等价于这一类中的其他值。

2.3等价类划分的步骤





需求分析

划分等价类（根据需求，有效等价类、无效等价类）并细化（根据计算机知识）

2.4等价类划分案例



步骤 1：需求分析









阅读文档

借助开发知识

与开发或用户沟通

了解用户群及行业知识



写出需求：



-99~99之间的整数



步骤 2：划分等价类并细化



有效类



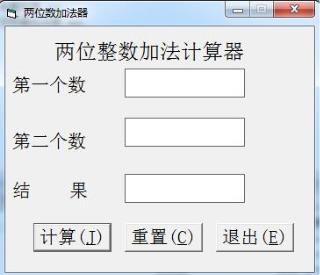


-99到 99之中的整数

细化



负数







0

正数



无效类



超范围





<-99

>99



非法类型





浮点数

字符（串）

2.5等价类划分练习

ATM机的取款测试：

取款最少 50，最多 5000，每笔 50的倍数，测试取一笔。

2.6等价类划分注意事项





不但要考虑有效等价类，也要考虑无效等价类

两块划成一块（等价类划分过粗），结果？



遗漏一种测试情况





一块划成两块（等价类划分过细），结果？



冗余测试

仔细划分，审查划分





过于粗略可能会漏掉软件缺陷

积累经验

2.7思考题







密码框的输入范围 4-10位

程序要求输入 BOOLEAN型数据

Microsoft

Office中，选择字体的组合框



要求用户名由字母或数字组成，必须由字母开头，不包含特殊字符，最大长度为

12



3边界值分析

3.1一个缺陷



缺陷原因

3.2边界值分析的思想与步骤





分析需求，找出边界。

写出边界值









最小值

小于最小值

最大值

大于最大值

3.3边界值分析案例



两位数加法计算器的边界值









-99

-100

99

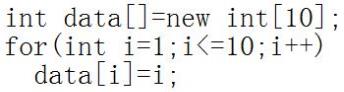
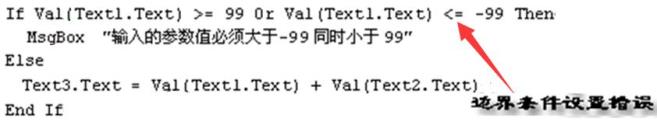
100

3.4为什么分析边界值

看看下面的代码有错误吗？



输入或输出的边界最容易产生错误。



3.5边界值分析练习

使用边界值分析方法写测试点



练习题讲解



测试知识的储备



了解开发原理



可能的编码方式







if（bl >= 'A'

if（bl >=65

if（bl > 64

&&

&&

bl <= 'Z'）

bl <=90）

bl < 91）

&&





if（bl>='0'

if（bl>=0

&&

bl<='9'）

bl<=9）

&&



if（bl>= -32768

if（bl >= -99

&&

bl <= 32767）



&&

bl<= 99）



部分 ASCII

3.6边界值分析思考题



即时贴程序，考虑标题的测试



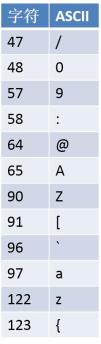
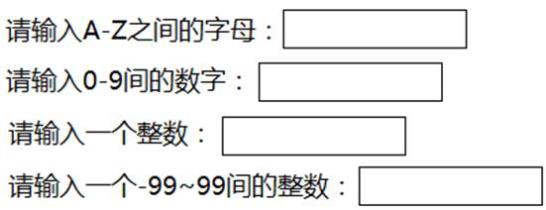


标题字数：1-40字节

标题字数：0-40字节



文本输入域允许输入 1-255个字符







下拉列表

分页

4决策表

4.1前面方法忽略的问题

测试两位数加法计算器的测试没有考虑输入组合。

4.2决策表的分析步骤



需求分析



分析输入和输出



用等价类划分分析输入的各种情况、输出的各种情况





画判定表

分析与简化判定表

4.3决策表案例



分析输入条件和输出条件



输入



输入 1：









条件 1： 0<=X<=99

条件 2： -99<=X<0

条件 3： X<-99

条件 4： X>99



输入 2：









条件 1： 0<=X<=99

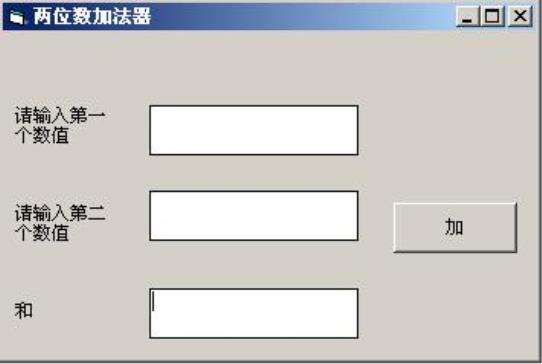
条件 2： -99<=X<0

条件 3： X<-99

条件 4： X>99



输出







正确计算

错误提示



原始决策表/判定表



优化决策表



优化策略







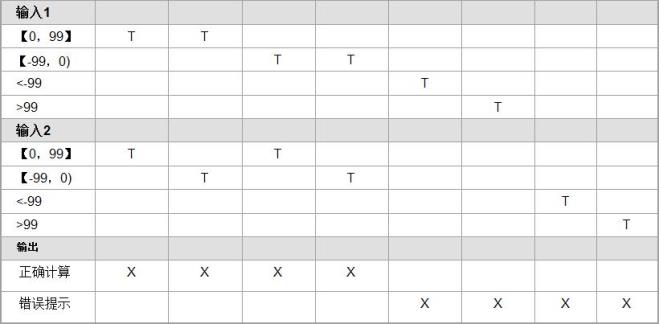
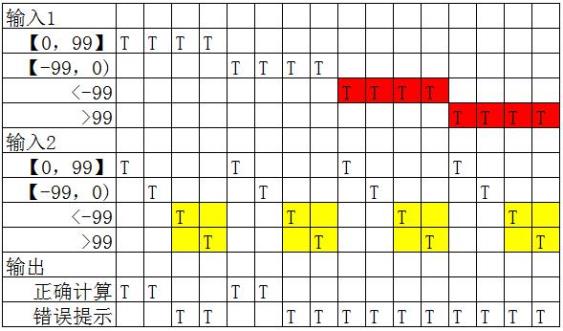
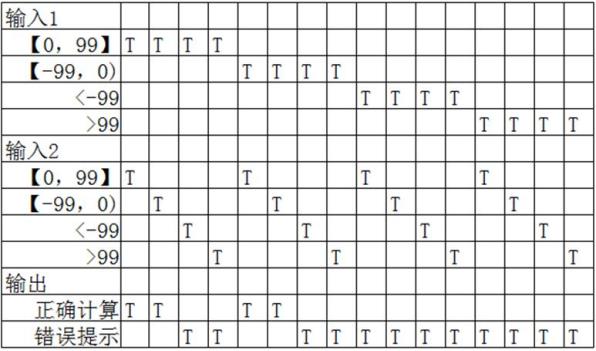
测试基本功能的保留；

一个输入错误，另外输入无所谓，可以整合；

所有输入都要错误过。



最终的决策表



4.4决策表练习



案例：某厂工资发放需求









工资分为年薪制，月薪制，两者互斥

错误程度分为普通和严重，两者可同时具备

年薪制员工犯普通错误扣工资 2％，犯严重错误扣工资 6％

月薪制员工犯普通错误扣工资 4％，犯严重错误扣工资 8％

4.5决策表的适用范围

适用于多种输入的存在组合情况时。

4.6决策表的局限性与优化策略

导致测试量爆炸。



决策表优化策略

音形码拼音码路名码行业类别

特征码

填

填

填

填

填

填

填

填

填

不填

填

填

填

填

不填

填

填

填

不填

填

填

填

不填

填

填

填

不填

不填

填

填

填

不填

不填

不填

不填

5错误推测

5.1测试若干原则回顾









测试不是验证软件正确，而是攻击软件，发现错误。

测试要时刻保持怀疑的态度，具有缺陷预防意识。

测试要寻求系统设计、功能设计的弱点。

设计负面的、异常的测试，如考虑错误的或者异常的输入，往往可以发现更多的软



件缺陷。

5.2什么是错误推测

在测试程序时，人们可以根据经验或直觉推测程序中可能存在的各种错误，从而有针对

性地编写检查这些错误的测试方法。



错误推测分类









输入数据测试方面

输出数据测试方面

数据结构测试方面

文件系统方面

5.3输入数据方面的错误推测

5.3.1输入非法数据





一般用于键盘输入数据时。

测试方法



输入非法类型



输入非法范围/长度

输入非法格式







注意



错误信息的检查：需要额外考虑错误提示信息的内容

错误信息和错误要对应一致





错误信息不能为空

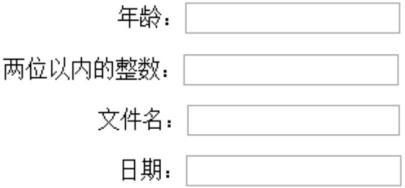


错误信息的内容不能只是错误代码，不能包含开发信息

错误信息不能中英文混合



案例



5.3.2输入默认值





适用于有默认值的地方。

测试方法









接受软件的默认值

键入空值

将默认值改为另外一个值

将默认值改为另外一个值，再变为空值

5.3.3输入特殊字符（集）





适用于不能输入有特殊含义的字符时。

测试方法

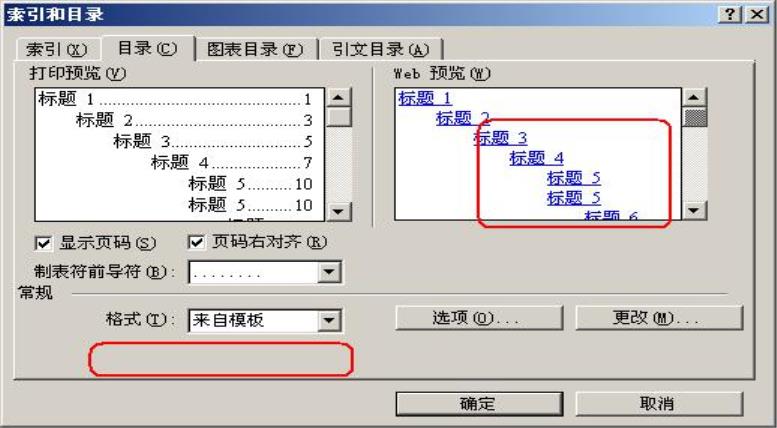


根据被测软件所处的操作系统、程序设计语言、后台数据库以及具体业务

等信息列出表格，进行讨论，标明哪些需要测试，哪些需要剔除。

了解具体行业知识，具体问题具体分析。









案例



文件命名下列特殊字符（33个）



不能使用：\ / < > | “ : \* ?，com0-com9，lpt0-lpt9，prn、aux、nul、

con。

思考



用户名有哪些特殊字符？



QQ昵称、聊天内容有哪些特殊字符？

5.3.4输入合法数据的非法组合





适用于输入值之间存在依赖关系时。

测试方法



输入可能是出现问题的组合值。



案例

5.3.5通过复制粘贴强制输入程序不允许输入的数据

5.4输出数据方面的错误推测

5.4.1同一个输入产生多种输出



案例



输入：一个电话打来

输出：















状态一：等待接听。

状态二：占线。

状态三：停机。

状态四：无法接通。

状态五：关机。

状态六：空号。





测试方法





详细测试每一种输出，不要有遗漏。

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，

列出关于程序输入于输出的一个列表，然后进行测试。

思考



QQ中有没有类似的测试？



5.4.2验证输出结果的正确性



测试方法





不仅测试输入的正确性，还要检查结果的正确性。

测试人员要尽可能多地学习所涉及问题的领域。

5.5数据结构方面的错误推测

5.5.1数据结构溢出





适用于程序中存在变量、数组等数据结构时。

测试方法



变量



上溢：值太大

下溢：值太小





数组



上溢：数据量太多

下溢：数据量太少





思考



QQ中有相关案例吗？

5.5.2计算结果溢出



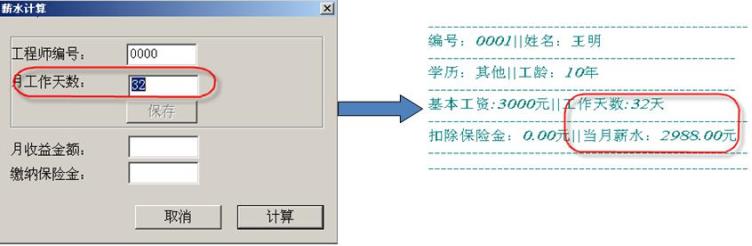
案例



测试方法



输入非法值或很大与很小数据，强制结果产生上溢或下溢。



5.5.3操作数和操作符不符



案例



是否是缺陷？



如果是缺陷，开发人员修改成什么样的结果，你作为测试人员会确认这个

缺陷已经被修复？





适用于需要进行数值计算程序和图形操作程序的测试时，如加、减、乘、除等。

测试方法



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。

5.6文件系统方面的错误推测

5.6.1使文件系统超载





适用于数据存储到硬盘中时。

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文件可能会有 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？



测试方法







创建满容量或近乎满容量的文件系统，然后强制执行各种通过输入或输出

访问文件系统的操作。

打开足够多的文件，文件打开时会强制创建备份副本，从而占用双倍的存

储空间。

使用工具 Canned Heat，模拟文件系统超载。

5.6.2更改文件访问权限





适用于对文件进行读写的应用程序。

测试方法



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。

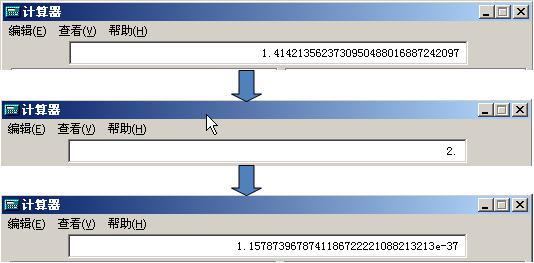


打开一个文件，在操作系统中修改该文件的访问权限。有些操作系统

允许权限高的用户控制一般用户已经打开的文件。



两个应用程序打开，关闭同一个文件。







如把同一应用程序的不同版本安装在同一机器上，在不同版本的应用

程序中打开和关闭同一文件；

试着在某个应用程序中打开在另一个程序中已打开的文件，这可能会

导致文件访问权限上出现冲突。

5.6.3使介质忙或不可用



适用于应用程序的运行需要消耗大量内存或运行时需求其他相关软件同时运

行的情况。



大多数操作系统能同时运行多个应用程序，但相互切换时会有延迟，但是

没有对错误响应。



测试方法



通过启动大量应用程序，强制它们都打开并保存文件来使文件系统处于忙

的状态；或者同时下载大量文件也可以使后台拥挤。

使用一些测试工具来模拟磁盘的状况。



5.6.4介质损坏



使用场合



损坏的介质可能使操作系统传回错误代码，这些错误代码可能没有在应用

程序中编程处理。



测试方法







损坏介质的方法使用不很多，只有少数公司采用，大多是开发操作系统、

设备驱动程序以及以安全为主的应用程序的公司会采用这种测试方法。确

定是否使用该方法，主要要考虑数据对用户的重要性。

该方法可以使用实际损坏了的介质。检查应用程序对错误的处理能力，应

用程序可以对错误进行处理或者将问题告诉用户，并且要确保用户数据文

件不丢失、不损坏。

也可以通过软件模拟。

5.7错误推测总结



























输入非法类型

输入非法范围（数值）

输入非法长度（个数）

输入非法格式

输入默认值

输入特殊字符

输入合法数据的非法组合

粘贴强制输入

一个输入多个输出不要遗漏

输出结果（含数据库）要正确

上溢、下溢（含结果）

操作数与操作符不符

文件超载









文件权限不足

介质忙或不可用

介质损坏

6编写测试点

将测试点写入测试需求分析说明书，或者 XMind等，留存下以供将来编写测试用例使

用。

四、需求评审

1意义







对软件需求进行正确性的检查。

保证软件需求的可测试性。

通过产品需求文档的评审，与市场、产品、开发等各部门相关人员沟通，使得大家

认识一致，避免在后期产生不同的理解，引起争吵。





通过产品需求文档的评审，更好地理解产品的功能性和非功能性需求

在需求文档评审通过后，测试的目标和范围就确定了。虽然此后会有需求的变更，

但可以得到有效的控制，这样可降低测试的风险。



评审是否完成是以需求文档获得多方“邮件确认”或“签字”通过为标志的。这不

应该只体现在“签字”形式上，更重要的是达到下面的结果。





所有参与方达成一致。

已发现的问题被阐述清楚、被修正。



2需求评审的质量要求

















正确性

完备性

易理解性

一致性

可行性

易修改性

可测试性

可追溯性

3需求评审的参加人员





















用户代表

需求人员

产品经理

项目经理

开发人员

开发经理

测试人员

测试经理

市场经理

质量保证人员

4测试人员参与评审时的注意事项



















明确自己的角色和责任。

熟悉评审内容，为评审做好准备，做细做到位。

在评审会上，针对问题阐述观点，而不是针对个人。

可以分别讨论主要的问题和次要的问题。

在会议前或会议后可以就存在的问题提出自己建设性的意见。

提高自己的沟通能力，采取适当的、灵活的表述方式。

对发现的问题跟踪下去。

应该在需求形成的过程中进行分阶段的多次评审，而不是一次评审。

测试人员要善于提问，多思考











这些需求都是用户提出来的吗？

有没有画蛇添足的需求？没有漏掉什么需求吗？

和竞争对手的产品做过比较吗？我们的产品优势体现在哪里？

是否正确地描述了每个需求？这条描述是否存在二义性的问题？

我的理解和文档作者的理解一致吗？

