中 华 人 民 共 和 国 国 家 标 准

GB/T 15532－200X

计算机软件测试规范

Specification of computer software testing

（征求意见稿）

200X－XX－XX 发布　　200X－XX－XX 实施

中 华 人 民 共 和 国
国家质量监督检验检疫总局　发布

目　　次

前　言

本标准的附录 A、附录 B、附录 C 和附录 D 是资料性附录。

本标准由中华人民共和国信息产业部提出。

本标准起草单位：信息产业部电子工业标准化研究所、总装备部测量通信总体研究所、上海计算机软件开发中心。

本标准主要起草人： 许聚常、冯惠、杨根兴、朱国庆、尹平、王欣、王宝艾、张旸旸。

计算机软件测试规范

1　范围

本标准规定了计算机软件生存周期内各类软件产品的基本测试方法、过程和准则。

本标准适用于计算机软件生存周期全过程。本标准适用于计算机软件的测试机构和测试人员。

2　规范性引用文件

下列文件中的有关条款通过引用而成为本标准的条款。凡注日期或版次的引用文件，其后的任何修改单（不包括勘误的内容）或修订版本都不适用于本标准，但提倡使用本标准的各方探讨使用其最新版本的可能性。凡不注日期或版次的引用文件，其最新版本适用于本标准。

GB/T 8566　信息技术　软件生存周期过程

GB/T 9386　计算机软件测试文档编制规范

GB/T 11457 软件工程术语

GB/T 16260.1-2006　软件工程　产品质量　第 1 部分：质量模型

GB/T 18234-2000　信息技术　CASE 工具的评价与选择指南

GB/T 18492　信息技术　系统及软件完整性级别

3　术语和定义

GB/T 11457 中确立的术语和定义适用于本标准。

4　总则

4.1　测试目的

计算机软件的测试目的是：

a）验证软件是否满足软件开发合同或项目开发计划、系统/子系统设计文档、软件需求规格说明和软件设计说明所规定的软件质量特性要求；

b）通过测试，发现软件错误；

c）为软件产品质量的评价提供依据。

4.2　测试级别

根据 GB/T 8566 的要求，本标准对如下测试级别作详细描述：

a）单元测试；

b）部件测试；

c）配置项测试；

d）系统测试。

可根据软件的规模、类型、完整性等级选择测试级别。

回归测试可出现在上述每个测试级别中，并贯穿于整个软件生存周期，故单独分级进行描述。

4.3　测试内容

本标准从GB/T16260.1 定义的质量子特性角度出发，确定软件部件测试、软件配置项测试和系统测试的测试内容，即，从适合性、准确性、互操作性、安全保密性、容错性、成熟性、易恢复性、

易理解性、易学性、易操作性、吸引性、时间特性、资源利用性、易改变性、稳定性、易测试性、易分析性、适应性、易安装性、易替换性、共存性和依从性方面确定测试内容，它们与传统的测试内容分类，其对应关系参见附录D。

4.4 测试过程

4.4.1 概述

软件测试过程包括四项活动，按顺序分别是：测试策划、测试设计和实现、测试执行、测试总结。

4.4.2 测试策划

确定需要测试的内容或质量特性；确定测试的充分性要求；提出测试的基本方法；确定测试的资源和技术需求；制定测试资源计划和测试进度计划。

4.4.3 测试设计与实现

分析测试用例集的层次结构，选取和设计测试用例；获取并验证测试数据；根据测试资源、风险等约束条件，确定测试用例执行顺序；获取测试资源，开发测试软件；建立并校准测试环境；进行测试就绪审查，主要审查测试计划的合理性和测试用例的正确性、有效性和覆盖充分性，审查测试组织、环境和设备工具是否齐备并符合要求。在进入下一阶段工作之前，应通过测试就绪评审。

4.4.4 测试执行

执行测试用例，获取测试结果；分析并判定测试结果。同时，根据不同的判定结果采取相应的措施；对测试过程的正常或异常终止情况进行核对，并根据核对结果，对未达到测试终止条件的测试用例，决定是停止测试，还是需要修改或补充测试用例集，并进一步测试。

4.4.5 测试总结

评估测试效果和被测软件项，描述测试状态。如，实际测试与测试计划和测试说明的差异、测试充分性分析、未能解决的测试事件等；描述被测软件项目状态，如，被测软件与需求的差异，发现的软件错误等；最后，完成软件测试报告，并通过测试评审。

4.5 测试方法

4.5.1 静态测试方法

静态测试方法包括检查单和静态分析方法，对文档的静态测试方法主要是以检查单的形式进行，而对代码的静态测试方法一般采用代码审查、代码走查和静态分析，静态分析一般包括控制流分析、数据流分析、接口分析和表达式分析。

应对软件代码进行审查、走查或静态分析；对于规模较小、安全性要求很高的代码也可进行形式化证明。

4.5.2 动态测试方法

动态测试方法一般采用白盒测试方法和黑盒测试方法。黑盒测试方法一般包括功能分解、边界值分析、判定表、因果图、随机测试、猜错法和正交试验法等；白盒测试方法一般包括控制流测试（语句覆盖测试、分支覆盖测试、条件覆盖测试、条件组合覆盖测试、路径覆盖测试）、数据流测试、程序变异、程序插桩、域测试和符号求值等。

在软件动态测试过程中，应采用适当的测试方法，实现测试要求。配置项测试和系统测试一般采用黑盒测试方法；部件测试一般主要采用黑盒测试方法，辅助以白盒测试方法；单元测试一般采用白盒测试方法，辅助以黑盒测试方法。

4.6 测试用例

4.6.1 测试用例设计原则

设计测试用例时，应遵循以下原则：

a) 基于测试需求的原则。应按照测试级别的不同要求，设计测试用例。如，单元测试依据详

细设计说明，部件测试依据概要设计说明，配置项测试依据软件需求规格说明，系统测试依据用户需求（系统/子系统设计说明、软件开发计划等）。

b) 基于测试方法的原则。应明确所采用的测试用例设计方法。为达到不同的测试充分性要求，应采用相应的测试方法，如等价类划分、边界值分析、猜错法、因果图等方法。

c) 兼顾测试充分性和效率的原则。测试用例集应兼顾测试的充分性和测试的效率；每个测试用例的内容也应完整，具有可操作性。

d) 测试执行的可重复性原则。应保证测试用例执行的可重复性。

4.6.2 测试用例要素

每个测试用例应包括以下要素：

a) 名称和标识。每个测试用例应有唯一的名称和标识。

b) 测试追踪。说明测试所依据的内容来源，如系统测试依据的是用户需求，配置项测试依据的是软件需求，部件测试和单元测试依据的软件设计。

c) 用例说明。简要描述测试的对象、目的和所采用的测试方法。

d) 测试的初始化要求。应考虑下述初始化要求：

1) 硬件配置。被测系统的硬件配置情况，包括硬件条件或电气状态；
2) 软件配置。被测系统的软件配置情况，包括测试的初始条件；
3) 测试配置。测试系统的配置情况，如用于测试的模拟系统和测试工具等的配置情况；
4) 参数设置。测试开始前的设置，如标志、第一断点、指针、控制参数和初始化数据等的设置；
5) 其它对于测试用例的特殊说明。

e) 测试的输入。在测试用例执行中发送给被测对象的所有测试命令、数据和信号等。对于每个测试用例应提供如下内容：

1) 每个测试输入的具体内容（如确定的数值、状态或信号等）及其性质（如有效值、无效值、边界值等）；
2) 测试输入的来源（例如，测试程序产生、磁盘文件、通过网络接收、手工键盘输入等），以及选择输入所使用的方法（例如，等价类划分、边界值分析、错误推测、因果图、功能图方法等）；
3) 测试输入是真实的还是模拟的；
4) 测试输入的时间顺序或事件顺序。

f) 期望测试结果。说明测试用例执行中由被测软件所产生的期望测试结果，即经过验证，认为正确的结果。必要时，应提供中间的期望结果。期望测试结果应该有具体内容，如确定的数值、状态或信号等，不应是不确切的概念或笼统的描述。

g) 评估测试结果的标准。判断测试用例执行中产生的中间和最后结果是否正确的标准。对于每个测试结果，应根据不同情况提供如下信息：

1) 实际测试结果所需的精度；
2) 实际测试结果与期望结果之间的差异允许的上限、下限；
3) 时间的最大和最小间隔，或事件数目的最大和最小值；
4) 实际测试结果不确定时，再测试的条件；
5) 与产生测试结果有关的出错处理；
6) 上面没有提及的其它标准。

h) 操作过程。实施测试用例的执行步骤。把测试的操作过程定义为一系列按照执行顺序排列的相对独立的步骤，对于每个操作应提供：

1) 每一步所需的测试操作动作、测试程序的输入、设备操作等；
2) 每一步期望的测试结果；

3) 每一步的评估标准；
4) 程序终止伴随的动作或错误指示；
5) 获取和分析实际测试结果的过程。

i) 前提和约束。在测试用例说明中施加的所有前提条件和约束条件，如果有特别限制、参数偏差或异常处理，应该标识出来，并要说明它们对测试用例的影响。

j) 测试终止条件。说明测试正常终止和异常终止的条件。

4.7 测试管理

4.7.1 过程管理

软件测试应由相对独立的人员进行。根据软件项目的规模等级和完整性等级以及测试级别，软件测试可由不同机构组织实施。

应对测试过程中的测试活动和测试资源进行管理。

4.7.2 配置管理

应按照软件配置管理的要求，将测试过程中产生的各种软件工作产品纳入配置管理。由开发组织实施的软件测试，应将测试工作产品纳入软件项目的配置管理；由独立测试组织实施的软件测试，应建立配置管理库，将被测试对象和测试工作产品纳入配置管理。

4.7.3 评审

4.7.3.1 测试就绪评审

在测试执行前，对测试计划和测试说明等进行审查，审查测试计划的合理性、测试用例的正确性、科学性和覆盖充分性，以及测试组织、测试环境和设备工具是否齐全并符合技术要求等。审查的具体内容和要求应包括：

a) 审查测试文档内容完整性、正确性和规范性；
b) 通过比较测试环境与软件真实运行的软件、硬件环境的差异，审查测试环境要求是否正确合理，满足测试要求；
c) 审查测试活动的独立性；
d) 审查测试项选择的完整性和合理性；
e) 审查测试用例的可行性、正确性和充分性。

4.7.3.2 测试评审

在测试完成后，审查测试过程和测试结果的有效性，确定是否达到测试目的。主要对测试记录、测试报告进行审查，其具体内容和要求应包括：

a) 审查文档和记录内容完整性、正确性和规范性；
b) 审查测试活动的独立性和有效性；
c) 审查测试环境是否符合测试要求；
d) 审查测试记录、测试数据以及测试报告内容与实际测试过程和结果的一致性；
e) 审查实际测试过程与测试计划和测试说明的一致性；
f) 审查未测试项和新增测试项的合理性；
g) 审查测试结果的真实性和正确性；
h) 审查对测试过程中出现异常的处理的正确性。

4.8 文档编写

软件测试文档一般包括测试计划、测试说明、测试报告、测试记录和测试问题报告（软件测试常用模板参见附录 C），测试文档的基本内容和要求见 GB/T9386。

按照 GB/T 18492，根据软件的完整性等级和软件规模等级进行合理的取舍与合并，其要求见表 1。

表1　测试文档的取舍与合并要求

性质 / 文档 取舍	规模：巨、中、大	规模：小、微		关键等级：A、B	关键等级：C、D	
测试计划	√	√	●	√	√	●
测试说明	√	√		√	√	
测试报告	√	√		√	√	
测试记录	√	√		√	√	
测试问题报告	√	√		√	√	
注：√　表示选取，●　表示合并。						

表 1 中指出文档的合并，在两个文档或多个文档合并时，具有相似内容的文档的第 1、2、3 章只用一次，即进行合并，后续的章节号依次编排。

4.9　测试工具

4.9.1　测试工具分类

软件测试工具可分为静态测试工具、动态测试工具和其它支持测试活动的工具，每类测试工具在功能和其他特征方面具有相似之处，支持一个或多个测试活动（见表 2）。应根据测试要求选择合适的工具。

表2　软件测试工具分类表

工具类型	功能和特征说明	举例	备注
静态测试工具	对软件需求、结构设计、详细设计和代码进行评审、走查和审查的工具。	复杂度分析、数据流分析、控制流分析、接口分析、句法和语义分析等工具。	针对软件需求、结构设计、详细设计的静态分析工具很少。
动态测试工具	支持执行测试用例和评估测试结果的工具，包括支持选择测试用例、设置环境、运行所选择测试、记录执行活动、故障分析和测试工作有效性评估等。	覆盖分析、捕获和回放、存储器测试、变异测试、仿真器及性能分析、测试用例管理等。	测试捕获和回放及数据生成器可用于测试设计。
支持测试过程活动的其它工具	支持测试计划、测试设计和整个测试过程的工具。	测试计划生成、测试进度和人员安排评估、基于需求的测试设计、测试数据生成、问题管理和测试配置管理等工具。	复杂度分析可用于测试计划的制定，捕获和回放、覆盖分析可用于测试设计与实现。

4.9.2　测试工具选择

软件测试应尽量采用测试工具，避免或减少人工工作。为让工具在测试工作中发挥应有的作用，应确定工具的详细需求，并制定统一的工具评估、采购（开发）、培训、实施和维护计划。

选择软件测试工具应考虑如下因素：

a）软件测试工具的需求及确认。

1) 应明确对测试工具的功能、性能、安全性等需求，并据此进行验证或确认。

2) 可通过在实际运行环境下的演示来确认工具是否满足需求，演示应依据工具的功能和技术特征、用户使用信息（安装和使用手册等）以及工具的操作环境描述等进行。

b）成本和收益分析。

1) 估计工具的总成本，除了最基本的产品价格，总成本还包括附加成本，如工具的挑选、安装、运行、培训、维护和支持等成本，以及为使用工具而改变测试过程或流程的成本等。

2) 分析工具的总体收益，如工具的首次使用范围和长期使用前景、工具应用效果、与其

它工具协同工作所提高的生产力程度等。

c）测试工具的整体质量因素。

1) 易用性；
2) 互操作性；
3) 稳定性；
4) 经济实用性；
5) 维护性。

4.10 软件完整性等级与测试的关系

根据失效所造成后果的危害程度，计算机软件的完整性等级被分为 A、B、C、D 四个等级（其定义见 GB/T 18492）。不同完整性等级软件的安全性要求不同，对软件的测试内容、测试要求和测试所采用方法也有所不同。针对各种不同的软件测试级别，应有安全性方面考虑。

5 单元测试

5.1 测试对象和目的

5.1.1 测试对象

软件单元测试的对象是软件单元。

5.1.2 测试目的

软件单元测试的目的是检查每个软件单元能否正确地实现设计说明中的功能、性能、接口和其它设计约束等要求，发现单元内可能存在的各种错误。

5.2 测试的组织和管理

一般由软件的供方组织并实施软件单元测试，也可委托第三方进行软件单元测试。软件单元测试的人员配备见表 3。

软件单元测试的工作产品一般应纳入软件的配置管理中。

表3 单元测试人员配备情况表

工作角色	具体职责
测试项目负责人	管理监督测试项目，提供技术指导，获取适当的资源，技术协调，负责项目的安全保密和质量管理。
测试分析员	确定测试计划、测试内容、测试方法、测试数据生成方法、测试（软、硬件）环境、测试工具，评估测试工作的有效性。
测试设计员	设计测试用例，确定测试用例的优先级，建立测试环境。
测试程序员	编写测试辅助软件。
测试员	执行测试、记录测试结果。
测试系统管理员	对测试环境和资产进行管理和维护。
注：一个人可承担多个角色的工作，一个角色可由多个人承担。	

5.3 技术要求

软件单元测试一般应符合以下技术要求：

a) 对软件设计文档规定的软件单元的功能、性能、接口等应逐项进行测试；

b) 每个软件特性应至少被一个正常测试用例和一个被认可的异常测试用例覆盖；

c) 测试用例的输入应至少包括有效等价类值、无效等价类值和边界数据值；

d) 在对软件单元进行动态测试之前，一般应对软件单元的源代码进行静态测试；

e) 语句覆盖率达到100％；

f) 分支覆盖率要达到100％；

g) 对输出数据及其格式进行测试。

对具体的软件单元，可根据软件测试合同或项目计划以及软件单元的重要性、完整性等级等要求对上述内容进行裁剪。

5.4 测试内容

5.4.1 总则

当静态测试时，所测试的内容与选择的测试方法有关。如，采用代码审查方法，通常要对寄存器的使用、程序格式、入口和出口的连接、程序语言的使用、存储器的使用等内容进行检查；采用静态分析方法，通常要对软件单元的控制流、数据流、接口、表达式等内容进行分析。详细内容可参见附录 A.1 中各种静态测试方法的描述。

当动态测试时，通常对软件单元的功能、性能、接口、局部数据结构、独立路径、错误处理、边界条件和内存使用情况进行测试。通常对软件单元接口的测试优先于其它内容的测试。对具体的软件单元，应根据软件测试合同或项目计划、软件设计文档的要求及选择的测试方法确定测试的具体内容。

5.4.2 接口

测试接口一般应包括以下内容：

a) 调用被测单元时的实际参数与被单元的形式参数的个数、属性、量纲、顺序是否一致；

b) 被测单元调用子模块时，传递给子模块的实际参数与子模块的形式参数的个数、属性、量纲、顺序是否一致；

c) 是否修改了只作为输入值的形式参数；

d) 调用内部函数的参数个数、属性、量纲、顺序是否正确；

e) 被测单元在使用全局变量时是否与全局变量的定义一致；

f) 在单元有多个入口的情况下，是否引用了与当前入口无关的参数；

g) 常数是否当作变量来传递；

h) 输入/输出文件属性的正确性；

i) OPEN 语句的正确性；

j) CLOSE 语句的正确性；

k) 规定的输入/输出格式说明与输入/输出语句是否匹配；

l) 缓冲区容量与记录长度是否匹配；

m) 文件是否先打开后使用；

n) 文件结束条件的判断和处理的正确性；

o) 输入/输出错误是否检查并做了处理以及处理的正确性。

5.4.3 局部数据结构

测试软件单元内部的数据能否保持其完整性，包括内部数据内容、格式及相互关系。

应设计测试用例检查以下错误：

a) 不正确或不一致的数据类型说明；

b) 错误的变量名，如变量名拼写错或缩写错；

c) 使用尚未赋值或尚未初始化的变量；

d) 错误的初始值或错误的缺省值；

e) 不一致的数据类型；

f) 下溢、上溢或是地址错误；

g) 全局数据对软件单元的影响。

5.4.4 独立路径

独立路径是指在程序中至少引进一个新的处理语句集合或一个新条件的任一路径。在程序的控制流图中，一条独立路径是至少包含有一条在其他独立路径中从未有过的边的路径。应设计适当的测试用例，对软件单元中的独立路径进行测试，特别是对独立路径中的基本路径进行测试。基本路径指在程序控制流图中，通过对控制构造的环路复杂性分析而导出的基本的、可执行的独立路径集合。

5.4.5 边界条件

应测试软件单元在边界处能否正常工作，如，测试处理 n 维数组的第 n 个元素；测试循环执行到最后一次执行循环体；测试取最大值或最小值；测试数据流、控制流中刚好等于、大于或小于确定的比较值，等等。

5.4.6 错误处理

测试软件单元在运行过程中发生错误时，其错误处理措施是否有效。

良好的单元设计要求能预见到程序投入运行后可能发生的错误，并给出相应的处理措施。这种错误处理也应当是软件单元功能的一部分。一般若出现下列情况之一，则表明软件单元的错误处理功能包含错误或缺陷：

a) 错误的描述难以理解；

b) 在对错误进行处理之前，错误条件已经引起系统的干预；

c) 所提供的错误描述信息不足以确定造成错误的位置或原因；

d) 显示的错误提示与实际错误不符；

e) 对错误条件的处理不正确；

f) 意外的处理不当；

g) 联机条件处理（即交互处理等）不正确。

5.4.7 功能

应对软件设计文档规定的软件单元的功能逐项进行测试。

5.4.8 性能

按软件设计文档的要求，对软件单元的性能（如精度、时间、容量等）进行测试。

5.4.9 内存使用

检查内存的使用情况，特别是动态申请的内存在使用上的错误（如对空指针赋值、指针使用越

界、内存泄露等）。

5.5 测试环境

测试环境包括测试的运行环境和测试工具环境。运行环境一般应符合软件测试合同或项目计划的要求，通常是开发环境或仿真环境。测试工具一般要求是经过认可的工具。

5.6 测试方法

软件单元测试一般应采用静态测试方法和动态测试方法。通常静态测试先于动态测试。

5.7 进入条件

进入软件单元测试一般应具备下列条件：

a) 具有测试合同或项目计划；

b) 具有软件设计文档，包含接口设计文档；

c) 所提交的被测软件单元受控；

d) 软件单元代码通过编译或汇编。

5.8 结束条件

结束条件用来评价软件单元的测试工作是否达到要求，通常包括下列条款：

a) 已按要求完成了合同或项目计划所规定的测试任务；

b) 实际测试过程遵循了原定的软件单元测试计划和软件单元测试说明；

c) 客观、详细地记录了测试过程和测试中发现的所有问题；

d) 测试文档齐全、符合规范；

e) 测试的全过程自始至终在控制下进行；

f) 测试中的问题或异常有合理解释或正确有效的处理；

g) 测试工作通过了软件单元测试审查；

h) 全部测试文档、被测软件单元、测试支持软件和评审结果已纳入配置管理。

5.9 测试过程

5.9.1 测试策划

测试分析人员一般根据测试合同或项目计划和被测试软件的设计文档对被测试软件单元进行分析，并确定以下内容：

a) 确定测试充分性要求。根据软件单元的重要性、软件单元测试目标和约束条件，确定测试应覆盖的范围及每一范围所要求的覆盖程度（如，分支覆盖率、语句覆盖率、功能覆盖率、单元的每一软件特性应至少被一个正常的测试用例和一个异常的测试用例所覆盖）；

b) 确定测试终止的要求。指定测试过程正常终止的条件（如，测试充分性是否达到要求），确定导致测试过程异常终止的可能情况（如软件编码错误）；

c) 确定用于测试的资源要求，包括软件（如操作系统、编译软件、静态分析软件、测试数据产生软件、测试结果获取和处理软件、测试驱动软件等）、硬件（如计算机、设备接口等）、人员数量、人员技能等；

d) 确定需要测试的软件特性。根据软件设计文档的描述确定软件单元的功能、性能、状态、接口、数据结构、设计约束等内容和要求，对其标识。若需要，将其分类。并从中确定需测试的软件特性；

e) 确定测试需要的技术和方法，如，测试数据生成与验证技术、测试数据输入技术、测试结果获取技术；

f) 根据测试合同或项目计划的要求和被测软件的特点，确定测试结束条件；

g) 确定由资源和被测试软件单元所决定的单元测试活动的进度。

根据上述分析研究结果，按照GB/T9386的要求编写软件单元测试计划。

应对软件单元测试计划进行评审。审查测试的范围和内容、资源、进度、各方责任等是否明确、测试方法是否合理、有效和可行，测试文档是否符合规范，测试活动是否独立。一般情况下，由软件的供方自行组织评审，评审细则也自行制定。在软件单元测试计划通过评审后，进入下一步工作；否则，需要重新进行单元测试的策划。

5.9.2 测试设计和实现

软件单元测试的设计和实现工作由测试设计人员和测试程序员完成，一般根据软件单元测试计划完成以下工作：

a) 设计测试用例。将需测试的软件特性分解，针对分解后的每种情况设计测试用例，每个测试用例的设计应符合4.6的要求；

b) 获取测试数据，包括获取现有的测试数据和生成新的数据，并按照要求验证所有数据；

c) 确定测试顺序，可从资源约束、风险以及测试用例失效造成的影响或后果几个方面考虑；

d) 获取测试资源，对于支持测试的软件和硬件，有的可从现有的工具中选定，有的需要研制开发；

e) 编写测试程序，包括开发测试支持工具，单元测试的驱动模块和桩模块；

f) 建立和校准测试环境；

g) 按照GB/ T9386的要求编写软件单元测试说明。

应对软件单元测试说明进行评审。审查测试用例是否正确、可行和充分，测试环境是否正确、合理，测试文档是否符合规范。通常由软件测试方自行组织单元测试的评审，评审细则也自行制定。在软件单元测试说明通过评审后，进入下一步工作；否则，需要重新进行测试设计和实现。

5.9.3 测试执行

执行测试的工作由测试员和测试分析员完成。

软件测试员的主要工作是按照软件单元测试计划和软件单元测试说明的内容和要求执行测试。在执行过程中，测试员应认真观察并如实地记录测试过程、测试结果和发现的错误，认真填写测试记录（参见附录C.2）。

测试分析员的工作主要有两方面。第一，根据每个测试用例的期望测试结果、实际测试结果和评价准则判定该测试用例是否通过，并将结果记录在软件测试记录中。如果测试用例不通过，测试分析员应认真分析情况，并根据以下情况采取相应措施：

a）软件单元测试说明和测试数据的错误。采取的措施是：改正错误，将改正错误信息详细记录，然后重新运行该测试；

b）执行测试步骤时的错误。采取的措施是：重新运行未正确执行的测试步骤；

c）测试环境（包括软件环境和硬件环境）中的错误。采取的措施是：修正测试环境，将环境修正情况详细记录，重新运行该测试；如果不能修正环境，记录理由，再核对终止情况；

d）软件单元的实现错误。采取的措施是：填写软件问题报告单（参见附录C.3），可提出软件修改建议，然后继续进行测试；或者把错误与异常终止情况进行比较，核对终止情况。软件更改完毕后，应根据情况对其进行回归测试；

e）软件单元的设计错误。采取的措施是：填写软件问题报告单（参见附录C.3），可提出软件修改建议，然后继续进行测试；或者把错误与异常终止情况进行比较，核对终止情况。软件更改完毕后，应根据情况对其进行回归测试或重新组织测试，回归测试中需要相应地修改测试设计和数据。

第二，当所有的测试用例都执行完毕，测试分析员要根据测试的充分性要求和失效记录，确定测试工作是否充分，是否需要增加新的测试。当测试过程正常终止时，如果发现测试工作不足，应对软件单元进行补充测试（具体要求见5.9.2和5.9.3），直到测试达到预期要求，并将附加的内容记录在软件单元测试报告中；如果不需要补充测试，则将正常终止情况记录在软件单元测试报告中。当测试过程异常终止时，应记录导致终止的条件、未完成的测试和未被修正的错误。

5.9.4　测试总结

测试分析员应根据被测试软件设计文档、软件单元测试计划、软件单元测试说明、测试记录和软件问题报告单等，对测试工作进行总结。一般包括下面几项工作：

a）总结软件单元测试计划和软件单元测试说明的变化情况及其原因，并记录在软件单元测试报告中；

b）对测试异常终止情况，确定未能被测试活动充分覆盖的范围，并将理由记录在测试报告中；

c）确定未能解决的软件测试事件以及不能解决的理由，并将理由记录在测试报告中；

d）总结测试所反映的软件单元与软件设计文档之间的差异，记录在测试报告中；

e）将测试结果连同所发现的错误情况同软件设计文档对照，评价软件单元设计与实现，提出软件改进建议，记录在测试报告中；

f）按照GB/T9386的要求编写软件单元测试报告，该报告应包括：测试结果分析、对软件单元的评估和建议。

g）根据测试记录和软件问题报告单编写测试问题报告。

应对测试执行活动、软件单元测试报告、测试记录和测试问题报告进行评审。审查测试执行活动的有效性，测试结果的正确性和合理性，是否达到了测试目的，测试文档是否符合规范。一般情况下，评审由软件测试方自行组织，评审细则也自行制定。

5.10　文档

软件单元测试完成后形成的文档有：

a) 软件单元测试计划

b) 软件单元测试说明

c) 软件单元测试报告

d) 软件单元测试记录

e) 软件单元测试问题报告

可根据需要对上述文档及文档的内容进行裁剪。裁剪要求见 4.8。

6 部件测试

6.1 测试对象和目的

6.1.1 测试对象

软件部件测试的对象包括：

a) 软件部件的组装过程；

b) 组装得到的软件部件。

6.1.2 测试目的

软件部件测试的目的是检验软件单元和软件部件之间的接口关系，并验证软件部件是否符合设计要求。

6.2 测试的组织和管理

软件部件测试一般由软件供方组织并实施，测试人员与开发人员应相对独立；也可委托第三方进行软件部件测试。软件部件测试的工作产品一般应纳入软件的配置管理中。软件部件测试的人员配备见表 4。

表4 部件测试人员配备情况表

工作角色	具体职责
测试项目负责人	管理监督测试项目，提供技术指导，获取适当的资源，技术协调和测试活动的质量管理，负责项目的安全保密。
测试分析员	确定测试计划、测试内容、测试方法、测试数据生成方法、测试（软、硬件）环境、测试工具，评估测试工作的有效性。
测试设计员	设计测试用例，确定测试用例的优先级，建立测试环境。
测试程序员	编写测试辅助软件。
测试员	执行测试、记录测试结果。
测试系统管理员	对测试环境和资产进行管理和维护。
注：一个人可承担多个角色的工作，一个角色可由多个人承担。	

6.3 技术要求

软件部件测试一般应符合以下技术要求：

a） 应对软件部件进行必要的静态测试，并先于动态测试；

b） 软件部件的每个特性应被至少一个正常的测试用例和一个被认可的异常测试用例覆盖；

c） 测试用例的输入应至少包括有效等价类值、无效等价类值和边界数据值；

d） 应采用增量法，测试组装新的软件部件；

e） 应逐项测试软件设计文档规定的软件部件的功能、性能等特性；

f） 应测试软件部件之间、软件部件和硬件之间的所有接口；

g） 应测试软件单元和软件部件之间的所有调用，达到100％的测试覆盖率；

h） 应测试软件部件的输出数据及其格式；

i） 应测试运行条件（如数据结构、输入/输出通道容量、内存空间、调用频率等）在边界状态下，进而在人为设定的状态下，软件部件的功能和性能；

j） 应按设计文档要求，对软件部件的功能、性能进行强度测试；

k） 对完整性的软件部件，应对其进行安全性分析，明确每一个危险状态和导致危险的可能原因，并对此进行针对性的测试。

对具体的软件部件，可根据软件测试合同或项目计划及软件部件的重要性、完整性等级对上述内容进行裁剪。

6.4 测试内容

6.4.1 总则

当对软件部件进行必要的静态测试时，所测试的内容与选择的静态测试方法有关。详见 5.4.1 中的静态测试内容部分。

当动态测试时，本标准从全局数据结构及软件部件的适合性、准确性、互操作性、容错性、时间特性、资源利用性这几个软件质量子特性方面考虑，确定测试内容。对具体的软件部件，应根据软件测试合同或项目计划、软件设计文档的要求及选择的测试方法来确定测试的具体内容。

6.4.2 全局数据结构

可测试全局数据结构的完整性，包括数据的内容、格式，并对内部数据结构对全局数据结构的影响进行测试。

6.4.3 适合性方面

从适合性方面考虑，应对软件设计文档分配给软件部件的每一项功能逐项进行测试。必要时，测试组装成的中间功能模块的功能。

6.4.4 准确性方面

从准确性方面考虑，可对软件部件中具有准确性要求的功能和精度要求的项（如，数据处理精度、时间控制精度、时间测量精度）进行测试。

6.4.5 互操作性方面

在互操作性方面，可考虑测试以下两种接口：所加入的软件单元和部件与已存在的软件单元和部件之间的接口；软件部件与支持其运行的其他软件部件、例行程序或硬件设备件的接口。对接口的输入和输出数据的格式、内容、传递方式、接口协议等进行测试。

测试软件部件的控制信息，如，信号或中断的来源，信号或中断的目的，信号或中断的优先级，信号或中断的表示格式或表示值，信号或中断的最小、最大和平均频率，响应方式和响应时间等。

6.4.6 容错性方面

在容错性方面，可考虑测试软件部件对错误输入、错误中断、漏中断等情况的容错能力，并考虑通过仿真平台或硬件测试设备形成一些人为条件，测试软件部件的功能、性能的降级情况。

6.4.7 时间特性方面

在时间特性方面，可考虑测试软件部件的运行时间，算法的最长路径下的计算时间。必要时，可考虑测试组装成的中间功能部件的运行时间。

6.4.8 资源利用性方面

在资源利用性方面，可考虑测试软件部件运行占用的内存空间。

必要时，可考虑测试组装成的中间功能部件运行占用的内存空间。

6.5 测试环境

测试环境应包括测试的运行环境和测试工具环境。

运行环境一般应符合软件测试合同或项目计划的要求，通常是开发环境或仿真环境。

测试工具一般要求是经过认可的工具。

6.6 测试方法

软件部件测试一般应采用静态测试方法和动态测试方法。静态测试方法常采用静态分析、代码走查等方法，动态测试方法常采用白盒测试方法和黑盒测试方法。通常，静态测试先于动态测试。

附录 A 介绍了常用的测试方法。

在由软件单元和软件部件组装成新的软件部件时，应根据软件单元和软件部件的特点选择便于测试的组装策略。

6.7 进入条件

进入软件部件测试一般应具备以下条件：

a) 具有测试合同或项目计划；

b) 具有软件需求规格说明（包含接口需求规格说明）、软件设计文档（含接口设计文档）、软件单元测试报告、被测软件部件的源程序和可执行代码；

c) 待测试的软件单元和部件已纳入软件受控库；

d) 待集成的软件单元已通过单元测试；

e) 软件部件源代码通过编译或汇编。

6.8 结束条件

结束条件用来评价软件部件的测试工作是否达到要求，通常包括下列条款：

a) 已按要求完成了测试合同或项目计划所规定的测试任务；

b) 实际测试过程遵循了原定的软件部件测试计划和软件部件测试说明；

c) 详细、客观地记录了测试过程中测试发现的所有问题；

d) 测试文档齐全、符合规范；

e) 测试工作通过了软件部件测试评审；

f) 测试的全过程自始至终在控制下进行；

g) 测试中的异常有合理解释或正确有效的处理；

h) 全部的测试文档、测试用例、测试软件、被测软件部件和评审结果已纳入配置管理。

6.9 测试过程

6.9.1 测试策划

测试分析人员应根据测试合同或项目计划和被测试软件的设计文档（含接口设计文档）对被测试软件部件进行分析，并确定以下内容：

a) 确定测试充分性要求。根据软件部件的重要性和完整性等级，确定测试应覆盖的范围及每一范围所要求的覆盖程度；

b) 确定测试终止的要求。指定测试过程正常终止的条件（如，测试的充分性要求是否达到），并确定导致测试过程异常终止的可能情况（如，软件接口错误）；

c) 确定用于测试的资源要求，包括软件（如操作系统、编译软件、静态分析软件、测试数据产生软件、测试结果获取和处理软件、测试驱动软件等）、硬件（如计算机、设备接口等）、人员数量、人员技能等；

d) 确定需要测试的软件特性。根据软件设计文档（含接口设计文档）的描述确定软件部件的功能、性能、状态、接口、数据结构、设计约束等内容和要求，对其标识。若需要，将其分类。并从中确定需测试的软件特性；

e) 确定测试需要的技术和方法，如，测试数据生成和验证技术、测试数据输入技术、测试结果获取技术、增量测试的组装策略；

f) 根据测试合同或项目计划的要求和被测软件的特点，确定测试结束条件；

g) 确定由资源和被测软件部件决定的软件部件测试活动的进度。

根据上述分析研究结果，按照GB/9386的要求编写软件部件测试计划。

应对软件部件测试计划进行评审。审查测试的范围和内容、资源、进度、各方责任等是否明确、测试方法是否合理、有效和可行，测试文档是否符合规范，测试活动是否独立。当测试活动由被测软件的供方实施时，软件部件测试计划的评审应纳入被测试软件的概要设计阶段评审。在软件部件测试计划通过评审后，进入下一步工作；否则，需要重新进行软件部件测试的策划。

6.9.2 测试设计和实现

测试设计和实现的工作由测试设计人员和测试程序员完成，一般根据软件部件测试计划完成以下工作：

a) 设计测试用例。将需测试的软件特性分解，针对分解后的每种情况设计测试用例，每个测试用例的设计应符合4.6的要求；

b) 获取测试数据，包括获取现有的测试数据和生成新的数据，并按照要求验证所有数据；

c) 确定测试顺序，可从资源约束、风险以及测试用例失效造成的影响或后果几个方面考虑；

d) 获取测试资源，对于支持测试的软件，有的需要从现有的工具中选定，有的需要开发；

e) 编写测试程序，包括开发测试支持工具，部件测试的驱动模块和桩模块；

f) 建立和校准测试环境；

g) 按照GB/T9386的要求编写软件部件测试说明。

应对软件部件测试说明进行评审。审查测试用例是否正确、可行和充分，测试环境是否正确、合理，测试文档是否符合规范。当测试活动由被测软件的供方实施时，软件部件测试说明的评审应纳入软件开发的阶段评审。在软件部件测试说明通过评审后，进入下一步工作；否则，需要重新对软件部件测试进行设计和实现。

6.9.3 测试执行

执行测试的工作由测试员和测试分析员完成。

测试员的主要工作是执行软件部件测试计划和软件部件测试说明中规定的测试项目和内容。在执行过程中，测试员应认真观察并如实地记录测试过程、测试结果和发现的错误，认真填写测试记录（参见附录C.2）。

测试分析员的工作主要有两方面。第一，根据每个测试用例的期望测试结果、实际测试结果和评价准则判定该测试用例是否通过。如果不通过，测试分析员应认真分析情况，并根据以下情况采取相应措施：

a) 软件部件测试说明和测试数据的错误。采取的措施是：改正错误，将改正错误信息详细记录，然后重新运行该测试；

b) 执行测试步骤时的错误。采取的措施是：重新运行未正确执行的测试步骤；

c) 测试环境（包括软件环境和硬件环境）中的错误。采取的措施是：修正测试环境，将环境修正情况详细记录，重新运行该测试；若不能修正环境，记录理由，再核对终止情况；

d) 软件部件的实现错误。采取的措施是：填写软件问题报告单（参见附录C.3），可提出软件修改建议，然后继续进行测试；或者把错误与异常终止情况进行比较，核对终止情况。软件更改完毕后，应根据情况对其进行回归测试；

e) 软件部件的设计错误。采取的措施是：填写软件问题报告单（参见附录C.3），可提出软件修改建议，然后继续进行测试；或者把错误与异常终止情况进行比较，核对终止情况。软件更改完毕后，应根据情况对其进行回归测试或重新组织测试，回归测试中需要相应地修改测试设计和数据。

第二，当所有的测试用例都执行完毕，测试分析员要根据测试的充分性要求和失效记录，确定测试工作是否充分，是否需要增加新的测试。当测试过程正常终止时，如果发现测试工作不足，应对软件部件进行补充测试（具体要求见6.9.2和6.9.3），直到测试达到预期要求，并将附加的内容记录在软件部件测试报告中；如果不需要补充测试，则将正常终止情况记录在软件部件测试报告中。当测试过程异常终止时，应记录导致终止的条件、未完成的测试和未被修正的错误。

6.9.4 测试总结

测试分析员应根据被测软件的设计文档（含接口设计文档）、部件测试计划、部件测试说明、测试记录和软件问题报告单等，分析和评估测试工作，一般包括下面几项工作：

a）总结软件部件测试计划和软件部件测试说明的变化情况及其原因，并记录在软件部件测试报告中；

b）对测试异常终止情况，确定未能被测试活动充分覆盖的范围，并将理由记录在测试报告中；

c）确定未能解决的软件测试事件以及不能解决的理由，并将理由记录在测试报告中；

d）总结测试所反映的软件部件与软件设计文档（含接口设计文档）之间的差异，记录在测试报告中；

e）将测试结果连同所发现的错误情况同软件设计文档（含接口设计文档）对照，评价软件部件的设计与实现，提出软件改进建议，记录在测试报告中；

f）按照GB/T9386的要求编写软件部件测试报告，该报告应包括：测试结果分析、对软件部件的评估和建议；

g）根据测试记录和软件问题报告单编写测试问题报告。

应对部件测试执行活动、软件部件测试报告、测试记录和测试问题报告进行评审。审查测试执行活动的有效性，测试结果的正确性和合理性，是否达到了测试目的，测试文档是否符合要求。当测试活动由被测试软件的供方实施时，评审由软件供方组织，软件需方和有关专家参加；当测试活动由独立的测试机构实施时，评审由软件测试机构组织，软件需方、供方和有关专家参加。

6.10 文档

软件部件测试完成后形成的文档有：

a) 软件部件测试计划

b) 软件部件测试说明

c) 软件部件测试报告

d) 软件部件测试记录

e) 软件部件测试问题报告

可根据需要对上述文档及文档的内容进行裁剪。裁剪要求见 4.8。

7 配置项测试

7.1 测试对象和目的

7.1.1 测试对象

软件配置项测试的对象是软件配置项。软件配置项是为独立的配置管理而设计的并且能满足最终用户功能的一组软件。

7.1.2 测试目的

软件配置项测试的目的是检验软件配置项与软件需求规格说明的一致性。

7.2 测试的组织和管理

应保证软件配置项测试工作的独立性。软件配置项测试一般由软件的供方组织，由独立于软件开发的组织实施。如果配置项测试由第三方实施，必须是军方认可的第三方测试组织。

软件配置项测试的人员配备见表 5。

表5 配置项测试人员配备情况表

工作角色	具体职责
测试项目负责人	管理监督测试项目，提供技术指导，获取适当的资源，制定基线，技术协调，负责项目的安全保密和质量管理。
测试分析员	确定测试计划、测试内容、测试方法、测试数据生成方法、测试[illegible]、硬件环境、测试工具，评估测试工作的有效性。
测试设计员	设计测试用例，确定测试用例的优先级，建立测试环境。
测试程序员	编写测试辅助软件。
测试员	执行测试、记录测试结果。
测试系统管理员	对测试环境和资产进行管理和维护。
配置管理员	设置、管理和维护测试配置管理数据库。
注 1：当软件的供方实施测试时，配置管理员由软件开发项目的配置管理员承担；当独立的测试组织实施测试时，应配备测试活动的配置管理员。 注 2：一个人可承担多个角色的工作，一个角色可由多个人承担。	

7.3 技术要求

软件配置项测试一般应符合以下技术要求：

a) 必要时，在高层控制流图中作结构覆盖测试；

b) 软件配置项的每个特性应至少被一个正常测试用例或一个被认可的异常测试用例所覆盖；

c) 测试用例的输入应至少包括有效等价类值、无效等价类值和边界数据值；

d) 应逐项测试软件需求规格说明规定的软件配置项的功能、性能等特性；

e) 应测试软件配置项的所有外部输入、输出接口（包括和硬件之间的接口）；

f) 应测试软件配置项的输出及其格式；

g) 应按软件需求规格说明的要求，测试软件配置项的安全保密性，包括数据的安全保密性；

h) 应测试人机交互界面提供的操作和显示界面，包括用非常规操作、误操作、快速操作测试界面的可靠性；

i) 应测试运行条件在边界状态和异常状态下，或在人为设定的状态下，软件配置项的功能和性能；

j) 应测试软件配置项的全部存储量、输入/输出通道和处理时间的余量；

k) 应按需求规格说明的要求，对软件配置项的功能、性能进行强度测试；

l) 应测试设计中用于提高软件配置项安全性、可靠性的结构、算法、容错、冗余、中断处理等方案；

m) 对完整性的软件配置项，应对其进行安全性分析，明确每一个危险状态和导致危险的可能原因，并对此进行针对性的测试；

n) 对有恢复或重置功能需求的软件配置项，应测试其恢复或重置功能和平均恢复时间，并且对每一类导致恢复或重置的情况进行测试；

o) 对不同的实际问题应外加相应的专门测试。

对具体的软件配置项，可根据软件测试合同或项目计划及软件配置项的重要性、完整性等级等要求对上述内容进行裁剪。

7.4 测试内容

7.4.1 总则

本标准从 GB /T6260.1 的子特性角度出发，确定软件配置项的测试内容。即从适合性、准确性、互操作性、安全保密性、容错性、成熟性、易恢复性、易理解性、易学性、易操作性、吸引性、时间特性、资源利用性、易改变性、稳定性、易测试性、易分析性、适应性、易安装性、易替换性、共存性和依从性方面考虑，确定测试内容。

对具体的软件配置项，可根据软件合同或项目计划及软件需求规格说明的要求对本标准给出的内容进行裁剪。

7.4.2 适合性方面

从适合性方面考虑，应测试软件需求规格说明规定的软件配置项的每一项功能。

7.4.3 准确性方面

从准确性方面考虑，可对软件配置项中具有准确性要求的功能和精度要求的项（如数据处理精度、时间控制精度、时间测量精度）进行测试。

7.4.4 互操作性方面

从互操作性方面考虑，可测试软件需求规格说明（含接口需求规格说明）和接口设计文档规定的软件配置项与外部设备的接口、与其他系统的接口。测试接口的格式和内容，包括数据交换的数据格式和内容；测试接口之间的协调性；测试软件配置项对系统每一个真实接口的正确性；测试软件配置项从接口接收和发送数据的能力；测试数据的约定、协议的一致性；测试软件配置项对外围设备接口特性的适应性。

7.4.5 安全保密性方面

从安全保密性方面考虑，可测试软件配置项及其数据访问的可控制性。

测试软件配置项防止非法操作的模式，包括防止非授权的创建、删除或修改程序或信息，必要时做强化异常操作的测试。

测试软件配置项防止数据被讹误和被破坏的能力。

测试软件配置项的加密和解密功能。

7.4.6 时间特性方面

从时间特性方面考虑，可测试软件配置项的响应时间、平均响应时间、响应极限时间；还可测试软件配置项的吞吐量、平均吞吐量、极限吞吐量；测试软件配置项的周转时间、平均周转时间、周转时间极限。

注 1：响应时间指软件配置项为完成一项规定任务所需的时间；平均响应时间指软件配置项执行若干并行任务所用的平均时间；响应极限时间指在最大负载条件下，软件配置项完成某项任务需要时间的极限；吞吐量指在给定的时间周期内软件配置项能成功完成的任务数量；平均吞吐量指在一个单位时间内软件配置项能处理并发任务的平均数；极限吞吐量指在最大负载条件下，在给定的时间周期内，软件配置项能处理的最多并发任务数；周转时间指从发出一条指令开始到一组相关的任务完成所用的时间；平均周转时间指在一个特定的负载条件下，对一些并发任务，从发出请求到任务完成所需要的平均时间；周转时间极限指在最大负载条件下，软件配置项完成一项任务所需要时间的极限。

在测试时，应标识和定义适合于软件应用的任务，并对多项任务进行测试，而不是仅测一项任务。

注 2：软件应用任务的例子，如在通信应用中的切换、数据包发送，在控制应用中的事件控制，在公共用户应用中由用户调用的功能产生的一个数据的输出等。

7.4.7 资源利用性方面

从资源利用性方面考虑，可测试软件配置项的输入/输出设备、内存和传输资源：

a) 执行大量的并发任务，测试输入/输出设备的利用时间；

b) 在使输入/输出负载达到最大的条件下，运行软件配置项，测试输入/输出负载极限；

c) 并发执行大量的任务，测试用户等待输入/输出设备操作完成需要的时间；

注：建议调查几次测试与运行实例中的最大时间与时间分布。

d) 在规定的负载下和在规定的时间范围内运行软件配置项，测试内存的利用情况；

e) 在最大负载下运行软件配置项，测试内存的利用情况；

f) 并发执行规定的数个任务，测试软件配置项的传输能力；

g) 在最大负载条件下和在规定的时间周期内，测试传输资源的利用情况；

h) 在传输负载最大的条件下，测试不同介质同步完成其任务的时间周期。

7.4.8 成熟性方面

在成熟性方面，可基于软件配置项操作剖面设计测试用例，根据实际使用的概率分布随机选择输入，运行软件配置项，测试软件配置项满足需求的程度并获取失效数据，其中包括对重要输入变量值的覆盖、对相关输入变量可能组合的覆盖、对设计输入空间与实际输入空间之间区域的覆盖、对各种使用功能的覆盖、对使用环境的覆盖。应在有代表性的使用环境中、以及可能影响软件配置项运行方式的环境中运行软件配置项，验证可靠性需求是否正确实现。对一些特殊的软件配置项，

如容错、实时嵌入式等，由于在一般的使用环境下常常很难在软件配置项中植入错误，应考虑多种测试环境。

测试软件配置项平均无故障时间。

选择可靠性增长模型（推荐模型参见附录 B），通过检测到的失效数和故障数，对软件配置项的可靠性进行预测。

7.4.9 容错性方面

从容错性方面考虑，可测试：

a) 软件配置项对中断发生的反应；

b) 软件配置项在边界条件下的反应；

c) 软件配置项的功能、性能的降级情况；

d) 软件配置项的各种误操作模式；

e) 软件配置项的各种故障模式（如数据超范围、死锁）；

f) 在多机系统出现故障需要切换时软件配置项的功能和性能的连续平稳性。

注：可用故障树分析技术检测误操作模式和故障模式。

7.4.10 易恢复性方面

从易恢复性方面考虑，可测试：

a) 具有自动修复功能的软件配置项的自动修复时间；

b) 软件配置项在特定的时间范围内的平均宕机时间；

c) 软件配置项在特定的时间范围内的平均恢复时间；

d) 软件配置项的可重启动并继续提供服务的能力；

e) 软件配置项的还原功能的还原能力。

7.4.11 易理解性方面

从易理解性方面考虑，可测试：

a) 软件配置项的各项功能，确认它们是否容易被识别和被理解；

b) 要求具有演示能力的功能，确认演示是否容易被访问、演示是否充分和有效；

c) 界面的输入和输出，确认输入和输出的格式和含义是否容易被理解。

7.4.12 易学性方面

从易学性方面考虑，可测试软件配置项的在线帮助，确认在线帮助是否容易定位，是否有效；还可对照用户手册或操作手册执行软件配置项，测试用户文档的有效性。

7.4.13 易操作性方面

从易操作性方面考虑，可测试：

a) 输入数据，确认软件配置项是否对输入数据进行有效性检查；

b) 要求具有中断执行的功能，确认它们能否在动作完成之前被取消；

c) 要求具有还原能力（数据库的事务回滚能力）的功能，确认它们能否在动作完成之后被撤消；

d) 包含参数设置的功能，确认参数是否易于选择、是否有缺省值；

e) 要求具有解释的消息，确认它们是否明确；

f) 要求具有界面提示能力的界面元素，确认它们是否有效；

g) 要求具有容错能力的功能和操作，确认软件配置项能否提示错误的风险、能否容易纠正错误的输入、能否从错误中恢复；

h) 要求具有定制能力的功能和操作，确认定制能力的有效性；

i） 要求具有运行状态监控能力的功能，确认它们的有效性。

注：以正确操作、误操作模式、非常规操作模式和快速操为框架设计测试用例。误操作模式有错误的数据类型作参数、错误的输入数据序列、错误的操作序列等。如有用户手册或操作手册，可对照手册逐条进行测试。

7.4.14 吸引性方面

从吸引性方面考虑，可测试软件配置项的人机交互界面能否定制。

7.4.15 易改变性方面

从易改变性方面考虑，可测试能否通过参数来改变软件配置项。

7.4.16 易测试性方面

从易测试性方面考虑，可测试软件配置项内置的测试功能，确认它们是否完整和有效。

7.4.17 易分析性方面

从易分析性方面考虑，可设计各种情况的测试用例运行软件配置项，并监测软件配置项的运行状态数据，检查这些数据是否容易获得、内容是否充分。如果软件配置项具有诊断功能，应测试该功能。

7.4.18 稳定性方面

本标准暂不规定软件配置项稳定性方面的测试内容。

7.4.19 适应性方面

从适应性方面考虑，可测试：

a) 软件配置项对诸如数据文件、数据块或数据库等数据结构的适应能力；

b) 测试软件配置项对硬件设备和网络设施等硬件环境的适应能力；

c) 测试软件配置项对系统软件或并行的应用软件等软件环境的适应能力；

d) 软件配置项是否易于移植。

7.4.20 易安装性方面

从易安装性方面考虑，可测试软件配置项安装的工作量、安装的可定制性、安装的简易性、手工安装操作的简易性、是否容易重新安装。

注 1：安装的简易性可分为三级：

a）最好：只需执行安装程序，安装过程中不需要人工干预；

b）好：按安装指南安装；

c）差：在安装中需要修改程序的源代码。

注 2：手工安装操作的简易性可分为四级：

a）非常容易：只需启动安装功能并观察安装过程；

b）容易：只需回答安装功能中提出的问题；

c）不容易：需要从表或填充框中看参数；

d）复杂：需要从文件中寻找参数，改变或写它们。

7.4.21 易替换性方面

当替换整个不同的软件配置项和用同一系列的高版本替换低版本时，在易替换性方面，可考虑测试：

a) 软件配置项能否继续使用被其替代的软件使用过的数据；

b) 软件配置项是否具有被其替代的软件中的类似功能。

7.4.22 共存性方面

从共存性方面考虑，可测试软件配置项与其他软件共同运行的情况。

7.4.23 依从性方面

当软件配置项在功能性、可靠性、易用性、效率、维护性和可移植性方面遵循了相关的标准、约定、风格指南或法规时，应酌情进行测试。

7.5 测试环境

测试环境应包括测试的运行环境和测试工具环境。

运行环境一般应符合软件测试合同或项目计划的要求，通常是实际计算机系统运行环境或相容的计算机系统运行环境。若选择仿真或模拟测试环境，应加以论证并获得批准。

测试工具一般要求是经过认可的工具。

7.6 测试方法

软件配置项测试一般应采用黑盒测试方法，详细内容参见附录 A.2.2。

7.7 进入条件

进入软件配置项测试一般应具备以下条件：

a） 具有测试合同或项目计划；

b） 具有软件需求规格说明（含接口需求规格说明）、软件设计文档（含接口设计文档）、用户手册或/和操作手册、软件部件测试报告、被测软件配置项的源程序和可执行代码；

c） 被测软件配置项的单元、部件已通过测试；

d） 所提交的被测软件配置项为本阶段最终版本，并已纳入软件配置管理中，取自受控库；

e） 软件配置项源代码通过编译或汇编；

f） 对需要固化运行的软件已提供固件。

7.8 结束条件

结束条件用来评价软件配置项的测试工作是否达到要求，通常包括下列条款：

a） 已按要求完成了软件测试合同或项目计划所规定的测试任务；

b） 实际测试过程遵循了原定的软件配置项测试计划和软件配置项测试说明；

c） 客观、完备地记录了测试过程和测试中发现的所有问题；

d） 测试文档齐全、符合规范；

e） 测试工作通过了软件配置项测试评审；

f） 测试的全过程自始至终在控制下进行；

g） 测试中的异常有合理解释或正确有效的处理；

h） 全部测试文档、测试用例、测试软件、被测软件配置项和评审结果已纳入配置管理。

7.9 测试过程

7.9.1 测试策划

测试分析人员应根据测试合同或项目计划和被测软件的需求规格说明（含接口需求规格说明）、

设计文档（含接口设计文档）对被测软件配置项进行分析，并确定以下内容：

a) 确定测试充分性要求。根据软件配置项的重要性和完整性等级，确定测试应覆盖的范围及每一范围所要求的覆盖程度；

b) 确定测试终止的要求。指定测试过程正常终止的条件（如测试的充分性要求是否达到），并确定导致测试过程异常终止的可能情况（如接口错误）；

c) 确定用于测试的资源要求，包括软件（如操作系统、编译软件、静态分析软件、测试数据产生软件、测试结果获取和处理软件、测试驱动软件等）、硬件（如计算机、设备接口等）、人员数量、人员技能等；

d) 确定需要测试的软件特性。根据软件测试合同或项目计划及软件需求规格说明（含接口需求规格说明）、设计文档（含接口设计文档）的描述确定软件配置项的功能、性能、状态、接口、数据结构、设计约束等内容和要求，对其标识。若需要，将其分类。从中确定需测试的软件特性；

e) 确定测试需要的技术和方法，如测试数据生成和验证技术、测试数据输入技术、测试结果获取技术、是否使用标准测试集等；

f) 根据测试合同或项目计划的要求和被测软件的特点，确定测试结束条件；

g) 确定由资源和被测软件配置项决定的配置项测试活动的进度。

根据上述分析研究结果，按照GB/T9386的要求编写软件配置项测试计划。

应对软件配置项测试计划进行评审。审查测试的范围和内容、资源、进度、各方责任等是否明确、测试方法是否合理、有效和可行，测试文档是否符合规范，测试活动是否独立。当测试活动由被测软件的供方实施时，软件配置项测试计划的评审应由软件的需方（在软件开发阶段评审）当测试活动由独立的测试机构实施时，软件配置项测试计划应通过软件的需方、供方和有关专家参加的评审。在软件配置项测试计划通过评审后，进入下一步工作；否则，需要重新进行配置项测试的策划。

7.9.2 测试设计和实现

测试设计和实现的工作由测试设计人员和测试程序员完成，一般根据软件配置项测试计划完成以下工作：

a) 设计测试用例。将需测试的软件特性分解，针对分解后的每种情况设计测试用例，每个测试用例的设计应符合4.6的要求；

b) 获取测试数据，包括获取现有的测试数据和生成新的数据，并按照要求验证所有数据；

c) 确定测试顺序，可从资源约束、风险以及测试用例失效造成的影响或后果几个方面考虑；

d) 获取测试资源，对于支持测试的软件，有的需要从现有的工具中选定，有的需要开发；

e) 编写测试程序，包括开发测试支持工具；

f) 建立和校准测试环境；

g) 按照GB/T9386的要求编写软件配置项测试说明。

应对软件配置项测试说明进行评审。审查测试用例是否正确、可行和充分，测试环境是否正确、合理，测试文档是否符合规范。当测试活动由被测软件的供方实施时，软件配置项测试说明应通过软件的需方和有关专家参加的审查；当测试活动由独立的测试机构实施时，软件配置项测试说明应

通过软件的需方、供方和有关专家参加的审查。在软件配置项测试说明通过评审后，进入下一步工作；否则，需要重新进行配置项测试的设计和实现。

7.9.3 测试执行

执行测试的工作由测试员和测试分析员完成。

测试员的主要工作是执行软件配置项测试计划和软件配置项测试说明中规定的测试项目和内容。在执行过程中，测试员应认真观察并如实地记录测试过程、测试结果和发现的错误，认真填写测试记录（参见附录C.2）。

测试分析员的工作主要有两方面。第一，根据每个测试用例的期望测试结果、实际测试结果和评价准则判定该测试用例是否通过。如果不通过，测试分析员应认真分析情况，并根据以下情况采取相应措施：

a) 软件配置项测试说明和测试数据的错误。采取的措施是：改正错误，将改正错误信息详细记录，然后重新运行该测试；

b) 执行测试步骤时的错误。采取的措施是：重新运行未正确执行的测试步骤；

c) 测试环境（包括软件环境和硬件环境）中的错误。采取的措施是：修正测试环境，将环境修正情况详细记录，重新运行该测试；若不能修正环境，记录理由，再核对终止情况；

d) 软件配置项的实现错误。采取的措施是：填写软件问题报告单（参见附录C.3），可提出软件修改建议，然后继续进行测试；或者把错误与异常终止情况进行比较，核对终止情况。软件更改完毕后，应根据情况对其进行回归测试；

e) 软件配置项的设计错误。采取的措施是：填写软件问题报告单（参见附录C.3），可提出软件修改建议，然后继续进行测试；或者把错误与异常终止情况进行比较，核对终止情况。软件更改完毕后，应根据情况对其进行回归测试或重新组织测试，回归测试中需要相应地修改测试设计和数据。

第二，当所有的测试用例都执行完毕，测试分析员要根据测试的充分性要求和失效记录，确定测试工作是否充分，是否需要增加新的测试。当测试过程正常终止时，如果发现测试工作不足，应对软件配置项进行补充测试（具体要求见7.9.2和7.9.3），直到测试达到预期要求，并将附加的内容记录在软件配置项测试报告中；如果不需要补充测试，则将正常终止情况记录在软件配置项测试报告中。当测试过程异常终止时，应记录导致终止的条件、未完成的测试和未被修正的错误。

7.9.4 测试总结

测试分析员应根据被测软件配置项的需求规格说明（含接口规格说明）、软件设计文档、配置项测试计划、配置项测试说明、测试记录和软件问题报告单等，分析和评估测试工作，一般包括下面几项工作：

a) 总结软件配置项测试计划和软件配置项测试说明的变化情况及其原因，并记录在测试报告中；

b) 对测试异常终止情况，确定未能被测试活动充分覆盖的范围，并将理由记录在测试报告中；

c) 确定未能解决的软件测试事件以及不能解决的理由，并将理由记录在测试报告中；

d) 总结测试所反映的软件配置项与软件需求规格说明（含接口规格说明）、软件设计文档（含接口设计文档）之间的差异，记录在测试报告中；

e) 将测试结果连同所发现的错误情况同软件需求规格说明（含接口规格说明）、软件设计文档（含接口设计文档）对照，评价软件配置项的设计与实现，提出软件改进建议，记录在测试报告中；

f) 按照GB/T9386的要求编写软件配置项测试报告，该报告应包括：测试结果分析、对软件配置项的评估和建议；

g) 根据测试记录和软件问题报告单编写测试问题报告。

应对软件配置项测试的执行活动、软件配置项测试报告、测试记录、测试问题报告进行评审。审查测试执行活动的有效性，测试结果的正确性和合理性，是否达到了测试目的，测试文档是否符合要求。当测试活动由被测软件的供方实施时，评审应由软件的供方组织，软件的需方和有关专家参加；当测试活动由独立的测试机构实施时，评审应由软件测试机构组织，软件的需方、供方和有关专家参加。

7.10 文档

软件配置项测试完成后形成的文档有：

a) 软件配置项测试计划

b) 软件配置项测试说明

c) 软件配置项测试报告

d) 软件配置项测试记录

e) 软件配置项测试问题报告

可根据需要对上述文档及文档的内容进行裁剪。裁剪的要求见 4.8。

8 系统测试

8.1 测试对象和目的

8.1.1 测试对象

系统测试的对象是完整的、集成的计算机系统，重点是新开发的软件配置项的集合。

8.1.2 测试目的

系统测试的目的是在真实系统工作环境下检验完整的软件配置项能否和系统正确连接，并满足系统/子系统设计文档和软件开发任务书规定的要求。

8.2 测试的组织和管理

系统测试一般由软件的需方组织，由独立于软件开发的组织实施。如果系统测试由第三方实施，必须是国家认可的第三方测试机构。

应加强系统测试的配置管理，已通过测试的系统状态和各项参数应详细记录，归档保存，未经测试负责人允许，任何人无权改变。

系统测试应严格按照由小到大、由简到繁、从局部到整体的程序进行。

系统测试的人员配备见表 6。

表6　系统测试人员配备情况表

工作角色	具体职责
测试项目负责人	管理监督测试项目，提供技术指导，获取适当的资源，制定基线，技术协调，负责项目的安全保密和质量管理。
测试分析员	确定测试计划、测试内容、测试方法、测试数据生成方法、测试（软、硬件）环境、测试工具，评估测试工作的有效性。
测试设计员	设计测试用例，确定测试用例的优先级，建立测试环境。
测试程序员	编写测试辅助软件。
测试员	执行测试、记录测试结果。
测试系统管理员	对测试环境和资产进行管理和维护。
配置管理员	设置、管理和维护测试配置管理数据库。
注 1：当软件的供方实施测试时，配置管理员由软件开发项目的配置管理员承担；当独立的测试组织实施测试时，应配备测试活动的配置管理员。 注 2：一个人可承担多个角色的工作，一个角色可由多个人承担。	

8.3　技术要求

系统测试一般应符合以下技术要求：

a）　系统的每个特性应至少被一个正常测试用例和一个被认可的异常测试用例所覆盖；

b）　测试用例的输入应至少包括有效等价类值、无效等价类值和边界数据值；

c）　应逐项测试系统/子系统设计说明规定的系统的功能、性能等特性；

d）　应测试软件配置项之间及软件配置项与硬件之间的接口；

e）　应测试系统的输出及其格式；

f）　应测试运行条件在边界状态和异常状态下，或在人为设定的状态下，系统的功能和性能；

g）　应测试系统访问和数据安全性；

h）　应测试系统的全部存储量、输入/输出通道和处理时间的余量；

i）　应按系统或子系统设计文档的要求，对系统的功能、性能进行强度测试；

j）　应测试设计中用于提高系统安全性、可靠性的结构、算法、容错、冗余、中断处理等方案；

k）　对完整性的系统，应对其进行安全性分析，明确每一个危险状态和导致危险的可能原因，并对此进行针对性的测试；

l）　对有恢复或重置功能需求的系统，应测试其恢复或重置功能和平均恢复时间，并且对每一类导致恢复或重置的情况进行测试；

m）　对不同的实际问题应外加相应的专门测试。

对具体的系统，可根据软件测试合同或项目计划及系统的重要性、完整性等级等要求对上述内容进行裁剪。

8.4　测试内容

8.4.1　总则

本标准从 GB /T16260.1 定义的软件质量子特性角度出发，确定系统测试的测试内容。即，从适合性、准确性、互操作性、安全保密性、容错性、成熟性、易恢复性、易理解性、易学性、易操作性、吸引性、时间特性、资源利用性、易改变性、稳定性、易测试性、易分析性、适应性、易安装性、易替换性、共存性和依从性方面考虑，确定测试内容。

对具体的系统，可根据测试合同或项目计划及系统/子系统设计文档的要求对本标准给出的内容进行裁剪。

8.4.2 适合性方面

从适合性方面考虑，应测试系统/子系统设计文档规定的系统的每一项功能。

8.4.3 准确性方面

从准确性方面考虑，可对系统中具有准确性要求的功能和精度要求的项（如数据处理精度、时间控制精度、时间测量精度）进行测试。

8.4.4 互操作性方面

从互操作性方面考虑，可测试系统/子系统设计文档、接口需求规格说明文档和接口设计文档规定的系统与外部设备的接口、与其他系统的接口。测试其格式和内容，包括数据交换的数据格式和内容；测试接口之间的协调性；测试软件对系统每一个真实接口的正确性；测试软件系统从接口接收和发送数据的能力；测试数据的约定、协议的一致性；测试软件系统对外围设备接口特性的适应性。

8.4.5 安全保密性方面

从安全保密性方面，可测试系统及其数据访问的可控制性。

测试系统防止非法操作的模式，包括防止非授权的创建、删除或修改程序或信息，必要时做强化异常操作的测试。

测试系统防止数据被讹误和被破坏的能力。

测试系统的加密和解密功能。

8.4.6 时间特性方面

从时间特性方面考虑，可测试系统的响应时间、平均响应时间、响应极限时间，系统的吞吐量、平均吞吐量、极限吞吐量，系统的周转时间、平均周转时间、周转时间极限。

注 1：响应时间指系统为完成一项规定任务所需的时间；平均响应时间指系统执行若干并行任务所需的平均时间；响应极限时间指在最大负载条件下，系统完成某项任务需要时间的极限；吞吐量指在给定的时间周期内系统能成功完成的任务数量；平均吞吐量指在一个单位时间内系统能处理并发任务的平均数；极限吞吐量指在最大负载条件下，在给定的时间周期内，系统能处理的最多并发任务数；周转时间指从发出一条指令开始到一组相关的任务完成的时间；平均周转时间指在一个特定的负载条件下，对一些并发任务，从发出请求到任务完成所需要的平均时间；周转时间极限指在最大负载条件下，系统完成一项任务所需要时间的极限。

在测试时，应标识和定义适合于软件应用的任务，并对多项任务进行测试，而不是仅测一项任务。

注 2：软件应用任务的例子，如在通信应用中的切换、数据包发送，在控制应用中的事件控制，在公共用户应用中由用户调用的功能产生的一个数据的输出等。

8.4.7 资源利用性方面

从资源利用性方面考虑，可测试系统的输入/输出设备、内存和和传输资源的利用情况：

a) 执行大量的并发任务，测试输入/输出设备的利用时间。

b) 在使输入/输出负载达到最大的系统条件下，运行系统，测试输入/输出负载极限。

c) 并发执行大量的任务，测试用户等待输入/输出设备操作完成需要的时间。

注：建议调查几次测试与运行实例中的最大时间与时间分布。

d) 在规定的负载下和在规定的时间范围内运行系统，测试内存的利用情况。

e) 在最大负载下运行系统，测试内存的利用情况。

f) 并发执行规定的数个任务，测试系统的传输能力。

g) 在系统负载最大的条件下和在规定的时间周期内，测试传输资源的利用情况。

h) 在系统传输负载最大的条件下，测试不同介质同步完成其任务的时间周期。

8.4.8 成熟性方面

在成熟性方面，可基于系统运行剖面设计测试用例，根据实际使用的概率分布随机选择输入，运行系统，测试系统满足需求的程度并获取失效数据，其中包括对重要输入变量值的覆盖、对相关输入变量可能组合的覆盖、对设计输入空间与实际输入空间之间区域的覆盖、对各种使用功能的覆盖、对使用环境的覆盖。应在有代表性的使用环境中、以及可能影响系统运行方式的环境中运行软件，验证系统的可靠性需求是否正确实现。对一些特殊的系统，如容错软件、实时嵌入式软件等，由于在一般的使用环境下常常很难在软件中植入错误，应考虑多种测试环境。

测试系统的平均无故障时间。

选择可靠性增长模型（推荐模型参见附录 B），通过检测到的失效数和故障数，对系统的可靠性进行预测。

8.4.9 容错性方面

从容错性方面考虑，可测试：

a) 系统对中断发生的反应。

b) 系统在边界条件下的反应。

c) 系统的功能、性能的降级情况。

d) 系统的各种误操作模式。

e) 系统的各种故障模式（如数据超范围、死锁）。

f) 测试在多机系统出现故障需要切换时系统的功能和性能的连续平稳性。

注：可用故障树分析技术检测误操作模式和故障模式。

8.4.10 易恢复性方面

从易恢复性方面考虑，可测试：

a) 具有自动修复功能的系统的自动修复的时间。

b) 系统在特定的时间范围内的平均宕机时间。

c) 系统在特定的时间范围内的平均恢复时间。

d) 系统的可重启动并继续提供服务的能力。

e) 系统的还原功能的还原能力。

8.4.11 易理解性方面

从易理解性方面考虑，可测试：

a) 系统的各项功能，确认它们是否容易被识别和被理解。

b) 要求具有演示能力的功能，确认演示是否容易被访问、演示是否充分和有效。

c) 界面的输入和输出，确认输入和输出的格式和含义是否容易被理解。

8.4.12 易学性方面

从易学性方面考虑，可测试系统的在线帮助，确认在线帮助是否容易定位，是否有效；还可对照用户手册或操作手册执行系统，测试用户文档的有效性。

8.4.13 易操作性方面

从易操作性方面考虑，可测试：

a) 输入数据，确认系统是否对输入数据进行有效性检查。

b) 要求具有中断执行的功能，确认它们能否在动作完成之前被取消。

c) 要求具有还原能力（数据库的事务回滚能力）的功能，确认它们能否在动作完成之后被撤消。

d) 包含参数设置的功能，确认参数是否易于选择、是否有缺省值。

e) 要求具有解释的消息，确认它们是否明确。

f) 要求具有界面提示能力的界面元素，确认它们是否有效。

g) 要求具有容错能力的功能和操作，确认系统能否提示错误的风险、能否容易纠正错误的输入、能否从错误中恢复。

h) 要求具有定制能力的功能和操作，确认定制能力的有效性。

i) 要求具有运行状态监控能力的功能，确认它们的有效性。

注：以正确操作、误操作模式、非常规操作模式和快速操作为原则设计测试用例。误操作模式有错误的数据类型作参数、错误的输入数据序列、错误的操作序列等。如有用户手册或操作手册，请对照手册逐条进行测试。

8.4.14 吸引性方面

从吸引性方面考虑，可测试系统的人机交互界面能否定制。

8.4.15 易改变性方面

从易改变性方面考虑，可测试能否通过参数来改变系统。

8.4.16 易测试性方面

从易测试性方面考虑，可测试软件内置的测试功能，确认它们是否完整和有效。

8.4.17 易分析性方面

从易分析性方面考虑，可设计各种情况的测试用例运行系统，并监测系统运行状态数据，检查这些数据是否容易获得、内容是否充分。如果软件具有诊断功能，应测试该功能。

8.4.18 稳定性方面

本标准暂不推荐软件稳定性方面的测试内容。

8.4.19 适应性方面

从适应性方面考虑，可测试：

a) 软件对诸如数据文件、数据块或数据库等数据结构的适应能力。

b) 软件对硬件设备和网络设施等硬件环境的适应能力。

c) 软件对系统软件或并行的应用软件等软件环境的适应能力。

d) 软件是否易于移植。

8.4.20 易安装性方面

从易安装性方面考虑，可测试软件安装的工作量、安装的可定制性、安装的简易性、手工安装操作的简易性、是否容易重新安装。

注 1：安装的简易性可分为三级：

a）最好：只需执行安装程序，安装过程中不需要人工干预；

b）好：按安装指南安装；

c）差：在安装中需要修改程序的源代码。

注 2：手工安装操作的简易性可分为四级：

a）非常容易：只需启动安装功能并观察安装过程；

b）容易：只需回答安装功能中提出的问题；

c）不容易：需要从表或填充框中看参数；

d）复杂：需要从文件中寻找参数，改变或写它们。

8.4.21 易替换性方面

当替换整个不同的软件系统和用同一软件系列的高版本替换低版本时，在易替换性方面，可考虑测试：

a) 软件能否继续使用被其替代的软件使用过的数据。

b) 软件是否具有被其替代的软件中的类似功能。

8.4.22 共存性方面

从共存性方面考虑，可测试软件与其他软件共同运行的情况。

8.4.23 依从性方面

当软件在功能性、可靠性、易用性、效率、维护性、可移植性方面进行测试时，应参照的标准、约定、风格指南或法规时，应酌情进行测试。

8.5 测试环境

测试环境应包括测试的运行环境和测试工具环境。

运行环境一般应符合软件测试合同或项目计划的要求，通常是软件及其所属系统的正式工作环境。

测试工具一般要求是经过认可的工具。

8.6 测试方法

系统测试一般应采用黑盒测试方法，详细内容参见附录 A.2.2。

8.7 进入条件

进入系统测试一般应具备以下条件：

a）具有测试合同或项目计划；

b）具有软件开发任务书、软件开发合同或系统/子系统设计文档、软件需求规格说明（含接口需求规格说明）、软件设计文档（含接口设计文档）、用户手册或（和）操作手册、软件配置项测试报告、被测系统的源程序和可执行代码；

c）软件系统的所有配置项已通过测试；

d）所提交的被测软件系统为本阶段最终版本，并已纳入软件配置管理中，取自受控库；

e）对需要固化运行的软件已提供固件。

8.8 结束条件

结束条件用来评价系统测试工作是否达到要求，通常包括下列条款：

a）已按要求完成软件测试合同或项目计划规定的测试任务；

b）实际测试过程遵循了原定的系统测试计划和系统测试说明；

c）客观、完备地记录了测试过程和测试中发现的所有问题；

d）测试文档齐全、符合规范；

e）测试工作通过了系统测试评审；

f）测试的全过程自始至终在控制下进行；

g）测试中的异常有合理解释或正确有效的处理；

h）全部的测试文档、测试用例、测试软件、被测软件系统和评审结果已纳入配置管理。

8.9 测试过程

8.9.1 测试策划

测试分析人员应根据测试合同或项目计划、被测软件的开发合同或系统/子系统设计文档分析被测系统，并确定以下内容：

a） 确定测试充分性要求。确定测试应覆盖的范围及每一范围所要求的覆盖程度；

b） 确定测试终止的要求。指定测试过程正常终止的条件（如测试充分性是否达到要求），并确定导致测试过程异常终止的可能情况（如接口错误）；

c） 确定用于测试的资源要求，包括软件（如操作系统、编译软件、静态分析软件、测试数据产生软件、测试结果获取和处理软件、测试驱动软件等）、硬件（如计算机、设备接口等）、人员数量、人员技能等；

d） 确定需要测试的软件特性。根据软件开发合同或系统/子系统设计文档的描述确定系统的功能、性能、状态、接口、数据结构、设计约束等内容和要求，对其标识。若需要，将其分类。并从中确定需测试的软件特性；

e） 确定测试需要的技术和方法，如测试数据生成和验证技术、测试数据输入技术、测试结果获取技术、是否使用标准测试集等；

f） 根据测试合同或项目计划的要求和被测软件的特点，确定测试结束条件；

g） 确定由资源和被测系统决定的系统测试活动的进度。

根据上述分析研究结果，按照 GB/T9386 的要求编写系统测试计划。

应对系统测试计划进行评审。审查测试的范围和内容、资源、进度、各方责任等是否明确、测试方法是否合理、有效和可行，测试文档是否符合规范，测试活动是否独立。当测试活动由被测软件的供方实施时，系统测试计划的评审应纳入软件开发过程的阶段评审；当测试活动由独立的测试机构实施时，系统测试计划应通过软件的需方、供方和有关专家参加的评审。在系统测试计划通过评审后，进入下一步工作；否则，需要重新进行系统测试的策划。

8.9.2 测试设计和实现

测试设计和实现的工作由测试设计人员和测试程序员完成，一般根据系统测试计划完成以下工作：

a） 设计测试用例。将需测试的软件特性分解，针对分解后的每种情况设计测试用例，每个测试用例的设计应符合 4.6 的要求；

b） 获取测试数据，包括获取现有的测试数据和生成新的数据，并按照要求验证所有数据；

c） 确定测试顺序，可从资源约束、风险以及测试用例失效造成的影响或后果几个方面考虑；

d） 获取测试资源，对于支持测试的软件，有的需要从现有的工具中选定，有的需要开发；

e） 编写测试程序，包括开发测试支持工具；

f） 建立和校准测试环境；

g） 按照 GBT9386 的要求编写系统测试说明。

应对系统测试说明进行评审。审查测试用例是否正确、可行和充分，测试环境是否正确、合理，测试文档是否符合规范。当测试活动由被测软件的供方实施时，评审应由软件的供方组织，软件的需方和有关专家参加；当测试活动由独立的测试机构实施时，评审应由测试机构组织，软件的需方、供方和有关专家参加。在系统测试说明通过评审后，进入下一步工作；否则，需要重新进行系统测试的设计和实现。

8.9.3 测试执行

执行测试的工作由测试员和测试分析员完成。

测试员的主要工作是执行系统测试计划和系统测试说明中规定的测试项目和内容。在执行过程中，测试员应认真观察并如实地记录测试过程、测试结果和发现的错误，认真填写测试记录（参见附录C.2）。

测试分析员的工作主要有两方面。第一，根据每个测试用例的期望测试结果、实际测试结果和评价准则判定该测试用例是否通过。如果不通过，测试分析员应认真分析情况，并根据以下情况采取相应措施：

a) 系统测试说明和测试数据的错误。采取的措施是：改正错误，将改正错误信息详细记录，然后重新运行该测试；

b) 执行测试步骤时的错误。采取的措施是：重新运行未正确执行的测试步骤；

c) 测试环境（包括软件环境和硬件环境）中的错误。采取的措施是：修正测试环境，将环境修正情况详细记录，重新运行该测试；若不能修正环境，记录理由，再核对终止情况；

d) 系统实现的错误。采取的措施是：填写软件问题报告单（参见附录C.3），可提出软件修改建议，然后继续进行测试；或者把错误与异常终止情况进行比较，核对终止情况。软件更改完毕后，应根据情况对其进行回归测试；

e) 系统设计的错误。采取的措施是：填写软件问题报告单（参见附录C.3），可提出软件修改建议，然后继续进行测试；或者把错误与异常终止情况进行比较，核对终止情况。软件更改完毕后，应根据情况对其进行回归测试或重新组织测试，回归测试中需要相应地修改测试设计和数据。

第二，当所有的测试用例都执行完毕，测试分析员要根据测试的充分性要求和失效记录，确定测试工作是否充分，是否需要增加新的测试。当测试过程正常终止时，如果发现测试工作不足，应对软件系统进行补充测试（具体要求见8.9.2和8.9.3），直到测试达到预期要求，并将附加的内容

记录在系统测试报告中；如果不需要补充测试，则将正常终止情况记录在系统测试报告中。当测试过程异常终止时，应记录导致终止的条件、未完成的测试和未被修正的错误。

8.9.4 测试总结

测试分析员应根据软件开发合同或系统/子系统设计文档、系统测试计划、系统测试说明、测试记录和软件问题报告单等，分析和评估测试工作，一般包括下面几项工作：

a) 总结系统测试计划和系统测试说明的变化情况及其原因，并记录在系统测试报告中；
b) 对测试异常终止情况，确定未能被测试活动充分覆盖的范围，并将理由记录在系统测试报告中；
c) 确定未能解决的软件测试事件以及不能解决的理由，并将理由记录在系统测试报告中；
d) 总结测试所反映的软件系统与软件开发合同或系统/子系统设计文档之间的差异，记录在系统测试报告中；
e) 将测试结果连同所发现的错误情况同软件开发合同或系统/子系统设计文档对照，评价软件系统的设计与实现，提出软件改进建议，记录在测试报告中；
f) 按照GB/T9386的要求编写系统测试报告，该报告应包括：测试结果分析、对软件系统的评估和建议；
g) 根据测试记录和软件问题报告单编写测试问题报告。

应对系统测试的执行活动、系统测试报告、测试记录、测试问题报告进行评审。审查测试执行活动的有效性，测试结果的正确性和合理性，是否达到了测试目的，测试文档是否符合要求。当测试活动由被测软件的供方实施时，评审应由软件的供方组织，软件的需方和有关专家参加；当测试活动由独立的测试机构实施时，评审应由测试机构组织，软件的需方、供方和有关专家参加。

8.10 文档

系统测试完成后形成的文档有：

a) 系统测试计划
b) 系统测试说明
c) 系统测试报告
d) 系统测试记录
e) 系统测试问题报告

可根据需要对上述文档及文档的内容进行裁剪。裁剪的要求见 4.8。

9 回归测试

9.1 测试对象和测试目的

9.1.1 测试对象

回归测试的对象包括：

a) 未通过软件单元测试的软件单元，在更改之后，应对其进行测试；
b) 未通过软件部件测试的软件部件，在更改之后，应对更改的软件单元和软件部件进行测试；
c) 未通过软件配置项测试的软件配置项，在更改之后，应对更改的软件单元、受更改影响的软件部件和软件配置项进行测试；

d) 未通过系统测试的软件，在更改之后，应对更改的软件单元、受更改影响的软件部件、软件配置项和系统进行测试；

e）因其他原因进行更改之后的软件单元、软件部件或软件配置项。

9.1.2 测试目的

回归测试的测试目的是：

a) 测试软件更改之后，更改部分的正确性和对更改需求的符合性；

b) 测试软件更改之后，软件原有的、正确的功能、性能和其他规定的要求的不损害性。

9.2 进入条件

进入回归测试一般应具备以下条件：

a）被测软件完成更改且已经置于软件配置管理之下；

b）相关的软件测试报告、软件更改报告单齐全；

c）具有相关测试的全部文档及资源；

d）具备相关测试的进入条件。

9.3 单元回归测试

9.3.1 测试组织和管理

通常应由原测试方组织并实施软件单元回归测试，特殊情况下可交由其他测试方进行软件单元回归测试，测试管理应纳入软件开发过程中。

9.3.2 技术要求

一般应符合原软件单元测试的技术要求，可根据更改情况酌情裁剪。当回归测试结果和原软件单元测试的正确结果不一致时，应对软件单元重新进行回归测试。

9.3.3 测试内容

一般应根据软件单元的更改情况确定软件单元回归测试的测试内容。可能包括以下三种情况：

a) 仅重复测试原软件单元测试做过的测试内容；

b) 修改原软件单元测试做过的测试内容；

c) 在前两者的基础上增加新的测试内容。

9.3.4 测试环境

软件单元回归测试的测试环境要求应与原软件单元测试的测试环境要求一致。

9.3.5 测试方法

当未增加新的测试内容时，软件单元回归测试应采用原软件单元测试的测试方法；否则，应根据情况选择适当的测试方法。

9.3.6 结束条件

软件单元回归测试的结束条件用来评价软件单元回归测试的工作是否达到要求。软件单元回归测试的结束条件应与原软件单元测试的结束条件一致。

另外，软件单元回归测试的文档应齐全、符合规范。

9.3.7 测试过程

软件单元回归测试的测试过程按顺序包括下面几步：

a）测试分析员根据测试问题报告和软件更改报告单，分析回归测试的测试范围，并确定原软件单元测试的充分性要求、终止要求、资源要求、软件特性、测试技术和方法的适用程度，

并酌情更改，确定回归测试的测试进度，按照GB/T9386完成软件单元回归测试计划。对软件单元回归测试计划进行评审，评审要求见软件单元测试计划的评审；

b）测试设计员和测试程序员根据软件单元回归测试计划确定测试用例，可从原软件单元测试说明中选择测试用例、或修改原有测试用例、或设计新的测试用例，补充相应的测试数据、测试资源和测试软件，建立相应的测试环境，确定相应的测试顺序，按照GB/T9386编写软件单元回归测试说明。对软件单元回归测试说明进行评审，评审要求见软件单元测试说明的评审；

c）测试员和测试分析员按照软件单元回归测试说明对更改的软件单元进行测试，具体要求见5.9.3；

d）测试分析员根据原测试问题报告、原软件更改报告单，软件单元回归测试计划、测试说明、测试记录、软件问题报告单对回归测试的工作进行总结，编写软件单元回归测试报告、测试问题报告，并对软件单元回归测试的执行活动、测试记录、软件单元回归测试报告和测试问题报告进行评审。具体要求见5.9.4。

9.3.8 文档

软件单元回归测试完成后形成的文档有：

a) 软件单元回归测试计划

b) 软件单元回归测试说明

c) 软件单元回归测试报告

d) 软件单元回归测试记录

e) 软件单元回归测试问题报告

可根据需要对上述文档及文档的内容进行裁剪。裁剪要求见4.8。

上述文档也可分别作为软件单元测试产生的文档的补充件。

9.4 部件回归测试

9.4.1 测试组织和管理

通常由软件的供方组织并实施软件部件回归测试，测试管理应纳入软件开发过程中；也可委托第三方实施软件部件回归测试。

9.4.2 技术要求

软件部件回归测试的技术要求一般应符合以下原则：

a）对更改的软件单元的测试，应符合原软件单元测试的技术要求，可根据更改情况进行裁剪；

b）对软件部件的测试，应符合原软件部件测试的技术要求，可根据受影响情况进行裁剪；

c）当回归测试结果和原软件单元测试和软件部件测试的正确结果不一致时，应对出现问题的软件单元和软件部件重新进行回归测试。

9.4.3 测试内容

软件部件回归测试的测试内容分两种情况考虑：

a）对更改的软件单元的测试可能存在以下三种情况：一是仅重复测试原软件单元测试做过的测试内容；二是修改原软件单元测试做过的测试内容；三是在前两者的基础上增加新的测试内容；

b） 对软件部件的测试。测试分析员应分析更改的软件单元对软件部件的影响域，并据此确定回归测试内容。可能存在三种情况：一是仅重复测试与更改相关的、并已在软件部件测试中做过的测试内容；二是修改与更改相关的、并已在软件部件测试中做过的测试内容；三是在前两者的基础上增加新的测试内容。

9.4.4 测试环境

软件部件回归测试的测试环境要求分两种情况：

a） 对更改的软件单元的测试，其测试环境要求应与原软件单元测试的测试环境要求一致；

b） 软件部件测试的测试环境要求应与原软件部件测试的测试环境要求一致。

9.4.5 测试方法

软件部件回归测试采用的测试方法分两种情况：

a） 对更改的软件单元的测试。当未增加新的测试内容时，对更改的软件单元的测试采用原软件单元测试的测试方法；否则，根据情况选择适当的测试方法；

b） 对软件部件的测试。当未增加新的测试内容时，软件部件回归测试采用原软件部件测试的测试方法；否则，根据情况选择适当的测试方法。

9.4.6 结束条件

软件部件回归测试的结束条件用来评价回归测试的工作是否达到要求，一般应符合以下原则：

a) 按照软件单元回归测试的要求（见 9.3），完成了对更改的软件单元的测试，并且无新问题出现；

b) 对软件部件的测试符合原软件部件测试的结束条件，并且无新问题出现。

另外，软件部件回归测试的文档应齐全、符合规范。

9.4.7 测试过程

软件部件回归测试的测试过程按顺序包括下面几步：

a） 按照9.3.7对更改的软件单元进行测试；

注：更改的软件单元通过测试后，才能对软件部件进行测试。

b） 测试分析员根据测试问题报告、软件更改报告单，分析软件部件测试的范围，确定原软件部件测试的充分性要求、终止要求、资源要求、软件特性、测试技术和方法的适用程度，并酌情更改，确定回归测试的测试进度，按照GB/T9386完成软件部件回归测试计划。对软件部件回归测试计划进行评审，评审要求见6.9.1中软件部件测试计划的评审；

c） 测试设计员和测试程序员根据软件部件回归测试计划确定测试用例，可从原软件部件测试说明中选择测试用例、或修改原有测试用例、或设计新的测试用例，补充相应的测试数据、测试资源和测试软件，建立相应的测试环境，确定相应的测试顺序，并按照GB/T9386编写软件部件回归测试说明。对软件部件回归测试说明进行评审，评审要求见6.9.2中软件部件测试说明的评审；

d） 测试员和测试分析员按照软件部件回归测试说明对软件部件进行测试，要求见6.9.3；

e） 测试分析员根据原测试问题报告、原软件更改报告单，软件部件回归测试计划、测试说明、测试记录、软件问题报告单对回归测试的工作进行总结，编写软件部件回归测试报告、测试问题报告，并对软件部件回归测试的执行活动、测试记录、软件部件回归测试报告和测

试问题报告进行评审。具体要求见6.9.4。

9.4.8 文档

软件部件回归测试完成后形成的文档有：

a) 软件单元回归测试计划

b) 软件单元回归测试说明

c) 软件单元回归测试报告

d) 软件单元回归测试记录

e) 软件单元回归测试问题报告

f) 软件部件回归测试计划

g) 软件部件回归测试说明

h) 软件部件回归测试报告

i) 软件部件回归测试记录

j) 软件部件回归测试问题报告

可根据需要对上述文档及文档的内容进行裁剪。裁剪要求见 4.8。

上述文档也可分别作为软件单元测试和软件部件测试产生的文档的补充件。

9.5 配置项回归测试

9.5.1 测试组织和管理

一般由软件的供方组织软件配置项回归测试，可由供方实施，或交独立的测试机构实施。对供方实施的回归测试，测试管理应纳入软件开发过程中；对独立的测试机构，测试管理按照7.2实施。

9.5.2 技术要求

软件配置项回归测试的技术要求应符合以下原则：

a) 对更改的软件单元的测试，应符合原软件单元测试的技术要求，可根据更改情况进行裁剪；

b) 对受更改影响的软件部件的测试，应符合原软件部件测试的技术要求，可根据受影响情况进行裁剪；

c) 对软件配置项的测试，应符合原软件配置项测试的技术要求，可根据受影响情况进行裁剪；

d) 当回归测试结果和原软件单元测试、软件部件测试和软件配置项测试的正确结果不一致时，应对出现问题的软件单元、受该单元影响的软件部件和软件配置项重新进行回归测试。

9.5.3 测试内容

软件配置项回归测试的测试内容分三种情况考虑：

a） 对更改的软件单元的测试可能存在以下三种情况：一是仅重复测试原软件单元测试做过的测试内容；二是修改原软件单元测试做过的测试内容；三是在前两者的基础上增加新的测试内容；

b） 对受更改影响的软件部件的测试。测试分析员应分析更改对软件部件的影响域，并据此确定回归测试内容。可能存在以下三种情况：一是仅重复测试与更改相关的、并已在原软件部件测试中做过的测试内容；二是修改与更改相关的、并已在原软件部件测试中做过的测试内容；三是在前两者的基础上增加新的测试内容；

c） 对软件配置项的测试。测试分析员应分析更改对软件配置项的影响域，并据此确定回归测

试内容。可能存在以下三种情况：一是仅重复测试与更改相关的、并已在原软件配置项测试中做过的测试内容；二是修改与更改相关的、并已在原软件配置项测试中做过的测试内容；三是在前两者的基础上增加新的测试内容。

9.5.4 测试环境

软件配置项回归测试的测试环境要求分三种情况：

a）对更改的软件单元的测试，其测试环境要求应与原软件单元测试的测试环境要求一致；

b）对受更改影响的软件部件的测试，其测试环境要求应与原软件部件测试的测试环境要求一致；

c）软件配置项测试的测试环境要求应与原软件配置项测试的测试环境要求一致。

9.5.5 测试方法

软件配置项回归测试不排除使用标准测试集和经认可的系统功能测试方法。本标准描述的测试方法是重复软件配置项开发各阶段的相关工作的方法。这种方法分三种情况：

a）对更改的软件单元的测试。当未增加新的测试内容时，对更改的软件单元的测试采用原软件单元测试的测试方法；否则，根据情况选择适当的测试方法；

b）对受更改影响的软件部件的测试。当未增加新的测试内容时，对受影响的软件部件的测试采用原软件部件测试的测试方法；否则，根据情况选择适当的测试方法；

c）对软件配置项的测试。当未增加新的测试内容时，对软件配置项的测试采用原软件配置项测试的测试方法；否则，根据情况选择适当的测试方法。

9.5.6 结束条件

软件配置项回归测试的结束条件用来评价回归测试的工作是否达到要求，一般应符合以下原则：

a) 按照软件单元回归测试的要求（见 9.3），完成了对更改的软件单元的测试，并且无新问题出现；

b) 按照软件部件回归测试的要求（见 9.4），完成了对受更改影响的软件部件的测试，并且无新问题出现；

c) 对软件配置项的测试符合原软件配置项测试的结束条件，并且无新问题出现。

另外，软件配置项回归测试的文档应齐全、符合规范。

9.5.7 测试过程

软件配置项回归测试的测试过程按顺序包括下面几步：

a）按照9.3.7的内容对更改的软件单元进行测试；

注：更改的软件单元通过测试后，才能对有关的软件部件进行测试。

b） 按照第9.4.7条的第b）～e）步对受影响的软件部件进行测试；

注：软件部件通过测试后，才能对软件配置项进行测试。

c）测试分析员根据测试问题报告、软件更改报告单，分析软件配置项回归测试的范围，确定原软件配置项测试的充分性要求、终止要求、资源要求、软件特性、测试技术和方法的适用程度，并酌情更改，确定回归测试的测试进度，按照GB/T9386完成软件配置项回归测试计划。对软件配置项回归测试计划进行评审，评审要求见7.9.1中软件配置项测试计划的

评审；

d）测试设计员和测试程序员根据软件配置项回归测试计划确定测试用例，或从原软件配置项测试说明中选择测试用例、或修改原有测试用例、或设计新的测试用例，补充相应的测试数据、测试资源和测试软件，建立相应的测试环境，确定相应的测试顺序，按照GB/T9386编写软件配置项回归测试说明。对软件配置项回归测试说明进行评审，评审要求见7.9.2中软件配置项测试说明的评审；

e）测试员和测试分析员按照软件配置项回归测试说明对软件配置项进行测试，要求见7.9.3；

f） 测试分析员根据原测试问题报告、原软件更改报告单，软件配置项回归测试计划、测试说明、测试记录、软件问题报告单对回归测试的工作进行总结，编写软件配置项回归测试报告和测试问题报告，并对软件配置项回归测试的执行活动、测试记录、软件配置项回归测试报告和测试问题报告进行评审。具体要求见7.9.4。

9.5.8 文档

软件配置项回归测试完成后形成的文档有：

a) 软件单元回归测试计划
b) 软件单元回归测试说明
c) 软件单元回归测试报告
d) 软件单元回归测试记录
e) 软件单元回归测试问题报告
f) 软件部件回归测试计划
g) 软件部件回归测试说明
h) 软件部件回归测试报告
i) 软件部件回归测试记录
j) 软件部件回归测试问题报告
k) 软件配置项回归测试计划
l) 软件配置项回归测试说明
m) 软件配置项回归测试报告
n) 软件配置项回归测试记录
o) 软件配置项回归测试问题报告

可根据需要对上述文档及文档的内容进行裁剪。裁剪要求见 4.8。

上述文档也可分别作为软件单元测试、软件部件测试和软件配置项测试产生的文档的补充件。

9.6 系统回归测试

9.6.1 测试组织和管理

一般应由软件的需方或供方组织系统回归测试，可由供方实施，或交独立的测试机构实施。对供方实施的回归测试，测试管理应纳入软件开发过程中；对独立的测试机构，测试管理按照 8.2 实施。

9.6.2 技术要求

系统回归测试的技术要求一般应符合以下原则：

a）对更改的软件单元的测试，应符合原软件单元测试的技术要求，可根据更改情况进行裁剪；

b）对受更改影响的软件部件的测试，应符合原软件部件测试的技术要求，可根据受影响情况进行裁剪；

c）对受更改影响的软件配置项的测试，应符合原软件配置项测试的技术要求，可根据受影响情况进行裁剪；

d）对系统的测试，应符合原系统测试的技术要求，可根据受影响情况进行裁剪；

e）当回归测试结果和原软件单元测试、软件部件测试、软件配置项测试和系统测试的正确结果不一致时，应对出现问题的软件单元和受该单元影响的软件部件、软件配置项和系统重新进行回归测试。

9.6.3 测试内容

系统回归测试的测试内容分四种情况考虑：

a）对更改的软件单元的测试可能存在以下三种情况：一是仅重复测试在原软件单元测试中做过的测试内容；二是修改在原软件单元测试中做过的测试内容；三是在前两者的基础上增加新的测试内容；

b）对受更改影响的软件部件的测试。测试分析员应分析更改的软件单元对软件部件的影响域，并据此确定回归测试内容。可能存在以下三种情况：一是仅重复测试与更改相关的、并在原软件部件测试中做过的测试内容；二是修改与更改相关的、并在原软件部件测试中做过的测试内容；三是在前两者的基础上增加新的测试内容；

c）对受更改影响的软件配置项的测试。测试分析员应分析更改对软件配置项的影响域，并据此确定回归测试内容。可能存在以下三种情况：一是仅重复测试与更改相关的、并在原软件配置项测试中做过的测试内容；二是修改与更改相关的、并在原软件配置项测试中做过的测试内容；三是在前两者的基础上增加新的测试内容；

d）对系统的测试。测试分析员应分析软件系统受更改影响的范围，并据此确定回归测试内容。可能存在以下三种情况：一是仅重复测试与更改相关的、并在原系统测试中做过的测试内容；二是修改与更改相关的、并在原系统测试中做过的测试内容；三是在前两者的基础上增加新的测试内容。

9.6.4 测试环境

系统回归测试的测试环境要求分四种情况：

a）对更改的软件单元的测试，其测试环境要求应与原软件单元测试的测试环境要求一致；

b）对受更改影响的软件部件的测试，其测试环境要求应与原软件部件测试的测试环境要求一致；

c）对受更改影响的软件配置项的测试，其测试环境要求应与原软件配置项测试的测试环境要求一致；

d）系统测试的测试环境要求应与原系统测试的测试环境要求一致。

9.6.5 测试方法

系统回归测试不排除使用标准测试集和经认可的系统功能测试方法。本标准描述的测试方法是重复软件系统开发各阶段的相关工作的方法。这种测试方法分四种情况：

a）对更改的软件单元的测试。当未增加新的测试内容时，对更改的软件单元的测试采用原软件单元测试的测试方法；否则，根据情况选择适当的测试方法；

b）对受更改影响的软件部件的测试。当未增加新的测试内容时，对受影响的软件部件的测试采用原软件部件测试的测试方法；否则，根据情况选择适当的测试方法；

c）对受更改影响的软件配置项的测试。当未增加新的测试内容时，对受更改影响的软件配置项的测试采用原软件配置项测试的测试方法；否则，根据情况选择适当的测试方法；

d）对系统的测试。当未增加新的测试内容时，系统测试采用原系统测试方法；否则，根据情况选择适当的测试方法。

9.6.6 结束条件

系统回归测试的结束条件用来评价回归测试的工作是否达到要求，一般应应符合以下原则：

a）按照软件单元回归测试的要求（见 9.3），完成了对更改的软件单元的测试，并且无新问题出现；

b）按照软件部件回归测试的要求（见 9.4），完成了对受更改影响的软件部件的测试，并且无新问题出现；

c）按照软件配置项回归测试的要求（见 9.5），完成了对受更改影响的软件配置项的测试，并且无新问题出现；

d）对系统的测试符合原系统测试的结束条件，并且无新问题出现。

另外，系统回归测试的文档应齐全、符合规范。

9.6.7 测试过程

系统回归测试的测试过程按顺序包括下面几步：

a） 按照9.3.7的内容对更改的软件单元进行测试；

注：更改的软件单元通过测试后，才能对受影响的软件部件进行测试。

b） 按照第9.4.7条的第b）～e）步对受更改影响的软件部件进行测试；

注：受影响的软件部件通过测试后，才能对受影响的软件配置项进行测试。

c） 按照第9.5.7条的第c）～f）步对受更改影响的软件配置项进行测试；

注：受影响的软件配置项通过测试后，才能进行系统测试

d） 测试分析员根据测试问题报告、软件更改报告单，分析系统测试的范围，确定原系统测试的充分性要求、终止要求、资源要求、软件特性、测试技术和方法的适用程度，并酌情更改，确定回归测试的测试进度，按照GB/T9386完成系统回归测试计划。对系统回归测试计划进行评审，评审要求见8.9.1中系统测试计划的评审；

e） 测试设计员和测试程序员根据系统回归测试计划确定测试用例，可从原系统测试说明中选择测试用例、或修改原有测试用例、或设计新的测试用例，补充相应的测试数据、测试资源和测试软件，建立相应的测试环境，确定相应的测试顺序，按照GB/T9386编写系统回归测试说明。对系统回归测试说明进行评审，评审要求见8.9.2中系统测试说明的评审；

f） 测试员和测试分析员按照系统回归测试说明对系统进行测试，要求见8.9.3；

g） 测试分析员根据原测试问题报告、原软件更改报告单，系统回归测试计划、测试说明、测试记录、软件问题报告单对系统回归测试的工作进行总结，编写系统回归测试报告、测试

问题报告，并对系统回归测试的执行活动、测试记录、系统回归测试报告和测试问题报告进行评审。具体要求见8.9.4。

9.6.8 文档

系统回归测试完成后形成的文档有：

a) 软件单元回归测试计划
b) 软件单元回归测试说明
c) 软件单元回归测试报告
d) 软件单元回归测试记录
e) 软件单元回归测试问题报告
f) 软件部件回归测试计划
g) 软件部件回归测试说明
h) 软件部件回归测试报告
i) 软件部件回归测试记录
j) 软件部件回归测试问题报告
k) 软件配置项回归测试计划
l) 软件配置项回归测试说明
m) 软件配置项回归测试报告
n) 软件配置项回归测试记录
o) 软件配置项回归测试问题报告
p) 系统回归测试计划
q) 系统回归测试说明
r) 系统回归测试报告
s) 系统回归测试记录
t) 系统回归测试问题报告

可根据需要对上述文档及文档的内容进行裁剪。裁剪要求见 4.8。

上述文档也可分别作为软件单元测试、软件部件测试、软件配置项测试和系统测试产生的文档的补充件。

附 录 A
（资料性附录）
软件测试方法

A.1 静态测试方法

A.1.1 代码审查

代码审查的测试内容：检查代码和设计的一致性；检查代码执行标准的情况；检查代码逻辑表达的正确性；检查代码结构的合理性；检查代码的可读性。

代码审查的组织：由四人以上组成，分别为组长、资深程序员、程序编写者与专职测试人员。组长不能是被测试程序的编写者，组长负责分配资料、安排计划、主持开会、记录并保存被发现的错误。

代码审查的过程：

a) 准备阶段:组长分发有关材料，被测程序的设计和编码人员向审查组详细说明有关材料，并回答审查组成员所提出的有关问题。

b) 程序阅读：审查组人员仔细阅读代码和相关材料，对照代码审查单，记录问题及明显缺陷。

c) 会议审查：组长主持会议，程序员逐句阐明程序的逻辑，其他人员提出问题，利用代码审查单进行分析讨论，对讨论的各个问题形成结论性意见。

d) 形成报告：会后将发现的错误形成代码审查问题表，并交给程序开发人员。对发现错误较多或发现重大错误，在改正错误之后再次进行会议审查。

以下是一个推荐的代码审查单，可以根据实际工作经验和具体被测程序对以下内容进行增删：

a) 寄存器使用：

1) 如果需要一个专用寄存器，指定了吗？
2) 宏扩展或子程序调用使用了已使用着的寄存器而未保存数据吗？
3) 默认使用的寄存器的值正确吗？

b) 格式：

1) 嵌套的 IF 是否已正确地缩进？
2) 注释准确并有意义吗？
3) 是否使用了有意义的标号？
4) 代码是否基本上与开始时的模块模式一致？
5) 是否遵循全套的编程标准？

c) 入口和出口连接：

1) 初始入口的最终出口正确吗？
2) 对另一模块的每一次调用：
全部所需的参数是否已传送给每一个被调用的模块？
被传送的参数值是否正确的设置？
栈状态和指针状态是否正确？

d) 程序语言的使用：

1) 模块中是否使用语言完整定义的有限子集？

2） 未使用内存的内容是否影响系统安全？处理是否得当？

e） 存储器使用：

1） 每一个域在第一次使用前正确地初始化了吗？

2） 规定的域正确吗？

3） 每个域是否由正确的变量类型声明？

4） 存储器重复使用吗？可能产生冲突吗？

f） 测试和转移：

1） 是否进行了浮点相等比较？

2） 测试条件正确吗？

3） 用于测试的变量正确吗？

4） 每个转换目标正确并至少执行一次？

5） 三种情况（大于 0，小于 0，等于 0）是否已全部测试？

g） 性能：

1） 逻辑是否被最佳地编码？

2） 提供的是一般的错误处理还是异常的例程？

h） 可维护性：

1） 所提供的列表控制是否有利于提高可读性？

2） 标号和子程序名符合代码的意义吗？

i） 逻辑：

1） 全部设计是否均已实现？

2） 编码是否做了设计所规定的内容？

3） 每个循环是否执行正确的次数？

4） 输入参数的所有异常是否已被接收？

j） 软件多余物：

1） 是否有不可能执行到的代码？

2） 是否有即使不执行也不影响程序功能的指令？

3） 是否有未引用的变量、标号和常量？

4） 是否有多余的程序单元？

代码审查问题表应写明所查出的错误类型、错误类别、错误严重程度、错误位置、错误原因。错误类型有文档错误、编程语言错误、逻辑错误、接口错误、数据使用错误、编程风格不当、软件多余物。错误类别有遗漏、错误、多余。

这种静态测试方法是一种多人一起进行的测试活动，要求每个人尽量多提出问题，同时讲述程序者也会突然发现一些问题，这时要放慢进度，把问题分析出来。

A.1.2 代码走查

代码走查的测试内容与代码审查的基本一样。

代码走查的组织：由四人以上组成，分别为组长、秘书、资深程序员与专职测试人员。走查人员不能是被测试程序的编写者，组长负责分配资料、安排计划、主持开会，秘书记录被发现的错误。

代码走查的过程：

a） 准备阶段:组长分发有关材料，走查组详细阅读材料和认真研究程序。

b） 生成实例：走查小组人员提出一些有代表性的测试实例。

c) 会议走查：组长主持会议，其他人员对测试实例用头脑来执行程序，也就是测试实例沿程序逻辑走一遍，并由测试人员讲述程序执行过程，在纸上或黑板上监视程序状态，秘书记录下发现的问题。

d) 形成报告：会后将发现的错误形成报告，并交给程序开发人员。对发现错误较多或发现重大错误，在改正错误之后再次进行会议走查。

这种静态测试方法是一种多人一起进行的测试活动，要求每个人尽量多提供测试实例，这些测试实例是作为怀疑程序逻辑与计算错误的启发点，在随测试实例游历程序逻辑时，在怀疑程序的过程中发现错误。这种方法不如代码审查检查的范围广，错误覆盖全。

A.1.3 静态分析

静态分析一般包括控制流分析、数据流分析、接口分析、表达式分析。此外，静态分析还可以完成下述工作。

a）提供间接涉及程序缺陷的信息：

1）每一类型语句出现的次数；

2）所有变量和常量的交叉引用表；

3）标识符的使用方式；

4）过程的调用层次；

5）违背编码规则；

6）程序结构图和程序流程图；

7）子程序规模、调用/被调用关系、扇入/扇出数。

b）进行语法/语义分析，提出语义或结构要点，供进一步分析。

c）进行符号求值。

d）为动态测试选择测试用例进行预处理。

静态分析常需要使用软件工具进行。静态分析是在程序编译的过程中，在静态测试之前进行的。

A.1.3.1 控制流分析

控制流分析是使用控制流程图系统地检查被测程序的控制结构的工作。控制流按照结构化程序规则和程序结构的基本要求进行程序结构检查。这些要求是被测程序不应包含：

a）转向并不存在的语句标号；

b）没有使用的语句标号；

c）没有使用的子程序定义；

d）调用并不存在的子程序；

e）从程序入口进入后无法达到的语句；

f） 不能达到停止语句的语句。

控制流程图是一种简化的程序流程图，控制流程图由“节点”和“弧”两种图形符号构成。

A.1.3.2 数据流分析

数据流分析是用控制流程图来分析数据发生的异常情况，这些异常包括被初始化、被赋值或被引用过程中行为序列的异常。数据流分析也作为数据流测试的预处理过程。

数据流分析首先建立控制流程图，然后在控制流程图中标注某个数据对象的操作序列，遍历控制流程图，形成这个数据对象的数据流模型，并给出这个数据对象的初始状态，利用数据流异常状态图分析数据对象可能的异常。

数据流分析可以查出引用未定义变量、对以前未使用的变量再次赋值等程序错误或异常情况。

A.1.3.3 接口分析

接口分析主要用在程序静态分析和设计分析。接口一致性的设计分析涉及模块之间接口的一致性以及模块与外部数据库之间的一致性。程序的接口分析涉及子程序以及函数之间的接口一致性，包括检查形参与实参的类型、数量、维数、顺序以及使用的一致性。

A.1.3.4 表达式分析

表达式错误主要有以下几种：

括号使用不正确，数组引用错误，作为除数的变量可能为零，作为开平方的变量可能为负，作为正切值的变量可能为π /2，浮点数变量比较时产生的错误。

A.2 动态测试方法

A.2.1 概述

动态测试是建立在对程序的执行过程中，根据是否对被测对象内部的了解，分为黑盒测试和白盒测试。

黑盒测试又称功能测试、数据驱动测试或基于规格说明的测试，这种测试不必了解被测对象的内部情况，而依靠需求规格说明中的功能来设计测试用例。

白盒测试又称结构测试、逻辑测试或基于程序的测试，这种测试应了解程序的内部构造，并且根据内部构造设计测试用例。

在单元测试时一般采用白盒测试，在配置项测试或系统测试时一般采用黑盒测试。

A.2.2 黑盒测试方法

A.2.2.1 功能分解

功能分解是将需求规格说明中每一个功能加以分解，确保各个功能被全面地测试。功能分解是一种较常用的方法。

步骤如下：

a) 使用程序设计中的功能抽象方法把程序分解为功能单元；

b) 使用数据抽象方法产生测试每个功能单元的数据。

功能抽象中程序被看成一种抽象的功能层次，每个层次可标识被测试的功能，层次结构中的某一功能有由其下一层功能定义。按照功能层次进行分解，可以得到众多的最低层次的子功能，以这些子功能为对象,进行测试用例设计。

数据抽象中，数据结构可以由抽象数据类型的层次图来描述，每个抽象数据类型有其取值集合。程序的每一个输入和输出量的取值集合用数据抽象来描述。

A.2.2.2 等价类划分

等价类划分是在分析需求规格说明的基础上，把程序的输入域划分成若干部分，然后在每部分中选取代表性数据形成测试用例。

步骤如下：

a) 划分有效等价类：对规格说明是有意义、合理的输入数据所构成的集合。

b) 划分无效等价类：对规格说明是无意义、不合理的输入数据所构成的集合。

c) 为每一个等价类定义一个唯一的编号。

d) 为每一个等价类设计一组测试用例，确保覆盖相应的等价类。

A.2.2.3 边界值分析

边界值分析是针对边界值进行测试的。使用等于、小于或大于边界值的数据对程序进行测试

的方法就是边界值分析方法。

步骤如下：

a）通过分析规格说明，找出所有可能的边界条件。

b）对每一个边界条件，给出满足和不满足边界值的输入数据。

c）设计相应的测试用例。

对满足边界值的输入可以发现计算错误，对不满足的输入可以发现域错误。该方法会为其它测试方法补充一些测试用例，绝大多数测试都会用到本方法。

A.2.2.4 判定表

判定表由四部分组成：条件桩，条件条目，动作桩，动作条目。任何一个条件组合的取值及其相应要执行的操作构成规则，条目中的每一列是一条规则。

条件引用输入的等价类，动作引用被测软件的主要功能处理部分，规则就是测试用例。

建立并优化判定表，把判定表中每一列表示的情况写成测试用例。

该方法的使用有以下要求：

a）规格说明以判定表形式给出，或是很容易转换成判定表。

b）条件的排列顺序不会影响执行哪些操作。

c）规则的排列顺序不会影响执行哪些操作。

d）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。

e）如果某一规则的条件得到满足，将执行多个操作，这些操作的执行与顺序无关。

A.2.2.5 因果图

因果图方法是通过画因果图，把用自然语言描述的功能说明转换为判定表，然后为判定表的每一列设计一个测试用例。

步骤如下：

a） 分析程序规格说明，引出原因（输入条件）和结果（输出或）并给每个原因和结果赋予一个标识符。

b） 分析程序规格说明中语义的内容，并将其表示成连接各个原因和各个结果的“因果图”。

c） 在因果图上标明约束条件。

d） 通过跟踪因果图中的状态条件，把因果图转换成有限项的判定表。

e） 把判定表中每一列表示的情况生成测试用例。

如果需求规格说明中含有输入条件的组合，宜采用本方法。有些软件的因果图可能非常庞大，以致于根据因果图得到的测试用例数目非常大，此时不宜使用本方法。

A.2.2.6 随机测试

随机测试指测试输入数据是在所有可能输入值中随机选取的。测试人员只需规定输入变量的取值区间，在需要时提供必要的变换机制，使产生的随机数服从预期的概率分布。该方法获得预期输出比较困难，多用于可靠性测试和系统强度测试。

A.2.2.7 猜错法

猜错法是有经验的测试人员，通过列出可能有的错误和易错情况表，写出测试用例的方法。

A.2.2.8 正交实验法

正交实验法是从大量的实验点中挑出适量的、有代表性的点，应用正交表，合理地安排实验的一种科学的实验设计方法。

利用正交实验法来设计测试用例时，首先要根据被测软件的规格说明书找出影响功能实现的操作对象和外部因素，把它们当作因子，而把各个因子的取值当作状态，生成二元的因素分析表。

然后，利用正交表进行各因子的状态的组合，构造有效的测试输入数据集，并由此建立因果图。这样得出的测试用例的数目将大大减少。

A.2.3 白盒测试方法

A.2.3.1 控制流测试

控制流测试依据控制流程图产生测试用例，通过对不同控制结构成份的测试验证程序的控制结构。所谓验证某种控制结构即指使这种控制结构在程序运行中得到执行，也称这一过程为覆盖。以下介绍几种覆盖：

a）语句覆盖：

语句覆盖要求设计适当数量的测试用例，运行被测程序，使得程序中每一条语句至少被执行一边，语句覆盖在测试中主要发现错误语句。

b）分支覆盖：

分支覆盖要求设计适当数量的测试用例，运行被测程序，使得程序中每个真值分支和假值分支至少执行一次，分支覆盖也称判定覆盖。

c）条件覆盖：

条件覆盖要求设计适当数量的测试用例，运行被测程序，使得每个判断中的每个条件的可能取值至少满足一次。

d）条件组合覆盖：

条件组合覆盖要求设计适当数量的测试用例，运行被测程序，使得每个判断中条件的各种组合至少出现一次，这种方法包含了“分支覆盖”和“条件覆盖”的各种要求。

e）路径覆盖：

路径覆盖要求设计适当数量的测试用例，运行被测程序，使得程序沿所有可能的路径执行，较大程序的路径可能很多，所以在设计测试用例时，要简化循环次数。

以上各种覆盖的控制流测试步骤为：

a）将程序流程图转换成控制流图；

b）经过语法分析求得路径表达式；

c）生成路径树；

d）进行路径编码；

e）经过译码得到执行的路径；

f）通过路径枚举产生特定路径的测试用例。

A.2.3.2 数据流测试

数据流测试是用控制流程图对变量的定义和引用进行分析，查找出未定义的变量或定义了而未使用的变量，这些变量可能是拼错的变量、变量混淆或丢失了语句。数据流测试一般使用工具进行。

数据流测试通过一定的覆盖准则，检查程序中每个数据对象的每次定义、使用和消除的情况。

数据流测试步骤：

a） 将程序流程图转换成控制流图；

b） 在每个链路上标注对有关变量的数据操作的操作符号或符号序列；

c） 选定数据流测试策略；

d） 根据测试策略得到测试路径；

e） 根据路径可以获得测试输入数据和测试用例。

动态数据流异常检查在程序运行时执行，获得的是对数据对象的真实操作序列，克服了静态

分析检查的局限，但动态方式检查是沿与测试输入有关的一部分路径进行的，检查的全面性和程序结构覆盖有关。

A.2.3.3 程序变异

是一种错误驱动测试，是为了查出被测软件在做过其它测试后还剩余一些的小错误。本方法应用于测试工具。

A.2.3.4 程序插装

程序插装是向被测程序中插入操作以实现测试目的方法。程序插装不应该影响被测程序的运行过程和功能。

有很多的工具有程序插装功能。由于数据记录量大，手工进行将是一件很烦琐的事。

A.2.3.5 域测试

域测试是要判别程序对输入空间的划分是否正确。该方法限制太多，使用不方便，供有特殊要求的测试使用。

A.2.3.6 符号求值

符号求值是允许数值变量取“符号值”以及数值。符号求值可以检查公式的执行结果是否达到程序预期的目的；也可以通过程序的符号执行，产生出程序的路径，用于产生测试数据。符号求值最好使用工具，在公式分支较少时手工推导也是可行的。

附 录 B
（资料性附录）
软件可靠性的推荐模型

B.1 斯奈德蕴德模型

B.1.1 斯奈德蕴德模型的目标

这个模型的目标是预测软件产品的下列属性：

a）将在给定时间内（执行时间、工作时间或日历时间）发生的失效数；

b）在该软件生存期间会发生的失效的最大数；

c）将在给定时间之后发生的失效的最大数；

d）发生规定的失效数所需的时间；

e）经过给定时间所纠正的故障数；

f）纠正给定数目的故障所需的时间；

g）在给定时刻未解决的（发现了但未纠正）故障数；

h）纠正给定数目未解决的故障所需增加的时间；

i）使未解决的故障达到给定数值所需的时间。

这个模型的基本原理是，失效的检测过程随着测试继续进行而变化。此外，在预测未来方面，近期的失效统计通常比早先的失效统计更有用。对失效统计的数据（即每时间单位查出的失效数）有三种利用方法。假设有 m 个测试时间段，并且在第 i 段查到 f_i 个失效，那么就能用下列方法之一进行处理：

方法一：利用这 m 个时间段的所有失效。

方法二：完全忽略前 s-1 个时间段（2≤s≤m）的失效统计数，只用从 s 到 m 个时间段的数据。

方法三：使用从 1 到 s-1 个时间段的累积失效统计数 F_{s-1}，即：

$$F_{s-1}=\sum_{i=1}^{s-1} f_i \qquad \text{（B.1）}$$

当人们认为所有时间段的失效统计数在预计未来的失效统计数中都有用时，应用第一种方法。当人们认为失效检查过程已发生显著变化，因而只有最后的 m-s+1 个时间段在未来的失效预测中有用时，就要用第二种方法。最后一种方法介于上述两种方法之间，这时人们认为前 s-1 个时间段的综合失效统计数与其余时间段的单独统计数对未来预测的失效和检测行为都是有代表性的。

B.1.2 斯奈德蕴德模型的假设

斯奈德蕴德模型的假设是：

a）在一个时间段内查出的失效数与其它时间段的失效统计数无关；

b）只统计新的失效；

c）故障纠正率与待纠正的故障数成正比；

d）软件的运行方式与预期的运行使用方式相似；

e）查出的平均失效数从一个时间段到下一个时间段逐步减少；

f）所有的时间段长度相同；

g）失效检测率正比于测试时程序中的故障数。假设失效检测过程是非齐次泊松过程，其实效检测率呈指数下降，第 i 个时间段失效检测率 d_i 表示为：

$$d_i = \alpha \exp(-\beta i) \qquad \text{（B.2）}$$

式中 $\alpha > 0, \beta > 0$，都是模型的常数。

B.1.3 斯奈德蕴德模型的构造

模型中使用了两个参数：α 是在时间 m=0 的失效率，β 是对在时间段内的失效率有影响的比例常数。在这些估值中：m 是最后的观察统计时间段；s 是时间段的标志；X_k 是在第 k 个时间段内发现的失效数；X_{s-1} 是从第 1 到第 s-1 个时间段内发现的失效数；$X_{s,m}$ 是从第 s 到第 m 个时间段发现的失效数；并且 $X_m = X_{s-1} + X_{s,m}$。将似然函数展开为

$$\log L = X_m[\log X_m - 1 - \log(1 - \exp(-\beta m))]$$
$$+ X_{s-1}[\log(1 - \exp(-\beta(s-1)))]$$
$$+ X_{s,m}[\log(1 - \exp(-\beta))]$$
$$- \beta \sum_{k=0}^{s-2} (s-1-k) X_{s-1-k}$$

这个函数用来为前述三种方法推导估计 α 和 β 的公式。在下列公式中 α 和 β 是总体参数的估计值。

参数估计：方法 1

使用从 1 到 m 所有时间段（即 s=1）的全部失效统计数。下列两个公式分别用来估计 α 和 β。

$$\frac{1}{\exp(\beta) - 1} - \frac{m}{\exp(\beta m) - 1} = \sum_{k=0}^{m-1} k \frac{X_{k+1}}{X_m} \qquad \text{（B.3）}$$

$$\alpha = \frac{\beta X_m}{1 - \exp(-\beta m)} \qquad \text{（B.4）}$$

参数估计：方法 2

只用 s 到 m 时间段的失效统计数（即 1≤s≤m），下列两个公式分别用来估计 α 和 β。

$$\frac{1}{\exp(\beta) - 1} - \frac{m - s + 1}{\exp(\beta(m - s + 1)) - 1} = \sum_{k=0}^{m-s} k \frac{X_{k+s}}{X_{s,m}} \qquad \text{（B.5）}$$

$$\alpha = \frac{\beta X_{s,m}}{1-\exp(-\beta(m-s+1))} \tag{B.6}$$

参数估计：方法 3

使用从 1 到 s-1 时间段的累积失效统计数和 s 到 m（即 2≤s≤m）时间段内各段的单独失效统计数。这个方法介于方法 1（使用全部数据）和方法 2（放弃“老”数据）之间。下列两个公式分别用来估计 α 和 β。

$$\frac{(s-1)X_{s-1}}{\exp(\beta(s-1))-1} + \frac{X_{s,m}}{\exp(\beta)-1} - \frac{mX_m}{\exp(\beta m)-1} = \sum_{k=0}^{m-s}(s+k-1)X_{s+k} \tag{B.7}$$

$$\alpha = \frac{\beta X_m}{1-\exp(-\beta m)} \tag{B.8}$$

均方差（MSE）准则能用来求得 s 的最佳值，从而在三种方法中选择一种。MSE 计算在 s≤i≤m 范围内模型预计值与实际累积的失效统计数 x(i)之间的方差和。下列公式适用于上述方法 2。对于方法 1 和方法 3，s=1。

$$MSE = \frac{\sum_{i=s}^{m}[\alpha/\beta(1-\exp(-\beta(i-s+1)))-x(i)]^2}{m-s+1} \tag{B.9}$$

这样，对于每个 s 值，用上式计算 MSE。选择使 MSE 最小的 s 值。结果得到对于数据集来说最佳的三个值（β, α, s），然后对数据运用合适的方法。

B.1.4 斯奈德蕴德模型的缺点

斯奈德蕴德模型的缺点主要表现在：

a）它不考虑不同时间段的失效可能的相关性；

b）它不计失效的重复；

c）它使用等长的时间段；

d）它不考虑随着软件的修改可能导致失效增长的可能性。

这些缺点可以通过将软件分解成多个版本而得到改进，这些版本表示原有版本加上若干修改。对于可靠性预计来说，每个版本代表一个不同的模块，用该模型预计每个模块的可靠性。

B.1.5 斯奈德蕴德模型的数据需求

仅有的数据需求是每个测试时段的错误数 f_i，$i=1,\cdots,m$。

虽然不要求有一个数据库，但出于下面所述的几条理由，建立并保持一个可靠性数据库是非常有用的：为了能在必要时重新使用输入数据集；为能对各种不同项目进行可靠性预计和评估；为能将这些项目的预计可靠性与实际可靠性进行对比。这个数据库使模型用户能进行以下几种分析：看模型执行得好坏；将不同项目的可靠性进行对比，以观察是否有开发因素对形成可靠性有影响；分析一个给定项目或从一个项目到另一个项目可靠性是否随着时间推移而得到改进。

B.1.6 斯奈德蕴德模型的应用

下面为模型的应用。这些应用都是该模型的一些单独而又相关的运用，它们共同组成一个综合的可靠性大纲。

a）预测：预测未来的失效数、故障纠正数和 B.1.7 所描述的有关量。

b）控制：将预测结果与预定目标进行比较，并标明不能满足目标要求的软件。

c）评估：确定对不满足目标要求的软件采取什么措施（例如，加强审查，加强测试，重新设计软件，改进过程）。明确表示测试策略也是评估的一部分；测试策略的表达涉及确定：优先级、测试延续时间和完成日期、测试的人员和计算机资源的分配。

B.1.7 可靠性预测

利用 B.1.3 给出的最佳的三个值（β,α,s）可计算不同的可靠性预测。下面给出了 B.1.1 中方法二的计算公式，其中 T≥s。对于方法一和方法三，s=1 且 T≥1，其中 T 推荐用执行时间，但也能用工作时间或日历时间。

a）设当前时间为 t，已发现的失效数为 X(t)，则检测到总数为 F 的失效数所需的时间：

$$T_f(t)=\log[\alpha/(\alpha-\beta(F(t)-X(t)))/\beta]-(t-s+1) \qquad (B.10)$$

b）对于 $\alpha>\beta(F(t)+X(t))$：

1）预测的 T 时间后的失效数：

$$F(T)=(\alpha/\beta)[1-\exp(-\beta(T-s+1))] \qquad (B.11)$$

2）最大的失效数（T=∞）：

$$F(\infty)=\alpha/\beta \qquad (B.12)$$

3）在 t 时刻，已发现 X(t)个失效数之后，预测的最大遗留失效数：

$$RF(t)=\alpha/\beta-X(t) \qquad (B.13)$$

4）T 时间后纠正的故障数：

$$C(T)=(\alpha/\beta)[1-\exp(-\beta((T-s+1)-\Delta t))] \qquad (B.14)$$

其中 Δt 是发现失效后纠正故障的平均滞后时间。

5）纠正 C 个故障所需的时间：

$$T_c=\Delta t+[(\log[\alpha/(\alpha-\beta C)])/\beta]+s-1 \qquad (B.15)$$

c）对于 $\alpha>\beta C$：

1）在 T 时刻遗留未解决的故障数：

$$N(T)=F(T)-C(T) \qquad (B.16)$$

2）未解决的故障的纠正时间：

$$\Delta T_N=[\log((N\beta\exp(\beta(T-s+1))/\alpha)+1)]/\beta \qquad (B.17)$$

其中 N 是从 T 时刻起要纠正的故障数。

3）未解决的故障时间：

预计的未解决的故障数达到 N 所需的时间是：

$$T_N=[(\log[(\alpha\exp(\beta\Delta T_N)-1)/\beta N])/\beta]+s-1 \qquad (B.18)$$

B.2 广义指数模型

B.2.1 广义指数模型的目标

许多通用的软件可靠性模型都产生相似的结果。广义指数模型的基本思想是，用一组公式来

表示有指数危险的若干模型，从而简化建模过程。

广义指数模型包含若干众所周知的软件可靠性模型的概念。主要概念是，失效发生率正比于软件中残留的故障数。其次，在两次失效发现之间失效率保持恒定，且每个软件故障被排除之后失效率降低一个相同的量。这样，纠正每个故障对减少软件危险都有相同的影响。这个模型的目标是将几个众所周知的模型表归纳为一个形式，它能用来预测：

a）经过给定的时间将发生的失效数；

b）软件生存期内发生失效的最大数；

c）在给定时间之后将发生的失效的最大数；

d）发生给定失效数所需的时间；

e）在给定时间以前所纠正的故障数；

f）纠正给定数目的故障所需的时间。

B.2.2 广义指数模型的假设

广义指数模型的基本假设是：

a）失效率正比于程序当前含有的故障数；

b）所有失效均有相等的可能性发生且相互独立；

c）每个失效的严重性级别都相同；

d）软件的运行方式与预期的运行使用方式相似；

e）引起失效的故障都被立即纠正，且不引入新故障。

B.2.3 广义指数模型的构造

广义指数模型的构造从简单的却比较一般的软件危险函数 $Z(X)$ 表示式开始。即：

$$Z(X) = K[E_0 - E_c(x)] \qquad \text{（B.19）}$$

其中：

X —测定项目进展的时间或资源变量。

E_0 — 程序中将引起失效的初始故障数。也可以把它看成如果测试无限地延续下去将经历的失效数。

E_c — 一旦已花费 X 单位的时间或工作量，就已发现并纠正的程序中的故障数。

K — 比例常数；每个资源单位或时间单位，每个残留故障所引起的失效数。

残留故障数 E_r 用下式表示：

$$E_r = Z(X)/K = [E_0 - E_c(x)] \qquad \text{（B.20）}$$

许多通用的模型能用上述这组公式表示，只是对参数和故障纠正函数 $E_C(x)$ 的表示各取不同的假设。表 B.1 对其中某些模型进行了概括和比较。表中每个模型都的初始开发使用了一个或更多的时间变量或资源变量。

表 B.1 适合失效率函数的广义指数，表示形式的通用可靠性模型

模型名	初始危险函数	参数等价量	注释
广义形式	$K[E_0 - E_c(x)]$		
指数模型	$K^{'}[E_0 / I_T - \varepsilon_c(x)]$	$\varepsilon_c = E_c / I_T$ $K = K^{'} / I_T$	相对于 I_T（指令数）规范化的
J-M	$\Phi[N-(i-1)]$	$\Phi = K$ $N = E_0$ $E_c = i-1$	在发现了一个错误而未纠正时适用此模型
基本模型	$\lambda_0[1-\mu/\gamma_0]$	$\lambda_0 = KE_0$ $\gamma_0 = E_0$ $\mu = E_c$	如果用相同的假设来预计 m 和 E_c，那么这个模型与指数模型一样
对数	$\lambda_0 \exp(-\varphi\mu)$	$\lambda_0 = KE_0$ $E_0 - E_c(x)$ $= E_0 \exp(-\varphi\mu)$	基本假设是残留的错误数呈指数下降

若给定 B.2.5 中定义的数据，则对表 B.1 中任何模型参数的估计就简化为一个统计参数估计问题。有三种基本方法：矩量法、最小二乘法和最大似然法。这三种方法对每种模型全部都适用。

最简单的参数估计方法是矩量法。考虑具有两个未知参数 K 和 E_0 的广义表示法。经典矩量估计技术将概率分布的 1 阶矩和 2 阶矩与相应的数据的矩量匹配。对这个过程稍加修改，在两个不同的 X 值处将 1 阶矩即均值匹配。即令试运行数为 n，以及运行失效数为 r，失效时钟表时间序列是 $t_1, t_2, \cdots, t_{n-r}$，以及无失效运行的钟表时间序列是 $T_1, T_2, \cdots, T_r$，得到：

$$Z(x) = \frac{\text{失效}(x)}{\text{时间}(x)} = \frac{n-r}{H} \qquad (B.21)$$

其中

$$H = \sum_{i=1}^{n-r} t_i + \sum_{i=1}^{r} T_i \qquad (B.22)$$

在两个不同的时刻用统一形式的公式（B.21）得到；

$$Z(x_1) = \frac{n_1 - r_1}{H_1} = K[E_0 - E_c(x_1)] \qquad (B.23)$$

$$Z(x_2) = \frac{n_2 - r_2}{H_2} = K[E_0 - E_c(x_2)] \qquad (B.24)$$

求解这两组联立方程式（B.23）和（B.24），得到参数的估计量，用 ^ 表示。

$$\hat{E}_0 = \frac{E_c(x_1) - \frac{Z(x_1)}{Z(x_2)} E_c(x_2)}{1 - \frac{Z(x_1)}{Z(x_2)}} = \frac{Z(x_2)E_c(x_1) - Z(x_1)E_c(x_2)}{Z(x_2) - Z(x_1)} \quad \text{（B.25）}$$

$$\hat{K} = \frac{Z(x_1)}{\hat{E}_0 - E_c(x_1)} = \frac{Z(x_2) - Z(x_1)}{E_c(x_1) - E_c(x_2)} \quad \text{（B.26）}$$

由于表 B.1 中的五个模型的参数都通过简单的变换而与 K 和 E_0 相关，所以（B.25）和（B.26）式与有关变换（参数等价量）一起保持有效。这些公式能用来获得所有模型的矩量估计值。

B.2.4 广义指数模型的缺点

广义指数模型的缺点主要有：

a）它不考虑每个失效也许依赖于其它失效这种可能性。

b）它假设在故障纠正过程中不引入新故障。

c）每个故障的检测在故障被纠正时可能对软件有不同影响。对数模型处理这个问题的方式是指明较早的故障纠正比以后纠正有更大的影响。

d）它不考虑由于程序演变故障能随着时间而增长的可能性，不过处理这种缺点的技术已经开发出来了。

B.2.5 广义指数模型的数据需求

测试期间要记录全部 n 次测试运行。测试结果包括 r 次成功和 n-r 次失败，并有失效发生时间，这个时间是用钟表时间和运行的执行时间测定的或者是在预测工作开始后，就日历时间来测定。此外应记录 r 次成功运行的时间。这样，需求的数据是运行总数 n、成功运行数 r、失效前钟表时间序列 $t_1, t_2, \cdots, t_{n-r}$，和无失效运行的钟表时间序列 $T_1, T_2, \cdots, T_r$。所有这些时间应是真实或仿真运行的时间；然而，只要测试时间可得到，就应当把它记录下来。对数据需要加说明，描述数据是代表运行、仿真运行，还是代表测试，描述支配输入数据的环境和条件。如果可能，应当记录类似的运行数据集。

B.2.6 广义指数模型的应用

广义指数模型的趋势都是乐观的。在运行剖面是“有正规的”，且软件排错过程得到良好控制（既故障纠正过程趋向于完备且不易出错）的条件下这类模型是适用的。

下面为模型的主要应用。这些单独而又相关的应用，共同组成综合的可靠性大纲。

a）预测：预计未来的失效时间、故障纠正和 B.2.7 所描述的有关量。

b）控制：将预计结果与预定目标进行比较，并标明不满足目标要求的软件。

c）评估：确定对不满足目标要求的软件采取什么措施（例如，加强审查，加强测试，重新设计软件，改进过程）。明确表示测试策略也是评估的一部分；它涉及确定：优先级、测试延续时间和完成日期，测试的人员和计算机资源的分配。

B.2.7 可靠性预测

除了故障总数的估计值 $\hat{E}_0$ 之外，其它估计值是：

a）排除下 m 个故障的估计时间：

$$\hat{T}_m = \sum_{j=n+1}^{n+m} \frac{1}{\hat{K}_0(\hat{E}_0 - j + 1)} \qquad \text{（B.27）}$$

b）在 τ 时刻的当前失效率估计值

$$\hat{\lambda}(\tau) = \hat{K}_0(\hat{E}_0 \exp(-\hat{K}_0\tau)) \qquad \text{（B.28）}$$

B.3 穆沙／奥库姆脱对数泊松执行时间模型

B.3.1 穆沙／奥库姆脱模型的目标

在按照特性很不一致的运行剖面进行测试的情况下，对数泊松模型特别适用。早期的故障纠正对失效强度函数的影响比以后纠正更显著，这种情况下失效强度函数趋于凸形，具有下降的斜率。于是对数泊松模型可以很适合这种情况。

如果有人还想把日历时间因素（例如：测试结束，资源管理等）与可靠性相关联，那么对数泊松模型是目前能达到这个目的唯一非指数模型。

可以对计算机利用情况、人员水平和当前的及计划的失效率进行综合权衡，以便对可靠性因素与时间及资源限制进行平衡。

如果修复过程的有效性不断降低，那么这种模型能产生无限的失效数，而故障数可能是有限的。

B.3.2 穆沙／奥库姆脱模型的假设

这个模型的特定假设是：

a）软件的运行方式与预期的运行使用方式相似。

b）各个失效相互独立。

c）失效强度随着经历的诸预期失效而呈指数下降。

注：第三条假设有两种后果：1，期望的失效数是时间的对数函数。2，模型可能报告无限的失效数。

B.3.3 穆沙／奥库姆脱模型的构造

从模型的假设可得经历τ 执行时间后的失效率函数λ (τ)：

$$\lambda(\tau) = \lambda_0 \exp[-\theta\mu(\tau)] \qquad \text{（B.29）}$$

式中λ $_0$ 是初始失效率函数，θ 是失效率下降参数，θ >0

将参数变换为：β $_0$=θ $^{-1}$，β $_1$=λ $_0$θ ，于是β $_0$ 和β $_1$ 的最大似然估计值可表示为下列方程式的解。

$$\theta = \hat{\beta}_0 - \frac{n}{1n(1+\hat{\beta}_1 t_n)} \qquad \text{（B.30）}$$

$$\theta = \frac{1}{\hat{\beta}_1}\sum_{i=1}^{n}\frac{t_i}{1+\hat{\beta}_1 t_i} - \frac{nt_n}{(1+\hat{\beta}_1 t_n)\ln(1+\hat{\beta}_1 t_n)} \qquad \text{（B.31）}$$

式中 t_n 是从开始到当前时刻累积的 CPU 时间。在这期间已发现总共 n 个失效。一旦得到β $_0$ 和β $_1$ 的最大似然估计值，就可利用这种估计值的不变特性，得到θ 和λ $_0$ 的最大似然估计值：

$$\hat{\theta}=\frac{1}{n}\ln(1+\hat{\beta}_1 t_n) \tag{B.32}$$

$$\hat{\lambda}_0=\hat{\beta}_0\hat{\beta}_1 \tag{B.33}$$

B.3.4 穆沙／奥库姆脱模型的缺点

两个缺点是：

a）各个失效也许不相互独立。

b）失效强度也许随着软件修改而上升。

B.3.5 穆沙／奥库姆脱模型的数据需求

要求的数据是下列二种之一：

a）失效间隔时间，即 Xi；

b）失效发生时间，即 t_i：

$$t_i=\sum_{j=1}^{i}X_j \tag{B.34}$$

B.3.6 穆沙／奥库姆脱模型的应用

下面为模型的主要应用。这些单独而又相关的应用共同组成综合的可靠性大纲。

a）预计：估计未来的失效时间、故障纠正和 B.3.7 所描述的有关量。

b）控制：将预计结果与预定目标进行比较，并标明不满足目标要求的软件。

c）评估：确定对不满足目标要求的软件采取什么措施（例如，加强审查，加强测试，重新设计软件，改进过程）。明确表示测试策略也是评估的一部分：它涉及确定、优先级、测试延续时间和结束日期，测试的人员和计算机资源的分配。

B.3.7 可靠性预计

穆沙、爱安尼诺(Iannino)和奥库姆脱在他们的书中表明，根据模型假设，并且失效率函数是均值函数的导数，有：

$$\hat{\lambda}(\tau)=\frac{\hat{\lambda}_0}{\hat{\lambda}_0\theta\tau+1} \tag{B.35}$$

到时刻τ 所经历的失效平均数$\hat{\mu}(\tau)$：

$$\hat{\mu}(\tau)=\frac{1}{\hat{\theta}}\ln(\hat{\lambda}_0\hat{\theta}\tau+1) \tag{B.36}$$

B.4 列透务德/弗尔洛模型

B.4.1 列透务德/弗尔洛模型的目标

列透务德/弗尔洛的意图是模拟软件失效过程的双随机性质。当软件失效并试图修复时需考虑不可靠性的两个基本来源。

首先，关于运行环境的特征有不确定性：我们不知道何时某个输入将显现，特别是我们不知道选定什么作为下一个输入。这样，即使我们完全知道哪些输入容易引起错误，我们仍然不能肯定何时会接收到下一个引起失效的输入。所有的软件可靠性模型都承认这个不确定性的来源，并常用一个简单的泊松过程来进行数学表示：即假设失效纯粹是随机地发生。这就意味着，例如，

到下一次失效的平均时间将具有指数分布。

不确定性的第二个来源与在试图排除引起失效的故障时偶然发生的事有关。前面提到的模型都假设失效过程是局部随机的，正是这种不确定性支配排错过程中失败率的变化：即它决定可靠性增长的特征。这里存在不确定性有两个主要原因：第一，显然不是所有的故障都对程序造成等量的不可靠性，而是有大有小。如果软件因一个已发现的故障而失效，该故障对总的可靠性有大的影响，那么排除该故障后可靠性将相应地有大的增长。第二，我们从来不能保证实际上已成功地排除了一个故障；的确可能引入某个新故障，而使程序的可靠性更差。这两种影响的结果是，随着排错继续，程序的失效率以随机方式变化：每次修复后失效率可能会向下跳变，但不一定，而且跳变的大小不能预计。

列透务德/弗尔洛模型与讨论过的其它模型不同，它考虑了失效过程中所有这两个不确定性的来源：为执行提供输入的环境所具有的基本不可预测性和在排错过程中人员活动影响的固定不确定性。

B.4.2 列透务德/弗尔洛模型的假设

下列假设适用于列透务德/弗尔洛模型：

a）软件在失效数据采集过程中的运行方式与要进行预计的那种方式相似；测试环境是运行环境的准确代表。

b）顺序的失效间隔的时间都是条件独立的指数随机变量，即失效过程是局部的（失效之间）纯随机的。

c）修复过程包括不确定性，使相继修复之后发生的相继失效能看成一个独立随机变量序列。

B.4.3 列透务德/弗尔洛模型的构造

这个模型把随着各次修理而出现的相继的失效率看成随机变量，并假设：

$$P(\lambda_i = \lambda) = \phi(i)e^{-\phi(i)\lambda} \quad (B.37)$$

把失效率 λ_i 序列看成随机地下降的独立随机变量序列。这就反映了修复会是有效的，但不一定。

假设：

$$g(\lambda_i) = \frac{\phi(i)^{\alpha}\,\lambda_i^{\alpha-1}e^{-\phi(i)\lambda_i}}{\Gamma(\alpha)}, \text{对于} \lambda_i \rangle 0 \quad (B.38)$$

这是一种具有参数是 α，$\phi(i)$ 的 Γ 分布。

函数 $\phi(i)$ 决定可靠性增长。通常如果 $\phi(i)$ 是 i 的增函数，就不难表明 Λ_i 形成一个随机下降的序列。对于这种模型，一次修复也许使程序更不可靠；即使发生了改进，改进的量也是不确定的。

列透务德/弗尔洛以最大似然法为基础，令 $\phi(i)$ 为 $\beta_0 + \beta_1 i$ 或 $\beta_0 + \beta_1 i^2$，并消去 α，从而给出了 β_0 和 β_1 的估计方法。从似然方程式消去 α，即该估计值可表示为其它两个参数估计值的函数。

参数（α，β_0，β_1）的最小二乘估计值可通过对

$$S(\alpha,\ \beta_0,\ \beta_1)=\sum_{i=1}^{n}\left(X_i-\frac{\phi(i)}{\alpha-1}\right)^2 \qquad (B.39)$$

求极小值而获得。

B.4.4 列透务德/弗尔洛模型的缺点

和所有贝叶斯分析一样，基本缺点是定下先验密度函数 $g(\lambda_i)$。其次一个缺点是它不能估计残留在软件中的故障数（这个估计值可能是无穷大，取决于 $\phi(i)$ 函数）。

B.4.5 列透务德/弗尔洛模型的数据需求

要求的数据是下列两种之一：

a）失效间隔时间，即 Xi；

b）失效发生时间，即：

$$t_i=\sum_{j=1}^{i}X_j \qquad (B.40)$$

B.4.6 列透务德/弗尔洛模型的应用

列透务德/弗尔洛模型是一种保守的悲观的模型。当运行剖面不均匀甚至不规则，特别当软件排错过程不完善时，这个模型就适用。这个模型有调整参数以反映这种情况的能力。

下面为模型的主要应用。这些单独而又相关的运用，共同组成一个综合的可靠性大纲。

a）预测：预测未来的失效时间、故障纠正和 B.4.7 所描述的有关量。

b）控制：将预测结果与预定目标进行比较，并标明不满足目标要求的软件。

c）评估：确定对不满足目标要求的软件采取什么措施（如进行更详细的测试，重新设计软件，改进过程）。明确表示测试策略也是评估的一部分；它涉及确定：优先级、测试延续时间和结束日期，测试的人员和计算机资源的分配。

B.4.7 可靠性预计

通过将各参数估计值代入到合适的表达式中，从而估计可靠性和其它相关项。平均失效前时间（MTTF）的估计值是：

$$MTT\hat{F}=\hat{E}(X_i)=\frac{[\hat{\phi}(i)]}{\hat{\alpha}-1} \qquad (B.41)$$

失效率的表达式是：

$$\hat{\lambda}(t)=\frac{\hat{\alpha}}{(t+\hat{\phi}(i))} \qquad (B.42)$$

可靠性函数是：

$$R(t)=P(T_i\rangle t)=\hat{\phi}(i)^{\hat{\alpha}}\left[t-\hat{\phi}(i)\right]^{\hat{\alpha}} \qquad (B.43)$$

在所有上述表达式中 $\hat{\phi}(i)$ 和 $\hat{\alpha}$ 都是 B.4.3 中两个响应参数的估计值。

附 录 C
（资料性附录）
软件测试常用摸板

C.1 软件测试用例

软件测试用例的格式如下所示：

用例名称			用例标识	
测试追踪				
用例说明				
用例的初始化	硬件配置			
	软件配置			
	测试配置			
	参数设置			
操作过程				
序号	输入及操作说明	期望测试结果	评价标准	备注
前提和约束				
过程终止条件				
结果评估标准				
设计人员		设计日期		

C.2 软件测试记录

软件测试记录的格式如下所示：

<table>
<tr><td colspan="2">用例名称</td><td colspan="2"></td><td>用例标识</td><td></td></tr>
<tr><td>用例说明</td><td colspan="5"></td></tr>
<tr><td rowspan="4">用例的初始化</td><td>硬件配置</td><td colspan="4"></td></tr>
<tr><td>软件配置</td><td colspan="4"></td></tr>
<tr><td>测试配置</td><td colspan="4"></td></tr>
<tr><td>参数设置</td><td colspan="4"></td></tr>
<tr><td colspan="6">操作过程</td></tr>
<tr><td>序号</td><td>输入及操作说明</td><td>期望测试结果</td><td>评估标准</td><td colspan="2">实测结果</td></tr>
<tr><td></td><td></td><td></td><td></td><td colspan="2"></td></tr>
<tr><td></td><td></td><td></td><td></td><td colspan="2"></td></tr>
<tr><td></td><td></td><td></td><td></td><td colspan="2"></td></tr>
<tr><td></td><td></td><td></td><td></td><td colspan="2"></td></tr>
<tr><td></td><td></td><td></td><td></td><td colspan="2"></td></tr>
<tr><td colspan="2">是否发生重启动 □</td><td>重启动是否成功 □</td><td>是否发生失效 □</td><td colspan="2">是否发生故障 □</td></tr>
<tr><td>测试结论</td><td colspan="5"></td></tr>
<tr><td>测试人员</td><td colspan="2"></td><td>测试日期</td><td colspan="2"></td></tr>
</table>

C.3 软件问题报告单

软件问题报告单的格式如下所示：

<table>
<tr><td colspan="2">问 题 标 识</td><td></td><td>项 目 名 称</td><td></td><td>程序/文档名</td><td></td></tr>
<tr><td colspan="2">发 现 日 期</td><td></td><td>报 告 日 期</td><td></td><td>报 告 人</td><td></td></tr>
<tr><td rowspan="2">问 题
性 质</td><td>类别</td><td>程序问题 ☐</td><td>文档问题 ☐</td><td>设计问题 ☐</td><td>其它问题 ☐</td><td></td></tr>
<tr><td>级别</td><td>1 级 ☐</td><td>2 级 ☐</td><td>3 级 ☐</td><td>4 级 ☐</td><td>5 级 ☐</td></tr>
<tr><td colspan="2">问 题 追 踪</td><td colspan="5"></td></tr>
<tr><td colspan="7">问题描述/影响分析：（可另加附页）
 

齐鲁软件测试
www.qltesting.cn</td></tr>
<tr><td colspan="7">附注及修改建议：（可另加附页）</td></tr>
</table>

附 录 D
（资料性附录）
软件测试内容的对应关系

依据GB/T16260.1对软件质量子特性的定义和传统测试内容分类的描述，本标准规定的测试内容与传统测试内容的对应关系见图D.1。

图 D.1 测试内容不同分类的对应关系