ICS 35.080
L 77

中华人民共和国国家标准

GB/T 38634.1—2020

系统与软件工程　软件测试
第1部分:概念和定义

Systems and software engineering—Software testing—Part 1:Concepts and definitions

(ISO/IEC/IEEE 29119-1:2013, Software and systems engineering—Software testing—Part 1: Concepts and definitions, MOD)

2020-04-28 发布　　2020-11-01 实施

国家市场监督管理总局
国家标准化管理委员会　发布

目　　次

前言 …… Ⅲ
引言 …… Ⅳ
1　范围 …… 1
2　符合性 …… 1
3　术语和定义 …… 1
4　软件测试概念 …… 10
附录A（资料性附录）　测试在验证和确认中的作用 …… 29
附录B（资料性附录）　度量和测度 …… 30
附录C（资料性附录）　不同生存周期模型中的测试 …… 31
附录D（资料性附录）　详细的测试子过程示例 …… 38
附录E（资料性附录）　测试角色和职责 …… 45
附录F（资料性附录）　本部分与ISO/IEC/IEEE 29119-1:2013相比的结构变化情况 …… 47
参考文献 …… 49

前　言

GB/T 38634《系统与软件工程　软件测试》分为以下4个部分：

——第1部分：概念和定义；

——第2部分：测试过程；

——第3部分：测试文档；

——第4部分：测试技术。

本部分为GB/T 38634的第1部分。

本部分按照GB/T 1.1—2009给出的规则起草。

本部分使用重新起草法修改采用ISO/IEC/IEEE 29119-1:2013《软件与系统工程　软件测试　第1部分：概念和定义》。

本部分与ISO/IEC/IEEE 29119-1:2013相比在结构上有较多调整，附录F列出了本部分与ISO/IEC/IEEE 29119-1:2013的章条编号对照一览表。

本部分还做了下列编辑性修改：

——将标准名称改为《系统与软件工程　软件测试　第1部分：概念和定义》；

——调整了参考文献，用我国标准代替了国际标准；

——在3.18术语和定义中增加了注释；

——删除了第3章和3.34中的注释。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别这些专利的责任。

本部分由全国信息技术标准化技术委员会(SAC/TC 28)提出并归口。

本部分起草单位：上海计算机软件技术开发中心、中国电子技术标准化研究院、中国航发控制系统研究所、中国航天系统科学与工程研究院、北京跟踪与通信技术研究所、国家应用软件产品质量监督检验中心、深圳赛西信息技术有限公司、西宁市大数据服务管理局、中国电子科技集团公司第五十四研究所、内蒙古安盾信息安全评测有限公司、中电莱斯信息系统有限公司、中国航天科工集团第三研究院第三〇四所、南京大学、厦门理工学院、上海同思廷软件技术有限公司、广东省科技基础条件平台中心、重庆市软件评测中心有限公司、中国司法大数据研究院有限公司、北京轩宇信息技术有限公司、北方民族大学、北京航空航天大学、浙江省电子信息产品检验所、上海浦东软件平台有限公司、中国电子科技集团公司第十研究所、上海第二工业大学。

本部分主要起草人：蔡立志、张旸旸、龚家瑜、左振雷、李文鹏、康京山、杨桂枝、王威、尹平、胡芸、吕雪、巩韶飞、吴克寿、赵昌平、阳长永、周震漪、白万芳、赵毅、李晓伟、徐宝文、丁晓明、路云峰、周晓明、江云松、孙凤丽、宋巍、王凤玲、韩强、郭新伟、陆澄澔、侯觅、孙海旺、李丽萍。

引　言

GB/T 38634 通过建立软件测试系列标准，可以使任何组织以任何形式执行规范化的软件测试。

软件、软件组织和方法多种多样。软件领域包括信息技术(IT)、个人电脑(PC)、嵌入式、移动型、科学型等诸多分类。软件组织的规模大小不同，有同地协作，也有全球范围合作；有商业化的，也有以公共服务为导向的。软件的方法包括面向对象、传统的、数据驱动的和敏捷方法。这些及其他因素都会影响软件测试。本标准可以在许多不同的环境中支持测试。

本部分通过引入标准术语，方便 GB/T 38634 其他部分使用，并提供其在实践中应用的例子。本部分信息提供定义的描述，即应用在本标准中的软件测试概念和软件测试过程方法的定义，可用于指导本标准的其他部分。

本部分讨论了软件测试的通用概念，描述了软件测试在组织和项目背景中的角色。解释了通用软件生存周期中的软件测试，介绍了如何为特定的测试项目或特定的测试目标建立软件测试过程和子过程。描述了软件测试如何适应不同的生存周期模型，阐明了测试计划中使用不同的实践以及如何使用自动化来支持测试。本部分也讨论了测试在缺陷管理中的应用。附录 A 描述了测试在验证和确认范围内的作用。附录 B 提供了度量和测量的简要介绍。附录 C 包含一组示例展示如何在不同的生存周期模型中应用本标准。附录 D 提供了详细的测试子过程的示例。附录 E 提供了关于测试组合测试人员独立性中常见的测试人员角色和职责的说明。

GB/T 38634 的测试过程模型在 GB/T 38634.2 进行详细定义。GB/T 38634.2 在组织级测试级别、测试管理级别和动态测试级别覆盖软件测试过程。测试是软件开发中风险处理的主要途径；本部分定义了基于风险的方法来测试。基于风险的测试是一种推荐的方法，用以管理测试，对测试进行优先级排序和聚焦。

GB/T 38634.3 定义了在测试过程中产生的测试文档的模板和实例。GB/T 38634.4 定义了在测试过程中使用的软件测试技术。

本标准旨在为任何组织中的利益相关方提供管理和执行软件测试的能力。

系统与软件工程　软件测试
第1部分:概念和定义

1　范围

GB/T 38634的本部分规定了软件测试的概念和定义。

本部分适用于GB/T 38634的软件测试过程、测试文档和测试技术。

2　符合性

GB/T 38634.1是资料性内容,不包含符合性要求。

GB/T 38634软件测试标准包括可声明符合性的三个部分:

——测试过程;

——测试文档;

——测试技术。

符合性由GB/T 38634.2、GB/T 38634.3和GB/T 38634.4具体提出。

3　术语和定义

下列术语和定义适用于本文件。

3.1

易访问性测试　accessibility testing

易用性测试的一种类型。用于测量某一测试项可以被用户所使用的程度,其中用户应覆盖最大范围的各种特征和不同能力的个体。

3.2

实测结果　actual result

作为测试执行结果可获取的测试项的行为、状态集,或相关数据、测试环境的状态集。

示例:向硬件的输出,对数据的更改,报表的生成和展现,通信消息的发送。

3.3

备份及恢复测试　backup and recovery testing

可靠性测试的一种类型。在失效事件中,在规定的时间、成本、完整性和准确性参数范围内,测量系统状态可以从备份中恢复的程度。

3.4

黑盒测试　black-box testing

见**基于规格说明的测试**(3.39)。

3.5

容量测试　capacity testing

性能效率测试的一种类型。用于评价增加负载(用户、交易、数据存储等)时,测试项能力水平和所需维持性能的折中点。

3.6

兼容性测试　compatibility testing

在与其他独立产品共享环境(共存)时,以及在需要与其他系统或组件进行信息交流(互操作)时,测量测试项满足要求的程度。

3.7

覆盖项　coverage item

见测试覆盖项(3.54)。

3.8

判定　decision

在两种或者多种可能的结果中作出选择,并决定动作集合的语句类型。

注:典型的判定有简单选择(例如,if-then-else),决定什么时候退出循环(例如,while-loop),以及多路开关语句(例如,case-1-2-3-...-*n*)。

3.9

动态测试　dynamic testing

需要运行测试项的测试。

3.10

持久测试　endurance testing

性能效率测试的一种类型。用于评价测试项在指定的时间段,是否能够持续承受所需的负载。

3.11

等价类　equivalence partition

变量或变量集的值域的子集。在测试项或者其接口中,预期测试项以同样的方式处理该子集中所有的值(即被认为是"等价")。

3.12

等价类覆盖率　equivalence partition coverage

测试集覆盖测试项中已识别等价类的比例。

注:在许多情况下,等价类识别是主观的(尤其是在"无效"等价类),所以对测试项中等价类进行明确计数是不可能的。

3.13

等价类划分　equivalence partitioning

测试设计技术的一种。使用每一个等价类中的一个或多个有代表性的成员来设计测试用例。

3.14

错误猜测　error guessing

测试设计技术的一种。基于以往失效经验或者失效模式的常识推导出测试用例。

注:相关知识能通过个人经验获得,或以缺陷数据库或"缺陷分类"的形式进行封装。

3.15

预期结果　expected result

根据规格说明或其他来源,在特定的条件下可获取的测试项的预期行为。

3.16

探索性测试　exploratory testing

一种基于经验的测试。测试者基于其现有的相关知识、测试项的前期探索(包括以前的测试结果)以及关于通常软件行为和故障类型的启发式"经验法则",自发地设计和执行测试。

注:探索性测试寻找隐含属性(包括隐含的行为),虽然其自身存在危害的可能性很小,但其可能干扰待测软件的其他属性,并因此产生软件失效的风险。

3.17

特征集　feature set

包含测试项的测试条件的集合，可以从风险、需求、功能、模型等方面收集得到。

注：可能是该项的所有特征（其全部特征集），或为特定的目的而标识的子集（功能特征集等）。

3.18

事件报告　incident report

事件发生、性质和状态的文档。

注：事件报告也称为异常报告、错误报告、缺陷报告、差错报告、问题报告等。

3.19

易安装性测试　installability testing

可移植性测试的一种类型。用于评价一个或一组测试项是否可以安装在所有指定的环境下。

3.20

负载测试　load testing

性能效率测试的一种类型。用于评价测试项在预期变化负载下的行为，负载通常位于低谷、典型和高峰使用的预期条件之间。

3.21

维护性测试　maintainability testing

评价修改测试项的有效性和效率。

3.22

组织级测试方针　organizational test policy

组织进行测试的目的、目标和整体范围。

注：组织级测试方针尽可能简短。

3.23

组织级测试过程　organizational test process

开发和管理组织级测试规格说明的测试过程。

3.24

组织级测试规格说明　organizational test specification

为一个组织的测试提供信息的文档，该信息并不针对具体项目。

示例：组织级测试规格说明最常见的例子是组织级测试方针和组织级测试策略。

3.25

组织级测试策略　organizational test strategy

为组织内所有项目执行测试提供一般要求的文档；提供有关如何执行测试的细节。

注1：与组织级测试方针保持一致。

注2：一个组织可以有多个组织级测试策略以应对不同背景的项目。

3.26

通过/不通过准则　pass/fail criteria

用于确定测试项或测试项的特征是否通过的判定规则。

3.27

性能测试　performance testing

用于评价测试项在给定时间或其他资源约束下，完成其指定功能程度的一种测试。

3.28

可移植性测试　portability testing

用于评价测试项从一个硬件或软件环境迁移到另一个硬件或软件环境的容易程度的一种测试，包

括在多种环境下运行测试项所需的修改程度。

3.29

规程测试 procedure testing

功能性测试的一种类型。用于评价与测试项交互或使用其输出的规程指令是否满足用户需求及其使用目的。

3.30

产品风险 product risk

产品在其功能、质量或结构的某些特定方面可能存在缺陷的风险。

3.31

项目风险 project risk

与项目管理有关的风险。

示例：人员短缺、严格的期限、需求变更。

3.32

回归测试 regression testing

测试项或其运行环境修改后执行的测试。

注：回归测试用例集的充分性取决于测试项本身及测试项和运行环境的修改。

3.33

可靠性测试 reliability testing

为评价测试项执行其要求功能的能力的一种测试，包括评价在指定条件下运行一段时间中故障发生的频率。

3.34

复测 retesting

重新执行测试结果为“不通过”的测试用例，以评价纠正措施的有效性。

3.35

基于风险的测试 risk-based testing

基于风险分析确定的风险类型和级别，有意识地管理、选择、排序和利用测试活动及资源的测试。

3.36

场景测试 scenario testing

一种测试设计技术，设计测试执行的各个场景。

注：场景可以是用户故事、用例、操作概念或软件可能遇到的事件序列等。

3.37

脚本测试 scripted testing

测试者的动作由测试用例中的书面指令来规定的动态测试。

注：脚本测试通常适用于人工执行的测试，而不是自动化脚本的执行。

3.38

信息安全性测试 security testing

为评价测试项及相关数据和信息受到保护程度的一种测试，以确保未经授权的人员或系统不能使用、读取或修改它们，且不拒绝授权人员或系统的访问。

3.39

基于规格说明的测试 specification-based testing

主要基于测试项外部输入和输出的一种测试，通常是依据规格说明而不是源码实现或可执行软件。

注：基于规格说明的测试的同义词为黑盒测试。

3.40

语句覆盖率　statement coverage

测试项中所有可执行语句被测试集覆盖的百分比。

3.41

语句测试　statement testing

设计测试用例用于执行测试项中某条语句的测试设计技术。

3.42

静态测试　static testing

在不运行代码的情况下，通过一组质量准则或其他准则对测试项进行检查的测试。

示例：评审、静态分析。

3.43

压力测试　stress testing

性能效率测试的一种类型。用于评价测试项在高于预期或指定容量负载需求，或低于最少需求资源的条件下的行为。

3.44

结构测试　structural testing

见**基于结构的测试**(3.45)。

3.45

基于结构的测试　structure-based testing

由测试项结构导出测试用例的动态测试。

注 1：基于结构的测试并不局限于在组件级别的使用，其可以被用于所有级别，如系统测试中菜单项覆盖。

注 2：技术包括分支测试、判定测试和语句测试。

注 3：基于结构的测试的同义词包括结构测试、白盒测试。

3.46

挂起准则　suspension criteria

用于(暂时)停止全部或部分测试活动的准则。

3.47

测试依据　test basis

作为测试分析和测试用例设计基础的知识体系。

注：测试依据可以采用文件的形式，例如需求规格说明、设计规格说明或模块规格说明，但也可以是非书面形式对需求行为的理解。

3.48

测试用例　test case

前置条件、输入(包括操作，如果适用)和预期结果的集合，用于驱动测试项的执行以满足测试目标，测试目标包括正确实现、错误识别、检查质量和其他有价值的信息。

注 1：测试用例是测试子过程的最低测试输入级别。(即，测试用例无法再划分为更细的测试用例)

注 2：测试用例的前置条件包括测试环境、已有数据(如数据库)、被测软件、硬件等。

注 3：输入是用于驱动测试执行的数据信息。

注 4：预期结果包含通过的准则、失效的校核。

3.49

测试用例规格说明　test case specification

一个或多个测试用例组成的文档集。

3.50

测试完成过程　test completion process

测试管理过程的子过程。用于确保有用的测试资产可供以后使用、测试环境保持在令人满意的状态、测试结果被记录并传达给利益相关方。

3.51

测试完成报告　test completion report

描述已完成测试的总结报告。

注：测试完成报告也被称为测试总结报告。

3.52

测试条件　test condition

组件或系统可测的方面，如作为测试依据的功能、事务、特征、质量属性或者结构元素。

注：测试条件常用来导出覆盖项，或由其本身构成覆盖项。

3.53

测试覆盖率　test coverage

以百分比表示的、用以表示一个或多个测试用例实现指定测试覆盖项的程度。

3.54

测试覆盖项　test coverage item

使用测试设计技术从一个或多个测试条件导出的属性或属性组合，可以用于测量测试执行的充分性。

3.55

测试数据　test data

为满足执行一个或多个测试用例的输入需求而创建或选择的数据，该数据可在测试计划、测试用例和测试规程中定义。

注：测试数据可以存储在被测的产品中(例如阵列、平面文件或数据库)，也可以从外部源获得或由外部源提供，如其他系统、其他系统组件、硬件设备或人员提供。

3.56

测试数据准备报告　test data readiness report

描述每个测试数据需求准备状态的文档。

3.57

测试设计和实现过程　test design and implementation process

生成和确定测试用例和测试规程的测试过程。

3.58

测试设计规格说明　test design specification

规定测试项的被测特征及其相应测试条件的文档。

3.59

测试设计技术　test design technique

用于构建测试模型的活动、概念、过程和模式，该模型用于识别测试项的测试条件，导出相应的测试覆盖项，并导出或选择测试用例。

3.60

测试环境　test environment

用于执行软件测试的设施、硬件、软件、固件、规程和文档集。

注：测试环境可包括多种环境以适应指定的测试子过程(例如单元测试环境、性能测试环境等)。

3.61

测试环境准备报告　test environment readiness report

描述每个测试环境需求实现程度的文档。

3.62

测试环境需求　test environment requirements

测试环境必要性质的描述。

注：所有或部分的测试环境需求可以参考可获取的信息，例如合适的组织级测试策略、测试计划和/或测试规格说明。

3.63

测试环境构建过程　test environment set-up process

建立和维护所需的测试环境的过程。

3.64

测试执行　test execution

在测试项上执行测试并产生实测结果的过程。

3.65

测试执行日志　test execution log

记录一个或多个测试规程执行细节的文档。

注：测试执行日志也称为测试记录。

3.66

测试执行过程　test execution process

动态测试过程的子过程。用于在准备好的测试环境中执行测试设计和实现过程中创建的测试规程，并记录其结果。

3.67

测试事件报告过程　test incident reporting process

动态测试过程的子过程。用于向利益相关方报告在测试执行过程中确定的、需要进一步处理的问题。

3.68

测试项　test item

作为测试对象的工作产品。

示例：系统、软件项、需求文档、设计规格说明、用户指南。

3.69

测试级别　test level

测试子过程的具体实例化。

示例：以下是常用的测试级别，可以实例化为测试子过程：组件测试级别/子过程、集成测试级别/子过程、系统测试级别/子过程、验收测试级别/子过程。

注：测试级别与测试阶段同义。

3.70

测试管理　test management

测试活动的策划、安排、预估、监测、报告、控制和完成。

3.71

测试管理过程　test management process

包含测试项目管理所需子过程的测试过程。

注：见测试策划过程、测试监测和控制过程、测试完成过程。

3.72

测试监测和控制过程　test monitoring and control process

测试管理过程的子过程。用以确保测试按照测试计划和组织级测试规格说明执行。

3.73

测试对象　test object

见**测试项**(3.68)。

3.74

测试阶段　test phase

测试子过程的具体实例化。

3.75

测试计划　test plan

描述需要达到的测试目标以及实现该测试目标的方法和安排的文档，用于协调测试项的测试活动。

注1：一个项目可以有多个测试计划，例如可以有一个项目测试计划(也称为主测试计划)，其包含了该项目所有的测试活动；更多测试活动的细节可在一个或多个测试子过程计划(即，系统测试计划或性能测试计划)中定义。

注2：通常测试计划是书面记录的，尽管其他的计划形式也可在组织或项目中局部定义。

注3：也可以为非项目活动编写测试计划，例如维护测试计划。

3.76

测试策划过程　test planning process

测试管理过程的子过程。用于完成测试策划和开发测试计划。

3.77

测试实践　test practice

便于实施测试的概念框架，该框架可应用于组织级测试过程、测试管理过程和/或动态测试过程。

3.78

测试规程　test procedure

测试用例的执行序列，以及任何与构建初始前置条件所需的相关动作和执行后的收尾活动。

注：测试规程包括如何连续运行一个或多个测试用例的详细说明，包括设置通用的前提条件，为每个测试用例提供输入并评价实际结果。

3.79

测试规程规格说明　test procedure specification

说明一个或多个测试规程的文档。这些测试规程是具有特定目标的测试用例的集合。

注1：测试集内的测试用例按测试规程的需求顺序列出。

注2：测试规程规格说明也称为人工测试脚本。自动化测试运行的测试规程规格说明通常被称为测试脚本。

3.80

测试过程　test process

为一个软件产品提供质量信息的过程，通常由多个活动组成，分为一个或多个测试子过程。

注：特定项目的测试过程可能包含多个子过程，如系统测试子过程、测试计划子过程(较大测试管理过程的一部分)或静态测试子过程。

3.81

测试需求　test requirement

见**测试条件**(3.52)。

3.82

测试结果　test result

指定的测试用例是否通过的标示，即观察到测试项输出的实测结果是否与预期结果一致或有偏差。

3.83

测试脚本　test script

人工测试或自动化测试的测试规程规格说明。

3.84

测试集　test set

一个或多个测试用例的集合，其执行时具有共同的约束。

示例：特定的测试环境、专业的领域知识或特定的目的。

3.85

测试规格说明　test specification

包含针对特定测试项的测试设计、测试用例和测试规程的全部文档集。

注：测试规格说明具体可以是一个文档、文档集合或以其他方式，例如文档和数据库条目的混合。

3.86

测试状态报告　test status report

提供在指定报告期间所执行测试的状态信息的报告。

3.87

测试策略　test strategy

测试计划的一部分。描述对特定测试项目或测试子过程进行测试的方法。

注1：测试策略和组织级测试策略是不同的。

注2：测试策略通常描述以下部分或全部：使用的测试实践、实现的测试子过程、采用的复测和回归测试、使用的测试设计技术和相应的测试完成准则、测试数据、测试环境和测试工具需求、测试交付物的预期。

3.88

测试子过程　test sub-process

通常在测试项目的整体测试过程的周境中，用于执行特定的测试级别（例如系统测试、验收测试）或测试类型（例如易用性测试、性能测试）的测试管理和动态（和静态）测试过程。

注：测试子过程可以包含一个或多个测试类型。根据使用的生存周期模型，测试子过程也通常被称为测试阶段或测试任务。

3.89

测试技术　test technique

见测试设计技术（3.59）。

3.90

测试追踪矩阵　test traceability matrix

用于识别文档集和软件中相关项（如测试相关的需求）的文档、电子表格或其他自动化工具。

注1：测试追踪矩阵也称为验证交叉引用矩阵、需求测试矩阵、需求验证表等。

注2：不同的测试追踪矩阵可能有不同的信息、格式和细节程度。

3.91

测试类型　test type

一组专注于特定质量特性的测试活动。

注：一种测试类型可以在单个测试子过程中执行或跨多个测试子过程执行（性能测试在组件测试子过程中完成，也在系统测试子过程中完成）。

示例：信息安全性测试、功能性测试、易用性测试和性能测试。

3.92

测试　testing

为发现和/或评价一个或多个测试项的属性而进行的一系列活动。

注：测试活动可包括针对测试的计划、准备、执行、报告和管理活动，其均与测试直接相关。

3.93

测试件　testware

计划、设计和执行测试所需的、在测试过程中产生的制品。

注：测试件可以包括文档、脚本文件、输入、预期结果、文件、数据库、环境以及任何用于测试过程中附加的软件或设备。

3.94

无脚本测试　unscripted testing

测试者的动作不是由测试用例中的书面指令所规定的动态测试。

3.95

容积测试　volume testing

性能效率测试的一种类型。用于评价测试项在吞吐量、存储容量或两者兼考虑的情况下处理指定数据量(通常达到最大指定容量或接近最大值)的能力。

3.96

白盒测试　white box testing

见**基于结构的测试**(3.45)。

4　软件测试概念

4.1　软件测试概述

4.1.1　概述

软件测试的必要性如下：

——决策者需要有关测试项质量特性的信息；

——被测项并不总是达到预期效果；

——被测项需验证；

——被测项需确认；

——被测项的评价需贯穿整个软件与系统开发生存周期。

无法构建出完美的软件是已经被普遍认同的。因此，在软件发布向用户之前有必要进行测试，降低软件产品出错的风险，以避免在使用软件时产生负面影响。同样，有必要确保测试能被很好地执行。

错误或缺陷很大程度上是不可避免的。人为引起的错误或差错导致了软件工作产品(例如，一个需求规格说明或软件组件)缺陷的出现。如果在软件使用时未遇到缺陷，缺陷不会对软件的操作产生影响。但是，如果在一定的条件下使用产品时遇到缺陷，缺陷可能会导致软件产品无法符合用户的合理需求。由软件失效可能会带来用户体验的严重后果。例如，一个缺陷可能会损害商业信誉、公共安全、商业可行性、商业或用户的信息安全、环境等。

动态测试是必要的，但无法保证软件能按预期执行。额外的静态测试活动，例如同行评审和静态分析，应与有效的动态测试活动结合使用。

测试的主要目标是：提供有关测试项的质量信息以及与测试项测试数量相关的任何残余风险；在发布之前发现测试项的缺陷；并减轻由产品质量不足带给相关利益方的风险。

软件质量信息可用于多种目的，包括：

——通过消除缺陷来完善测试项；

——将提供的质量和风险信息作为决策的基础来改善管理决策；

——通过标识允许出现缺陷和/或可能发现但依然未发现缺陷的过程，来改进组织中的过程。

产品质量有多个方面，包括符合规格说明的要求、没有缺陷，以及满足产品用户需求的符合程度。可通过测试来测量和评价 GB/T 25000.10—2016 系统与软件质量模型定义的八个质量特性(见 4.5.4)。

在给定成本和进度的约束下，软件测试应侧重于提供有关软件产品的质量信息，并在开发过程中尽早且尽可能多地发现缺陷。

测试考虑的因素包括：

——测试是一个过程。过程是将输入转为输出的一系列相互关联或相互作用的活动。本标准的目

的是提出并描述通用的测试过程(详细内容参见 GB/T 38634.2 测试过程)。

——应用于跨组织的项目和功能的测试方针和测试策略,它由组织级测试过程设置和维护。

——应对测试进行计划、监测和控制。GB/T 38634.2 所包含的测试管理过程,可应用于所有开发生存周期中的测试和探索性测试的管理。

——测试过程和子过程可以应用于任何测试阶段或级别(如系统测试)或任何类型的测试(如性能测试)。

——测试需要检查测试项。

——无须在计算机上运行产品也可以进行测试。这种测试技术在本标准和许多行业领域中被称为静态测试,尽管在其他标准中(例如,GB/T 32421—2015)也许更具体地被称为审查、走查或检查。本标准承认并认可测试者在静态测试活动中的作用,即使这些活动可能在其他标准工作组或在其他非测试标准中定义。静态测试活动对于全生存周期的测试非常重要,并且测试介入对于早期缺陷检测,降低项目总成本和保证项目进度至关重要。

——静态测试包括使用静态分析工具在不运行代码的前提下发现代码和文档中的缺陷(例如编译器、圈复杂度分析器,或代码的安全分析器)。

——动态测试不仅仅是运行可执行的测试项,还包括了准备活动和后续活动。GB/T 38634.2 中描述的动态测试过程涵盖了动态测试所要执行的每一个活动。

——验证是通过提供客观证据来认定在给定的工作项中已经满足了指定的需求。

——确认表明用户可以使用该工作项来执行其特定任务。

——无论是静态测试还是动态测试,旨在提供验证和确认两种认定所需要的信息,尽管由于发现了缺陷而无法立即认定。

4.1.2 测试在验证和确认中的作用

软件测试是验证和确认的主要活动,本标准仅涉及验证和确认中的软件测试,不涉及验证和确认的其他活动(如 V&V 分析、形式化方法)。其他标准,如 ISO/IEC/IEEE 12207 和 GB/T 32423—2015,涉及其他的验证和确认活动。只有将本标准与其他标准结合使用,并作为整个工程项目的一部分,才能开展完整的产品验证和确认活动。有关验证和确认活动的层次结构参见附录 A。

4.1.3 穷尽测试

由于系统和软件的复杂性,测试不可能覆盖测试项的每个方面。测试人员应该明白穷尽测试是不可能的,测试活动应聚焦于最能满足测试项的测试目标。基于风险的测试是一种用风险来指导测试工作的方法。基于风险的测试见 4.4。

4.1.4 启发式测试

在工程(软件工程)中,启发式是一种基于经验(尝试错误)的方法,可用于帮助解决和设计问题。虽然启发式算法可以用来解决问题,但它并不可靠,无法解决所有问题,或只能解决部分问题。许多系统和软件的测试都是基于启发式。例如,通过对被测系统进行建模,启发式测试非常有用,但是可能无法对系统完全建模,因此即使测试似乎已经完成,也可能无法找到系统中的缺陷。如果认识到测试方式可能是错误的,可以采用多种测试策略来降低无效测试策略的风险。

4.2 组织和项目环境中的软件测试

4.2.1 概述

从事软件产品开发或采购的企业有兴趣开发和使用有效、高效和可重复的过程。为此,他们通常会

研制一套可靠的过程体系，应用于所执行的项目中。本标准既适用于整个组织，也适用于特定项目。组织在采用本标准时，可在必要时补充其他规程、实践、工具和方针。在组织内开发特定软件或系统时，通常需符合组织内定义的过程体系，而不用直接符合本标准。在某些情况下，若组织不具备适当的过程体系，其开展的项目可直接按本标准的规定执行。

在各种类型的软件生产组织中，无论是拥有数千名测试人员的跨国公司还是只有少量测试人员的小公司，软件测试都宜得到组织管理最高层的承诺，无论是CEO、开源指导委员会或部门经理。承诺作为组织内所有软件测试的基础，最好以组织级测试方针和一个或多个组织级测试策略表示。通常只有在成熟度较高的组织中，才具备测试方针和组织级测试策略。在成熟度较低的组织中，缺乏正式的测试方针和组织级测试策略也可以开展测试，但会造成测试执行在组织内缺乏连贯性，会降低测试执行的效率和有效性。

软件测试是周境管理的过程。这意味着宜对软件测试进行计划、监测和控制。周境可以是开发项目(覆盖从多人、多年的正式开发项目到几个人时的非正式开发项目)或正在进行维护的业务系统。理解测试周境的因素有：总体预算、进度需求、风险、组织文化、客户/用户期望、可用于测试的基础设施环境、项目范围、项目重要性等。行业经验显示任何单一的测试策略、计划、方法或过程都不可能适用于所有情况。组织和项目宜参考本标准来定制和改进测试的细节。

完整的项目计划应考虑将测试活动作为其中的一部分。项目测试计划宜同时反映组织级测试方针和组织级测试策略以及与组织级指导文件之间的偏差，还宜考虑项目计划中给出的限制因素。项目测试计划包括项目测试策略和导出此策略的项目特定决策(包括假设)。测试计划的一个重要内容是衡量各个测试需求和平衡各个测试间的资源，并将分析结果记录在测试计划中。本部分中，风险是决定测试需求的主要方法(见4.4)，4.6中描述的测试实践可归为策略。

一个项目的测试通常会包括多个测试子过程，每一个测试子过程可具有相应的测试计划(测试子过程计划，例如系统测试计划或性能测试计划)，其包括与项目测试策略一致的测试子过程策略，以及测试子过程的具体细节。

图1显示了测试的多层周境。适用时，测试是建立在组织的监管状态下。周境是由法律、法规和行业标准组成的。在这种监管状态下，组织为了项目能够成功，需制定必要的方针和规程，并执行组织级的测试策略。在组织内的每个项目都经过实例化，以满足组织已经确定的需求或机会。项目周境有助于确定选择生存周期模型。在此级别，测试策略的确定基于项目周境和生存周期模型。项目计划和测试策略构成了项目测试计划的基础。

项目测试计划描述了总体测试策略和测试过程，通过确定目标、实践、资源和进度安排来建立项目测试周境；同时也确定了适用的测试子过程(如系统测试、性能测试)。然后在子过程测试计划(例如，系统测试计划、性能测试计划)中描述所识别的子过程。测试计划还描述了所使用的测试设计技术(静态或动态)，包括特定子过程计划中所用的测试技术。有关测试设计技术的更多信息，见4.5.7和GB/T 38634.4。

每个子过程测试计划可用于多个测试级别(例如，信息安全性测试计划可能涉及多个测试级别)和多个测试类型(例如，系统测试计划用于处理系统测试级别的功能和性能测试)。子过程测试计划还包括测试执行的策略(例如，脚本、无脚本或两者的混合)。

测试计划包含测试策略。测试策略适用于特定的测试项目周境，用来解决图1中列出的特定项，这些项在本标准的其他部分中进行了定义。每个测试计划(和策略)都是独一无二的，包含不同的测试子过程、自动化等级、测试技术、测试完成准则以及日程安排和资源等。在项目早期就应做出计划和选择，并持续作为风险变化等因素贯穿于测试生存周期。尽管这些变更可能会受到项目、组织及监管约束的限制，但是测试计划和策略中的大部分内容还是会发生变更。

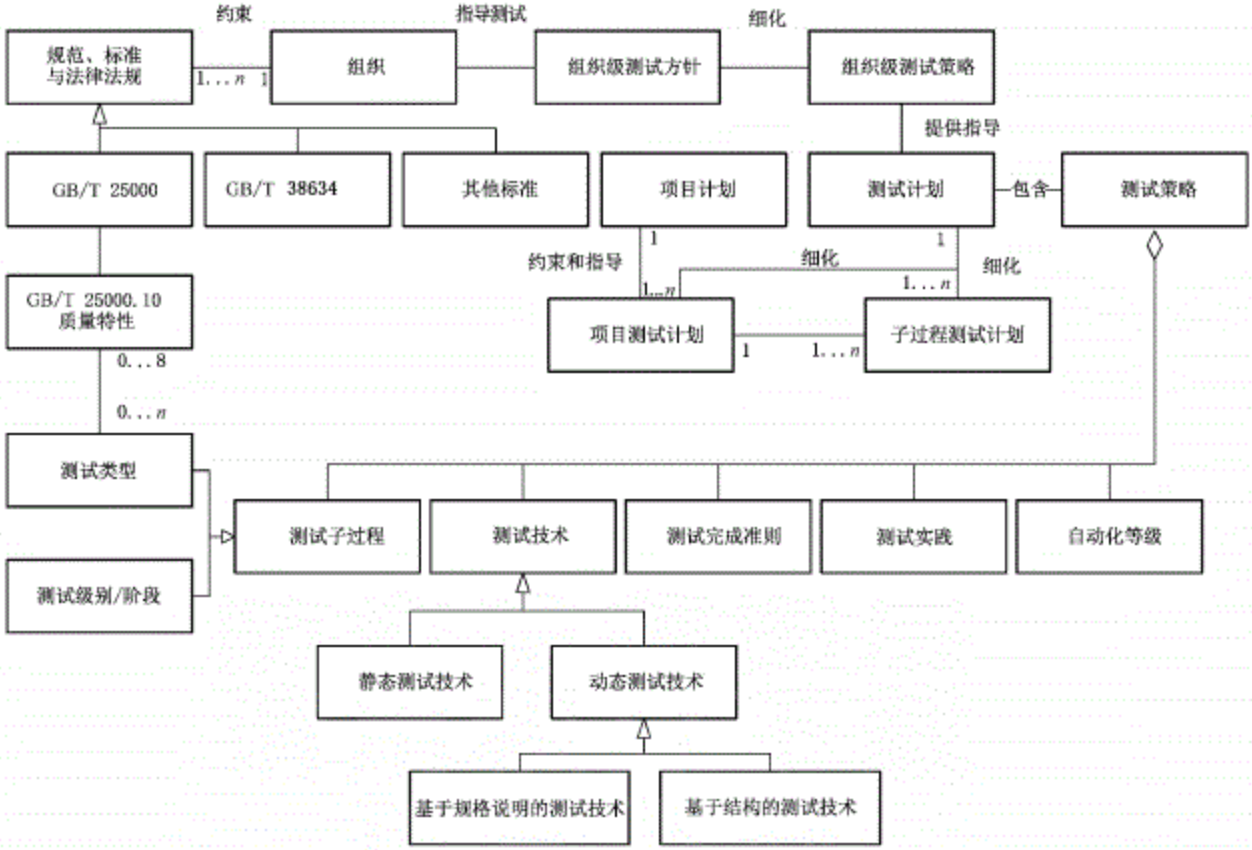

图 1　多层次测试周境图

图 2 详细描述了通用测试过程、通用测试子过程、测试级别/阶段和测试类型之间的关系，展示了通用测试过程作为特定的测试级别和测试类型的实现。通用测试子过程可以应用在以下几个方面：

——在测试级别或阶段方面，即每个测试级别是测试子过程的特定应用(例如，组件测试阶段，验收测试级别)；

——在测试类型方面，即每种测试类型是通用测试子过程的特定应用(例如，性能测试、易用性测试)；

——测试级别子过程可包含多个测试类型子过程(例如，功能和性能测试作为系统测试的一部分)；

——项目测试过程可包含一系列测试子过程(例如，组件测试、集成测试、系统测试和验收测试子过程)。

图 2 还明确了测试类型和质量特性之间的关系(如 GB/T 25000.10—2016 定义的系统和软件质量模型)。每种测试类型都针对一个特定的质量特性。测试类型和质量特性之间关系的更多信息见4.5.4。

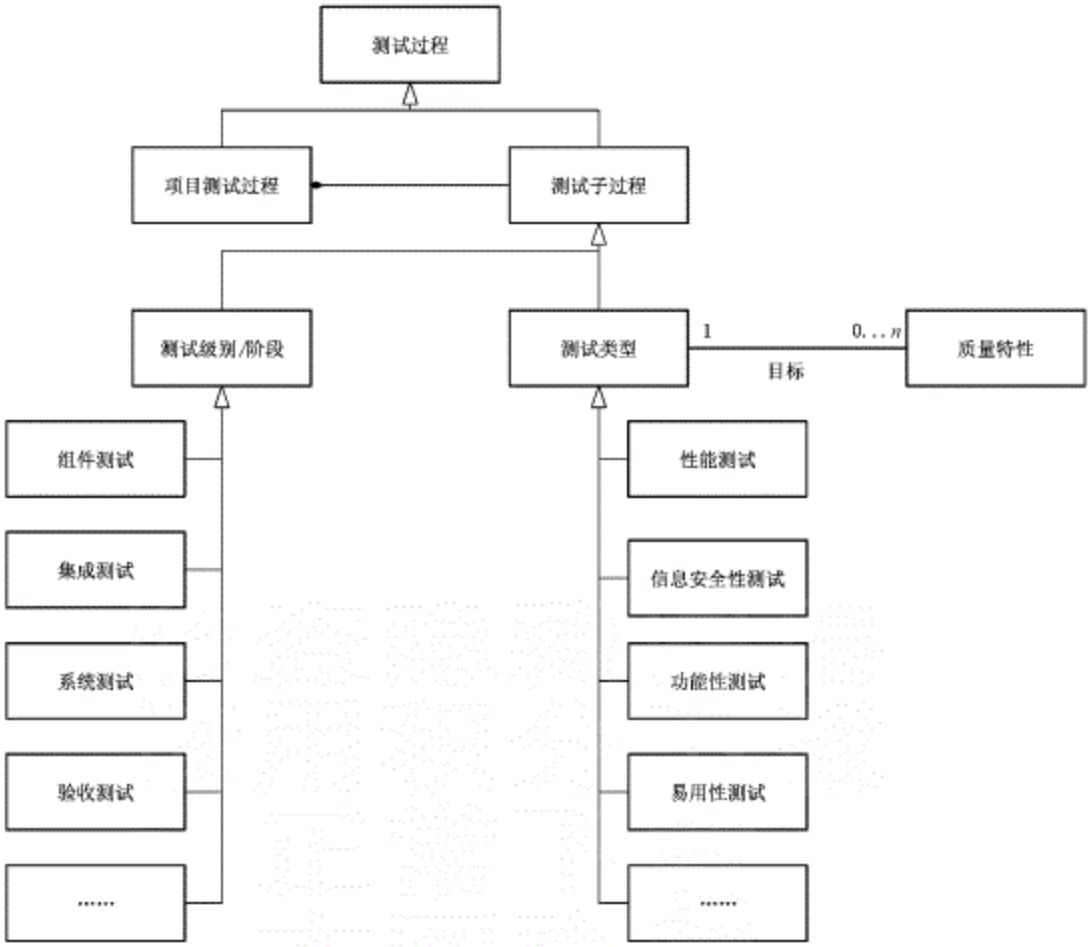

图 2　通用测试子过程、测试级别和测试类型之间的关系

4.2.2　测试过程

本标准采用了三层过程模型。图 3 给出了高层次的模型，其详细描述见 GB/T 38634.2。过程模型以管理高层(组织级)测试规格说明的组织级开始，如组织级测试方针和组织级测试策略。中间层是测试管理(项目测试管理、阶段测试管理、类型测试管理)，底层定义了用于动态测试的测试过程。

三层过程模型如图 3 所示。

图 3　测试过程之间的多层关系

测试方针是用业务术语表达了组织对软件测试的管理期望和方法，适用于所有参与测试的人员，但主要对象是高层管理者。测试方针还指导了有关组织级测试策略的制定和组织级测试过程的实施。组织级测试方针的创建、实现和维护由组织级测试过程定义。

组织级测试策略表达了对组织内运行的所有项目执行测试管理过程和动态测试过程的要求和约束。当组织内各项目间性质上存在较大差异时，可制定多个组织级测试策略。组织级测试策略与组织级测试方针保持一致，并描述如何执行测试。组织级测试策略的创建、实施和维护也由组织级测试过程定义。

测试管理过程中对执行测试的过程提出管理要求。对已识别项目风险和约束条件进行分析，并考虑组织级测试策略，提出项目级测试策略。在策略中定义和详细阐述要执行的静态和动态测试、总体人员配置、平衡给定约束(资源和时间)、要完成的测试工作范围和质量要求等方面，并记录在项目测试计划中。在测试期间，执行监测活动以确保测试按计划进行，并确保风险得到正确处理。同时，如果需要对测试活动进行任何变更，则向相关测试过程或子过程发出控制指令。在监测和控制期间，可定期生成测试状态报告，以告知利益相关方测试进度。项目测试的总体结果记录在项目测试完成报告中。

测试管理过程如图 4 所示。

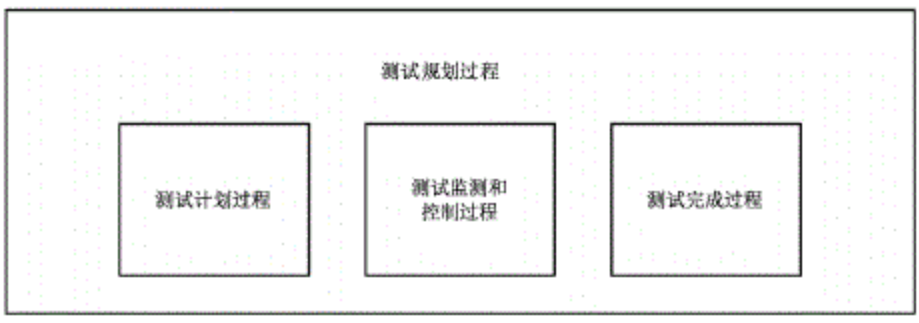

图 4　测试管理过程

项目的整体测试通常分解为较小的测试子过程(例如组件测试、系统测试、易用性测试、性能测试)，这些子过程也宜以类似于整体测试项目的方式进行管理、执行和报告。测试管理过程也可应用于测试子过程。例如，测试子过程计划可以是系统测试计划、验收测试计划或性能测试计划。

测试子过程可包含静态测试和动态测试。动态测试过程在图 5 中概述，并在 GB/T 38634.2 中详细说明。静态测试过程由其他已发布的标准(例如，GB/T 32421—2015)描述。

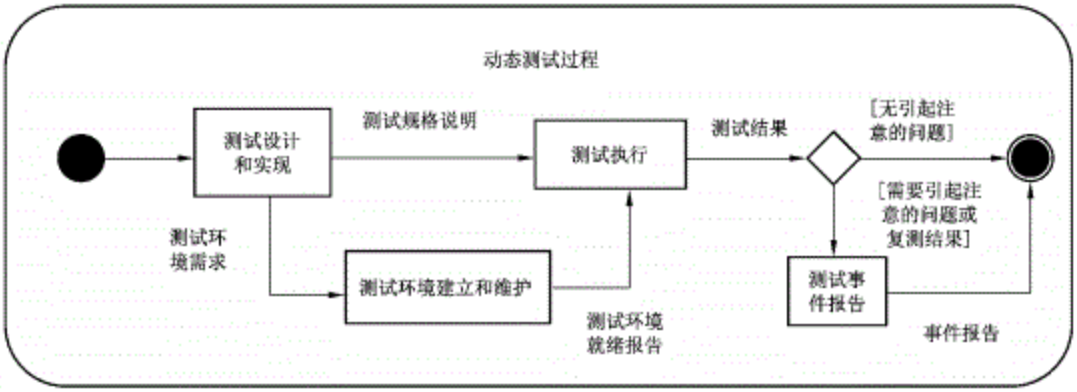

图 5　动态测试过程

有关测试过程的更多信息，包括组织级测试过程、测试管理过程和动态测试过程，参见GB/T 38634.2。

4.3　软件生存周期中的通用测试过程

4.3.1　概述

软件从最初构想到最终退役都有一个预期的生存周期。软件测试在软件开发和维护周境中进行。本条概述了软件开发生存周期以及其子过程和测试过程之间关系的示例。ISO/IEC/IEEE 12207:2017详细描述了软件生存周期。ISO/IEC/IEEE 15288详细描述系统生存周期过程。系统的生存周期样例如图6所示。

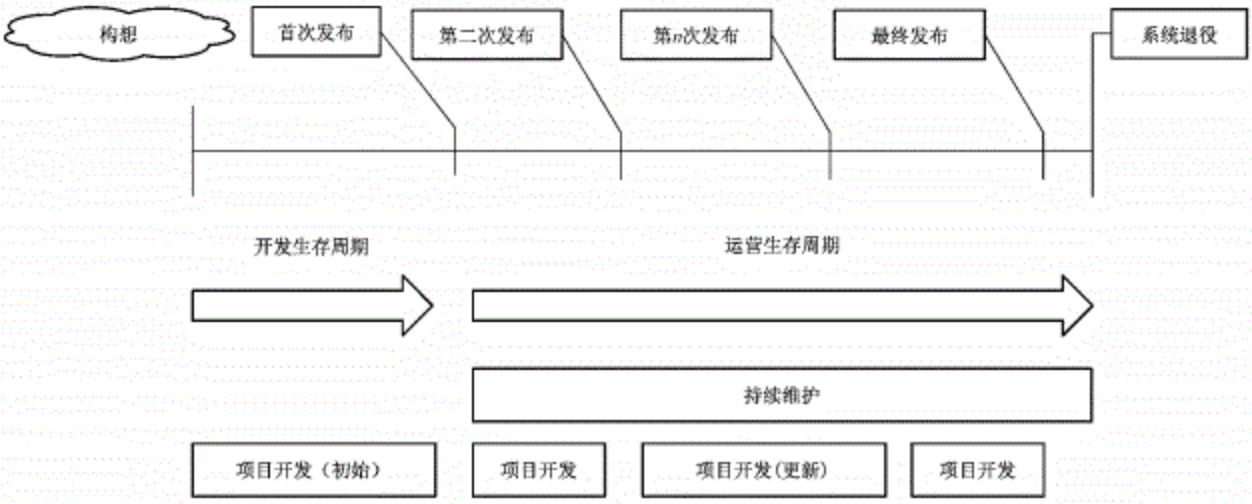

图 6　系统生存周期样例

软件生存周期通常包括多个子生存周期。图6表明了一个软件生存周期往往由一个或多个开发生存周期和一个或多个运营生存周期组成的。

从构想到最初发布被称为开发生存周期，是软件生存周期的子集。开发生存周期在开发项目中管理和控制。

系统从首次发布到退役之前一直处于运行状态；这可能是几个小时到几十年的时间跨度。运行期通常包括使用特定版本系统的时间点，同时新版本正在开发，以待发布。任何正在开发的新版本都应视为一个开发项目，其中包含相应的测试。通常还会进行持续维护以保持系统可用并可按预期运行。

在没有相应开发项目的情况下，可在运营系统上进行测试的情况，例如“试运行”灾难恢复测试。本标准提出的过程也可适用于这种情况。

测试的执行也可用于评价购买的软件是否满足业务需求。GB/T 25000.51—2016 提供了用于评价和测试就绪可用软件(RUSP)的框架。

4.3.2 开发项目子过程及其结果

软件和系统开发通常由几个常见的构建模块组成。在软件行业，这些构建模块通常被称为“阶段”“时期”“步骤”“级别”或更普遍地称为“开发子过程”。包括：

——需求工程；

——架构设计；

——详细设计；

——编码。

组织或项目采用特定的系统开发方法来决定如何策划开发子过程。在顺序开发项目中，需求分析本身可以是一个初始过程。在敏捷开发项目中，每隔几周就会为每次增量交付确定需求。

在每个开发子过程中都会产生一些成果；可以是一个高度结构化的详细文档，也可以是非正式记录或未记录的决策。

在任何给定的开发项目中，各个开发子过程可以执行一次，或者根据需求重复执行。通用开发子过程和相关工作产出物、子系统和完整软件系统如图 7 所示。

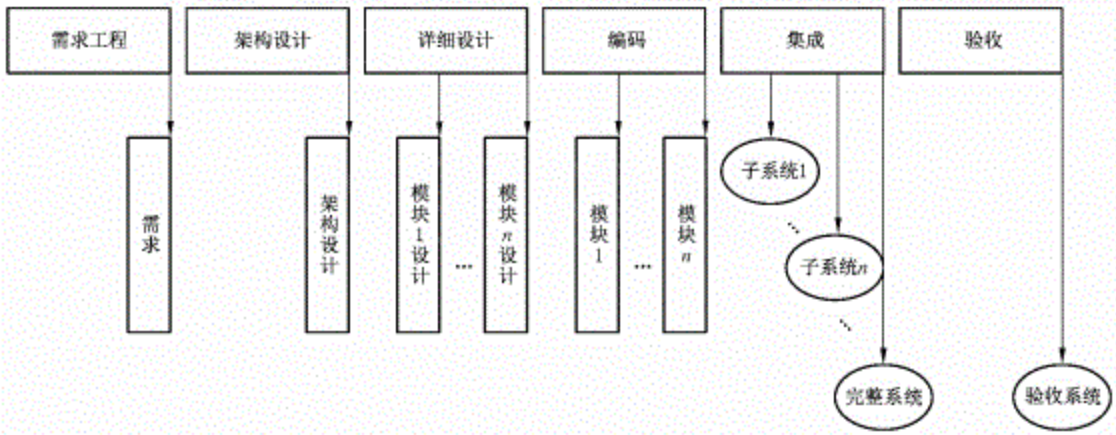

图 7 开发子过程示例

每个工作产品、软件系统组件和完整的软件系统都是潜在的测试项。

请注意，开发模型的定义超出了本标准的范围。图 7 中描述的阶段仅是说明测试如何应用于开发的示例。

4.3.3 持续维护及其结果

持续维护在软件生存周期的运营期间进行，即在初始开发之后进行，并且该过程可以在应用管理或 IT 服务管理过程框架的周境中进行管理。项目可以设置为持续维护，这与开发项目完全不同，例如财务模型可能不同。常见的财务模型是在特定时期内预算用于维护的金额，这可在客户和维护组织之间的服务等级协议(SLA)中说明。

持续维护过程通常关注保持系统可接受或商定的可靠性和可用性水平。会涉及系统新版本的生产，其中已修改了运行中发现的最高优先级缺陷，并可能引入对该功能高优先级的变更。当选定的修正和变更完成时，或基于固定频率(例如每 3 个月)，这些版本可以作为“实时”系统临时发布。如果是临时

发布的维护，则会包含选定的修正或变更，通常会尽快实施维护和发布。如固定周期的版本维护，则可以在该时间段内根据商定的优先级进行修正或变更。各发布阶段的输出示例如图 8 所示。

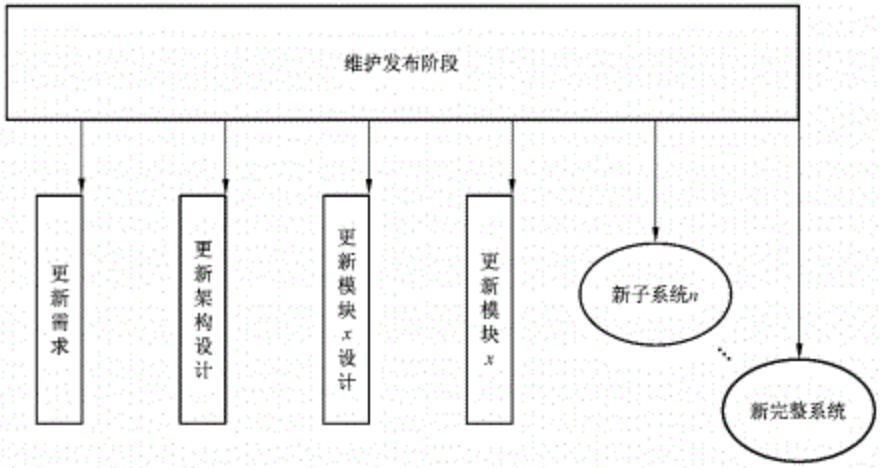

图 8　维护发布阶段示例

根据发布目标的不同，图 8 中所示的每个输出可以在单个发布时段中产生，或者依据需求提供部分输出。例如，临时修正可能不会影响需求和设计，而只影响编码和已完成的系统；也可能是所有工作产品在系统准备发布之前受多次影响。

可依据需要，在系统的运营生存周期内重复维护发布阶段。

4.3.4　软件开发生存周期的支持过程

4.3.4.1　概述

在组织内，需要支持过程来帮助软件开发生存周期。其中包括：

——质量保证；

——项目管理；

——配置管理；

——过程改进。

4.3.4.2　质量保证和测试

质量保证是一套计划的、系统的支持过程和活动，用以提供足够的信心，相信过程或工作产品满足既定的技术或质量要求。其通过加强方法、标准、工具和技能来实现的，这些方法、标准、工具和技能被认为是周境中的适当实践。质量保证的过程使用测试结果和其他信息来调查、排序和报告软件工程过程设计、策划或执行中的任何问题(包括风险)。

应在测试过程中收集测度，该测度可提供有关测试过程、测试项质量信息及其在每个项目中的应用效果。

4.3.4.3　项目管理和测试

项目管理指用于策划和控制项目进程的支持过程，包括整体项目中的测试管理。无论谁负责各个过程，项目管理都和测试管理过程密切相关，如图 9 所示。

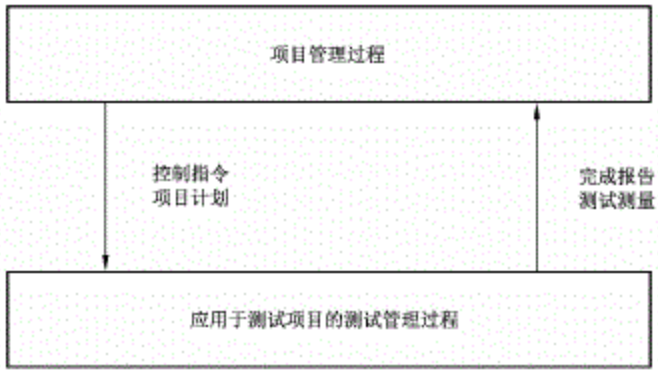

图 9　整体项目和测试项目的关系

测试活动的估算、风险分析和进度安排应与整体项目策划统一。项目计划是项目管理过程中的信息项，因此在用于管理测试项目时，它是测试管理过程的输入。

在测试项目过程中，测试经理会分析从详细的测试活动中收集的测量结果，并将其传达给项目经理，以便在项目周境中进行分析。这可能会导致影响测试项目的项目计划变更，更新的项目计划和相应的指令应发布到测试项目中，以帮助确保测试项目得到控制。

当测试子过程或测试项目完成时，向项目经理提供一份总结测试子过程或测试项目的过程和结果的完成报告。

4.3.4.4　配置管理和测试

配置管理是测试交互的另一组支持过程。配置管理的目的是建立和维护工作产品的完整性。好的做法是在运营使用之前对组织或项目的配置管理系统进行测试，以确定其是否符合组织或项目要求。

配置管理过程包括产品周期内所选工作产品、系统组件和系统的唯一标识、受控存储、发布审核、变更控制和状态报告。配置管理的对象称为配置项。配置管理过程是事件驱动的，即都是依据各自的过程独立启动的。

测试过程中的工作产品可以纳入配置管理，包括：

——组织级测试规范(例如测试方针、组织级测试策略)；

——测试计划；

——测试规范；

——测试环境配置项，如：测试工具、测试数据(库)、驱动、桩模块。

本标准中定义的测试过程模型中的三个层次都可以与配置管理进行交互。提供给测试过程的配置项是过程需要作为输入并纳入配置管理的工作产品。从测试过程传递的配置项是测试过程产生的工作产品，其需要纳入配置管理。

例如，组织级测试过程可以产生测试方针和组织级测试策略，其被纳入配置管理。项目测试经理可以从配置管理中获取项目计划，并作为项目测试计划的基础，随后该项目测试计划纳入配置管理。执行动态测试子过程的测试人员可以从配置管理中获取需求规格说明，并作为测试规格说明的基础，该测试规格说明随后纳入配置管理。

为了能够复现问题以便进一步分析，最好的做法是(在可能的情况下)使配置管理系统足够全面和健壮，以便在将来的任何时候都可以在恰好与以前相同的条件情况下重复测试。只要在组织级测试策略或项目计划中明确异常，就可以从重复性需求中排除某些类型的测试，例如单元测试。

测试过程的配置管理报告宜提供分析事件进度和状态所需的详细措施。

4.3.4.5 过程改进与测试

过程改进是采取措施来完善组织的过程，以便更有效地满足组织的业务目标。

测试过程和过程改进过程以两种方式相互作用：

a) 测试过程提供过程改进操作可参考的信息(以及从其他来源获得的信息)；

b) 测试过程本身可以进行过程改进。

当测试为过程改进提供信息时，通常在项目级应用的测试管理过程和过程改进过程之间进行交互。测试测量可以直接从测试管理过程进行传递。如果配置管理包含了变更控制，过程改进过程还可以从配置管理中获取一些与测试相关的度量。

对测试过程的过程改进，可在组织范围内进行，也可在独立于整个组织测试过程内进行。在任一情况下，过程改进过程都宜从多个来源获取信息，以诊断要改进的测试过程以及如何改进，并定义改进的测试过程。所选测试过程会产生新的版本，然后将其推广到相应的用户。确定的改进可能仅适用于给定项目，也可以推广到组织中的所有项目。宜对新的测试过程进行监测，以评价它们是否产生预期的投资回报。

4.4 基于风险的测试

4.4.1 概述

对软件系统进行穷尽测试是不可能的，因此测试是一种抽样活动。本部分讨论和概述了现有各种测试概念(例如实践、技术和类型)以帮助选择合适的样本进行测试。本部分的一个关键前提概念是使用基于风险的方法在给定的约束和周境中执行最优测试。

这是通过识别不同测试策略的相对价值来实现的，包括为利益相关方缓解已完成系统和开发系统的风险。执行基于风险的测试可确保在测试期间最高优先级的风险得到最高的关注。一些其他标准可帮助确定运行中系统的风险，例如 ISO/IEC 16085 风险管理。

风险可以通过多种方式进行分类。例如，无法满足法规法律的要求、未能履行合同义务、不成功的项目进展和完成，以及没有达到预期行为的工作产品(产品风险)。

在执行基于风险的测试时，风险分析用于识别和评价风险，以便对交付的系统(以及该系统的开发)中识别的风险进行评分、排序和分类，随后进行缓解。

4.4.2 组织级测试过程中使用基于风险的测试

组织级测试策略为组织中的测试设定周境。在这样做时，宜识别可能影响测试过程的组织风险。例如，生产医疗软件的组织可能需要遵守严格的质量标准，该组织的组织级测试方针可能包括处理安全相关标准的要求，从而尝试减轻业务失效时的影响。

组织级测试策略宜提出在测试过程中管理风险的方法。在上述生产医疗软件组织的示例中，组织级测试策略可以提供在组织中执行测试的指南，以助于管理不符合监管标准的风险。除此之外，组织级测试策略可以强制要求采用特定方法进行测试管理过程中的风险管理。

4.4.3 测试管理过程中使用基于风险的测试

作为测试计划的一部分，测试策略描述了已经过分类的风险概况(已识别的风险集及其评分)，用于确定需对项目执行哪些测试。例如，风险评分可用于确定测试的严格程度(如测试子过程、测试设计技术、测试完成准则)。风险的优先级可以用来确定测试时间表(通常较早处理高优先级风险相关的测试)。风险类型可用于决定最合适的测试形式。例如，如果组件之间的接口存在可感知的风险，则有必要进行集成测试；如果感知到存在用户发现应用程序难以使用的风险，那么有必要进行易用性测试。

好的做法是尽可能广泛地让利益相关方参与这些活动，以确保识别出尽可能多的风险并确定其各自的缓解活动(或者决定不处理这些风险)。

该方法的好处是，在任何时候，交付风险都可以作为残余风险简单地传达给利益相关方。

针对已感知的风险以及不断变化的业务情况进行测试，自然会导致风险概况随时间变化，因此需将基于风险的测试视为持续的活动。

4.4.4 动态测试过程中使用基于风险的测试

风险概况为所有动态测试过程提供指导。在测试设计过程中，产品风险用于帮助测试人员选择最为合适的测试设计技术。同时，还可用于指导在测试设计过程中进行多少测试分析，高风险域比低风险域需要更多的关注。风险也可以用于确定特征集的优先级，并且可以导出测试条件和测试用例。

动态测试执行是一种风险缓解活动。运行测试用例可以缓解风险，因为如果测试通过，则风险发生的可能性通常较低，因此风险评分会降低。同样，如果测试失败，风险则可能会增加。然后使用测试事件报告过程来确定失败的测试用例是否导致了问题或是否需要进一步检查。

4.5 测试子过程

4.5.1 概述

基于项目测试策略通常会将测试项目构建成多个测试子过程。测试子过程可以与软件生存周期阶段相关联，和/或关注于特定质量属性。测试子过程可包括静态测试和动态测试。

测试管理过程贯穿每个测试子过程。测试子过程从测试策划开始，测试监测和控制活动贯彻于整个测试子过程，监测活动获取的信息可能会导致策划进行适当重审。

测试策划包括确定测试目标、测试范围、测试子过程相关的风险。测试策划指导了测试子过程的策略，包括为测试子过程策划的静态测试和动态测试。在某些情况下，项目测试策略中概述的内容可以直接用于测试子过程的策略，在其他情况下，测试子过程的具体策略应基于要处理的特定风险。

在描述测试子过程时，重要的是要指定测试项的特征是否需要测试。这是为了确保清楚正确地理解测试范围的预期。

测试子过程很少只包含一轮动态测试或静态测试。如果仅定义了每种类型的一轮测试，则可能需要重复该过程，以满足测试完成准则。重复动态测试过程通常称为复测和回归测试。可以执行复测和回归测试，以确定对工作产品已有缺陷所做的更改，且不会引起进一步的缺陷(更多信息见 4.5.6)。图 10 给出了测试子过程的示例。

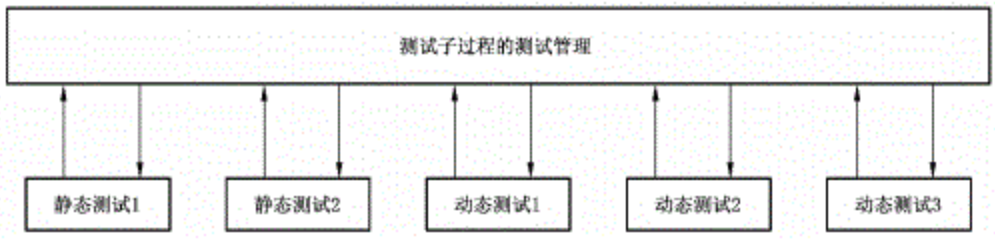

图 10 测试子过程的通用示例

测试项目中的测试子过程数量取决于测试策略和为整个项目定义的生存周期阶段，但不依赖于开发生存周期(即，瀑布或螺旋式开发周期不能确定其自身所需的测试子过程的数量)。

附录 D 详细介绍了测试子过程的示例。

测试目标、测试项目、测试依据和风险是每个测试子过程特有的，用于指导测试子过程中所执行的测试活动和所使用测试设计技术的选择。下面给出了测试目标、测试项、测试依据和测试设计技术的

示例。

4.5.2 测试目标

测试的执行用于实现一个或多个目标。本标准所涵盖的测试目标包括：

——向风险管理活动提供信息；

——提供有关产品质量的信息；

——评价产品是否符合利益相关方的期望；

——评价缺陷是否已正确修复，同时未引入其他不良影响；

——评价变更是否正确实施，同时未引入其他不良影响；

——评价需求满足的情况(如：规范、设计、合同等)。

测试是用以完成特征或特征集的测试目标。所测的特征类型将决定所需的测试类型。特征可用于体现需满足的质量特性。测试者通过特征或特征集所组成的质量特性来确定达到测试目标所需使用的测试类型。

一些测试目标可能仅与特定测试子过程或测试项类型相关。确定测试项的相关测试目标有助于确定合适的测试子过程。例如，在测试商业现货产品时，评价缺陷修复的测试目标可能不适用了，因为供应商已经完成了健壮性测试并修复缺陷。因此，在这种情况下采用验收测试子过程以满足测试目标，即实施变更而未引入其他不良影响。

4.5.3 测试项

应依据测试项的预期对测试项进行测试；这种预期是由测试依据(见 4.5.5)描述。测试项是诸如管理、开发维护、测试本身或其他支持过程等活动的结果。

测试项示例包括：

——代码相关测试项：

- 可执行的软件组件；
- 子系统；
- 完整系统。

——文件相关测试项：

- 计划，例如项目计划、测试计划或者配置管理计划；
- 需求规格说明；
- 架构设计；
- 详细设计；
- 源代码；
- 手册，例如用户手册或安装手册；
- 测试规格说明和测试规程。

上述列表中的最后一条表明，尽管测试规格说明和测试规程是由测试人员开发的，但也应接受测试(静态测试)，因为它们并非没有缺陷。

4.5.4 质量特性的测试

GB/T 25000.10—2016 系统与软件质量模型概述了软件的质量模型。该模型描述了八个质量特性，用于定义测试项目的质量属性。测试是一种测量给定测试项的重要质量特性的活动。八个质量特性如下：

——功能性：在指定条件下使用时，产品或系统提供满足明确或隐含功能的程度。

——性能效率：性能与在指定条件下所使用的资源量有关。

——兼容性：在共享相同的硬件或软件环境的条件下，产品、系统或组件能够与其他产品、系统或组件交换信息，和/或执行其所需功能的程度。

——易用性：在指定的使用周境中，产品或系统在有效性、效率和满意度特性方面为了指定的目标可为指定用户使用的程度。

——可靠性：系统、产品或组件在指定条件下、指定时间内执行指定功能的程度。

——信息安全性：产品或系统保护信息和数据的程度，以使用户、其他产品或系统具有与其授权类型和授权级别一致的数据访问度。

——维护性：产品或系统能够被预期的维护人员修改的有效性和效率的程度。

——可移植性：系统、产品或组件能够从一种硬件、软件或其他运行（或使用）环境迁移到另一种环境的有效性和效率的程度。

为了测试质量特性，需要具体化测试子过程。例如，策划和执行测试用以测量信息安全性可能需要实施信息安全性测试子过程（安全测试）。

每一个质量特性具有多个子特性，可以对其进行测试以提供质量的整体视图。另外，并非所有质量特性都适用于所有系统，例如，可移植性对于专用嵌入式系统可能并不重要。注意，上述质量特性不一定是详尽的，可为特定测试项定义更为适合的质量特性。

测试从业人员通常将功能性的测试称为“功能测试”，并将其他质量特性的测试称为“非功能性”测试。用于测量功能性以外的质量特性的测试类型通常被称为非功能性测试，可包括负载测试、压力测试、渗透测试、易用性测试等。

在制定测试计划时，测试经理应考虑所有质量特性。针对不同的质量特性及其子特性的测试重点可能会根据以下情况而变化：

——正在开发系统的风险概况：例如，安全关键应用可能会强调可靠性。

——正在开发系统的业务门类：例如，银行应用程序可能会强调信息安全性。

4.5.5 测试依据

本标准中，“测试依据”是指策划、设计、推断、实现和管理测试项所需的任何形式的知识体系。这可以包括选择测试行为、通过-不通过准则、输入、环境、实践和技术。测试依据的性质也随着开发生存周期阶段的不同而变化。

测试依据的示例包括：

——文档格式和内容的预期，通常是标准和/或检查列表的形式。

——软件系统（新的或现有的）客户/用户的预期，通常是书面的需求规格说明的形式。可表现为含有“应”的条款、用例、用户故事或其他形式的非正式或书面的功能/非功能描述的需求。可能还包括某些类型产品应遵守的规范标准，例如制药行业或火车或飞机等运输系统的安全关键软件。

——测试人员或拥有与用户需求和产品历史相关的其他领域的专家知识。

——软件系统组件之间的直接或间接接口和其他软件系统组件共存的期望，通常采用诸如图表或书面编写协议描述架构设计的形式。

——在编码中实现软件系统组件的期望，通常是详细设计的形式。

注：一个测试子过程中的测试项，可作为另一个测试子过程中的测试依据，例如，需求规格说明可以是静态测试子过程的测试项，也可以是后续系统测试子过程的测试依据。

需求可分为两个主要类别，即：

——功能性需求：指定该项目应做什么，与 GB/T 25000.10—2016 系统与软件质量模型中的功能性保持一致；

——非功能性需求：指定功能应如何表现，与 GB/T 25000.10—2016 系统与软件质量模型中其他

质量特性保持一致。非功能性需求与部分或所有功能相关，并且通常功能性需求可与单独或成组的非功能性需求相关联。

4.5.6 复测和回归测试

复测是评价事件的解决方案是否已纠正原有问题的测试。复测通常通过重新运行产生原有问题非预期结果的测试用例来执行。然而，为了有效地进行复测，可识别、分析新的测试条件并编写新的测试用例。

回归测试是对先前已经过测试的系统或组件进行选择性测试，以验证变更未引起意外影响，并且系统或组件仍然符合其原有需求。回归测试的目的是确定变更未在系统的未改变部分中引起缺陷。在策划测试时应考虑回归测试和复测，因为通常同时需要两者来完成子过程的测试。测试执行进度安排宜考虑两者所需的时间。

4.5.7 测试设计技术

4.5.7.1 概述

测试设计技术包括静态测试技术和动态测试技术。测试设计技术的目的是帮助测试人员尽可能有效和高效地发现测试项的缺陷。静态测试通常通过识别文档测试项中的明显缺陷(“问题”)或源代码中的异常来实现。动态测试通过执行测试项引起失效来实现此目的。

4.5.7.2 静态测试设计技术

静态测试可以包括各种活动，如静态代码分析、跨文档的可跟踪性分析和审查有关软件产品的其他信息。

静态测试是一种测试形式，通常作为测试策略的一部分，因此也属于本标准的范围。其他标准可用于完整地描述静态测试技术。本条仅介绍静态测试的概念，并给出与其他标准的关系。

GB/T 32421—2015 定义了软件开发中的审查技术和过程。GB/T 32423—2015 定义了验证与确认活动(参见附录 A)。任何静态测试设计技术的主要目的是发现缺陷，但也有其他目的，如：走查用于知识交流，技术审查以获得共识。评审有助于预防未来的缺陷。在任何特定情况下选择的静态测试设计技术的类型并不依赖于测试项的类型，而是依赖于涉及的风险(风险越高，则推荐更正式的静态测试设计技术)与静态测试设计技术的次要目的。

虽然静态测试通常属于组织或项目软件测试的活动范围，但是本部分就上述相关内容进行描述，其他部分不再阐述静态测试的技术和过程。

4.5.7.3 动态测试设计技术

动态测试的测试设计技术用于识别测试条件、测试覆盖项以及随后在测试项上执行的测试用例。动态测试设计技术根据测试输入的导出方式分为三类。分别是基于规格说明、基于结构和基于经验的技术。GB/T 38634.4 详细描述了每类动态测试设计技术。

基于规格说明的测试设计技术用于从描述测试项的预期行为的测试依据导出测试用例。使用这些技术，从测试依据中导出测试用例测试的输入和预期结果。特定情况下选择哪种基于规格说明的测试设计技术取决于测试依据和/或测试项目的性质和所涉及的风险。GB/T 38634.4 中涉及的基于规格说明的测试设计技术示例包括边界值分析、状态转移测试和判定表测试。

基于结构的测试设计技术基于结构特征导出测试用例，例如源代码结构或菜单结构。如果将这些技术用于应用程序源代码，则测试用例的预期结果将从测试依据导出。在任何特定情况下选择使用哪种基于结构的测试设计技术取决于测试依据的性质和所涉及的风险。在 GB/T 38634.4 测试技术中详

细定义和描述了该类技术。GB/T 38634.4 中涉及的基于结构的测试设计技术的示例包括分支测试、状态测试和数据流测试。

4.6 测试实践

4.6.1 概述

4.4 中所描述基于风险的测试已被广泛采用，是本标准的基本方法。有许多不同的实践用于策划和实施项目的测试。传统的实践是依据需求进行测试，在测试执行前人工设计测试用例，并混合使用人工和自动化测试来执行。使用基于风险的测试并不意味着不能使用这些实践作为声称符合GB/T 38634.2的测试计划的一部分。测试策略的选择取决于各种风险，例如组织、项目和测试项目的风险。例如，如果组织无法确保每项要求都经过测试，则可能面临违反合同的风险。因此，使用基于需求的实践将是管理组织风险的方法。本条介绍了多个测试实践，每个测试实践都可以作为符合GB/T 38634.2创建的测试策略的一部分。通常这些测试实践中的任何一种都不可能单独使用，而是用作更大的测试策略的一部分。

本条阐述了当前使用的一些测试实践，以演示一些可在测试计划时使用的选项。

4.6.2 基于需求的测试

基于需求的测试主要目的是确保在测试中解决(即“覆盖”)测试项的需求，以确定测试项是否满足最终用户需求。此外，还用于在生存周期中收集并向利益相关方提供其他有价值的信息，例如测试项中标识的缺陷。在使用此实践时，需求为测试计划、测试设计和实现以及执行过程的决策提供信息。注意，基于需求的测试可以部分用于完成 ISO/IEC/IEEE 12207 中概述的需求验证。

基于需求的测试主要使用脚本测试。测试用例是在测试执行之前，在测试设计和实现过程中编写的。然后根据测试过程中定义的计划，在测试执行过程中执行测试用例。随后对测试执行结果进行分析，完善测试，可能还会有更深入的测试设计和测试用例脚本。在此测试设计期间所创建的测试用例将被记录，并在此测试执行生存周期后期执行。

其他测试实践可以支持基于需求的测试，用以证明需求已经过充分的测试。除了使用基于需求的测试以确保满足所有需求之外，还可以采用其他实践(即基于经验的测试)来解决其他风险。例如，交付产品中常常存在回归缺陷。或许探索性测试可以解决这些回归风险。

使用基于需求的测试的实用性是高度依赖于周境的。如果需求不完整或不一致，则测试可能会受此影响。即使需求明确，但预算和时间限制可能意味着无法测试所有需求。当需求补充有相关优先级的信息时，应重新确定测试优先级(在这种情况下，可以使用基于风险的方法来确定优先级较高需求的优先级)。在实践中，使用基于需求测试的测试人员通常会以这种方式补充基本需求，以便更彻底和更早地测试最重要(最高风险)的需求。

4.6.3 基于模型的测试

所有的测试都使用模型的概念来表示测试项的预期行为可用作测试依据。该模型可以用自然语言表述的需求、思维图、数学公式、图形符号(例如状态转移图、UML 图)或矩阵(例如决策表)的形式表示，或者混合这些方式来表示。传统上，测试人员使用模型人工导出执行基于规格说明测试的测试输入和预期结果，以及执行基于结构测试的预期结果(此处测试输入是通过测试项结构得出的)，随后人工或使用测试工具执行测试。

基于模型的测试虽然也是基于预期行为的模型，但其实践从根本上不同。不同之处在于，对于基于模型的测试，模型应充分形式化、足够详尽，以便从模型中导出有用的测试信息。模型信息可包括测试计划、设计、规程、输入、风险、结果参照物。使用基于模型测试的优点包括可生成测试信息、提高测试生

存周期中自动化水平(计划到文档),以及能尽早识别某一些错误类型。这种自动化水平可在相对较短的时间内自动生成、执行和检查数百万个测试用例。

使用基于模型的测试的典型周境是针对关键应用程序,其失效可能导致很大的损失,并且应用程序的需求行为适合由基于模型的测试工具进行建模(熟练工可以创建和更新模型)。UML 图形符号通常用作这些模型的基础,也可采用其他符号。使用基于模型测试的另一个考量因素是应用程序维护过程中,修改模型将比修改自动化测试脚本更容易(每次更新模型时,基于模型的测试工具随后可以相对较低的成本导出并运行测试)。

模型的使用可以为其他领域提供益处。例如,基于模型的自动源代码生成可以减少在开发期间出现的某些类型的缺陷(即错误的代码和可以通过模型检查检测到缺陷)。这有助于减少测试过程中开发过程引入测试项的缺陷数量。

4.6.4 基于数学方法的测试

当可以充分详细地描述测试项输入或输出范围时,基于数学方法的测试实践可用于计划、设计、选择数据和设置输入条件。数学方法实践使用数学分支来推动。这些实践有助于减少测试用例选择和优先级排序中人为因素的影响。

基于数学方法的实践用到了多种技术,特别是以下测试设计技术(在 GB/T 38634.4 中描述):

——组合测试;

——随机测试用例选择。

此外,统计建模可用于确定测试覆盖率(未在 GB/T 38634.4 中提及),例如:

——分层抽样;

——模糊测试;

——聚簇抽样;

——专家判断抽样。

基于数学方法的测试实践,也可使用基于数学的工具和技术客观地对测试用例进行采样。

使用数学方法的实践通常会产生大量的输入,因此通常需要使用自动化工具来实现。

4.6.5 基于经验的测试

基于经验的测试是基于:

——以往的测试经验;

——特定的软件和系统知识;

——领域知识;

——以往项目(在组织和行业内)的指标。

GB/T 38634.4 描述基于经验的测试设计技术中的错误猜测。其他基于经验的测试方法包括(但不限于)探索性测试、软件攻击和特别测试。探索性测试和软件攻击不作为 GB/T 38634.4 所包含的测试设计技术。因为这两种技术应用了多种测试设计技术。

探索性测试压缩了 GB/T 38634.2 中“测试设计和实现”和“测试执行”过程描述的活动,以便动态执行这些活动完成测试目标。当使用探索性测试时,测试用例通常不会提前设计和记录,测试人员使用直觉、好奇心和先前测试的结果来决定接下来测试的内容和方法。

探索性测试、错误猜测和特殊测试是不依赖于大量文档(例如测试规程)来执行测试的测试实践。在脚本测试到无脚本测试中,基于经验的测试实践主要是无脚本测试。使用这种方法可能只能符合 GB/T 38634.2 的剪裁符合性。

4.6.6 脚本测试和无脚本测试

表 1 描述了脚本和无脚本测试的优点和缺点。但应该指出的是，这两种方法在项目上并不相互排斥；它们通常结合起来，根据测试项的风险等级混合实践。

在决定是使用脚本测试、无脚本测试还是两者的混合时，主要考虑的是测试项的风险情况。例如，混合实践可能使用脚本测试来测试高风险测试项，而无脚本测试来测试同一项目中的低风险测试项。

表 1 测试执行中的脚本和无脚本实践的对比

测试方法	优点	缺点
脚本测试	测试是可重复的：测试用例可以简单地再次运行，从而为验证与确认活动提供良好的可审计性。 脚本化的测试用例可以明确追溯到需求，允许将测试覆盖率记录在跟踪矩阵中。 测试用例可以保留以便在当前和将来项目中重用，可减少将来测试设计和实施所需的时间	通常比无脚本的测试执行更耗时且成本更高；但是，如果脚本化的测试用例可重用，则可能会随着时间的推移而节省成本。 在测试执行之前定义的测试用例不太能够适应系统本身。 由于大多数分析工作已经完成，因此可以减少对测试执行者的激励。这可能导致测试人员在测试执行中心不在焉并忽视细节
无脚本测试	测试人员不受脚本约束，可以实时跟踪测试执行产生的想法。 测试人员可以实时定制“测试设计和实现”和“测试执行”以适应系统的行为。 测试项可迅速执行	测试一般不可重复。 测试人员应能够根据需要应用各种测试设计技术，因此有经验的测试人员通常比经验不足的测试人员更容易找到缺陷。 无脚本测试几乎不提供已完成的测试执行记录。因此，除非使用工具来捕获测试执行，否则很难测量动态测试执行过程

4.7 自动化测试

GB/T 38634.2 中测试管理过程和动态测试过程中描述的多个任务和活动，以及 GB/T 38634.4 中的测试技术，均可能被自动完成。自动化测试可用于支持 4.6 中描述的测试实践(例如，基于模型的测试几乎总是基于自动测试执行工具的使用)。自动化测试通常需要使用测试工具软件。自动化测试主要关注脚本测试的测试执行，而不是让测试人员人工执行测试，但是基于软件的工具可以支持许多额外的测试任务和活动。以下提供了测试工具涵盖的一些领域的示例：

——测试用例管理；
——测试监测控制；
——测试数据生成；
——静态分析；
——测试用例生成；
——测试用例执行；
——测试环境实现和维护；
——基于会话的测试。

可供使用的自动化测试工具有多种，其可通过内部开发、商业购买或从开源社区获取。

4.8 缺陷管理

项目测试经理通常负责项目的缺陷管理(也称为事件管理)。GB/T 38634.2 中包含了缺陷管理，用

以帮助测试人员记录测试事故报告和在必要时对缺陷进行重新测试。本标准未直接涵盖缺陷管理的其他部分。缺陷管理概念和过程可以在以下标准中获取：ISO/IEC/IEEE 12207、ISO/IEC 20000 和 GB/T 32422—2015。

附 录 A
（资料性附录）
测试在验证和确认中的作用

图 A.1 定义了一种验证与确认(V&V)活动的分类。V&V 可以在系统、硬件和软件产品上完成。这些活动在 GB/T 32423—2015 和 ISO/IEC/IEEE 12207 中进行了定义和细化。许多 V&V 都是通过测试完成的。GB/T 38634 解决了软件的动态和静态测试(直接或通过引用),因此涵盖了测试和验证的部分内容。GB/T 38634 并没给出 V&V 模型所有的元素,但对测试人员来说,理解测试在 V&V 模型中的位置很重要。

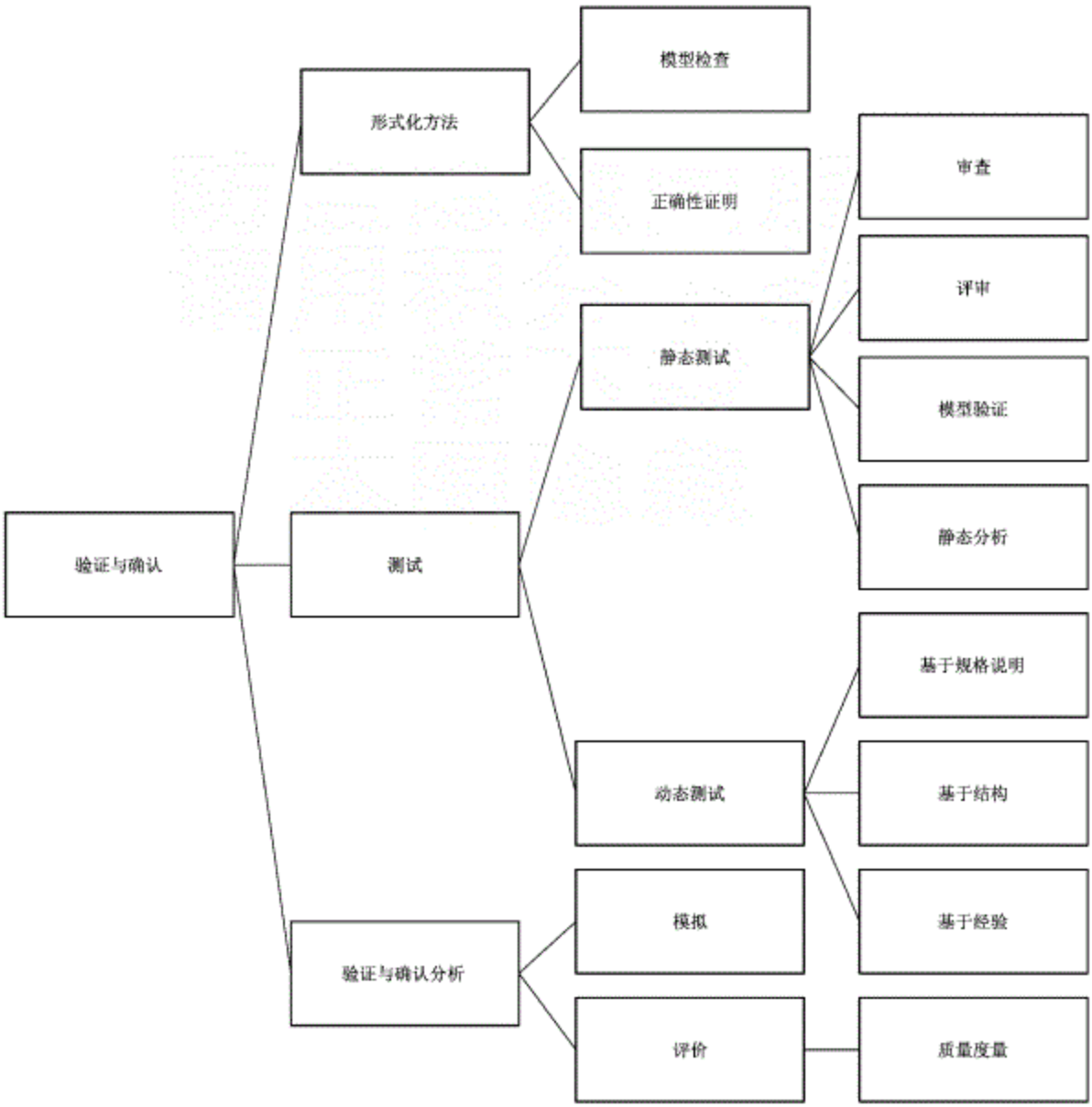

图 A.1 验证和确认活动的层次结构图

附 录 B
（资料性附录）
度量和测度

测试的主要目的是提供有助于管理风险的信息。为了监测和控制测试并及时向利益相关方提供信息，有必要对测试过程进行有效地测量。为了测量测试过程，有必要定义要提供的信息，以及如何获得和呈现这些信息。因此，所有的测试工作都需要定义和使用度量，并提供关于产品和过程的测度。

可以在测试中使用度量的示例如下：

——残余风险：减轻风险数量/识别的风险数量。

——累积缺陷打开和关闭：每天打开的缺陷数量与每天关闭的缺陷数量之比。

——测试用例进度：执行的测试用例数/计划执行的测试用例数。

——缺陷检测百分比：测试中发现的缺陷数/发现的缺陷数(总体)。

附 录 C
（资料性附录）
不同生存周期模型中的测试

C.1 概述

本附录给出了测试如何适用于不同的软件开发生存周期模型的示例。对于一个给定的项目，需要确定必要的开发阶段和/或活动，以及在开发生存周期过程中这些阶段和/或活动的相互关系。开发阶段及其关系的集合称为“开发生存周期模型”，或者更简单地称为“生存周期模型”。

多年来，软件行业已经使用了许多生存周期模型。这里按字母顺序（和它们的出现顺序相反）对典型生存周期模型进行了分类，但这不是一个完整的列表：

——敏捷；

——演化；

——顺序（即瀑布模型）。

在所有生存周期模型中执行的开发活动或多或少是相同的；主要区别在于它们范围的定义、产生的文档数量和性质以及它们在开发生存周期中重复的频率。

注：本部分的目的不是对不同的生存周期模型进行标准化，而是建立测试周境以助于实施这些生存周期。

C.2 敏捷开发和测试

C.2.1 敏捷开发原则

敏捷开发通常遵循以下基本原则：

——增量开发，每个周期交付有用和可用的产品；

——迭代开发，允许需求随着项目的开发而演进（即变更和添加）；

——以人为本的开发，依赖项目团队的质量（例如开发人员和测试人员）而不是明确定义的过程，允许敏捷团队自我管理；期待开发团队与业务利益相关方之间的日常互动；

——技术和工程卓越，通过严谨的方法来开发、集成和测试产品。

有许多敏捷开发的方式和框架，包括：极限编程（XP）、Scrum、水晶方法、看板和特征驱动开发。虽然它们有着不同的方法，但都具有基本敏捷原则，该原则体现在敏捷宣言中（见 http://agilemanifesto.org /）。由于本部分没有足够的时间和篇幅来提供使用每种不同的敏捷方法和框架实现示例，因此本部分将使用 Scrum 框架作为示例。Scrum 不是一种开发方法（也就是说，它不会告诉你描述需求或如何编写代码的最佳方法），而更适合描述为一个项目管理框架，在此框架中，开发人员应用了敏捷方法（经常使用极限编程）。

Scrum 项目包含许多称为冲刺的迭代，每次冲刺通常会产生可以交付给用户的新功能（见图 C.1）。这个新功能可能是对现有功能的增强或新功能的引入。每个冲刺通常持续一周至四周。通常，冲刺的数量在开始时是未知的，因为在典型的敏捷项目中，在项目开始时并未完全理解需求。随着项目的进展，需求也在不断变化。在产品待办事项列表中收集客户需求；通常表示为用户故事。

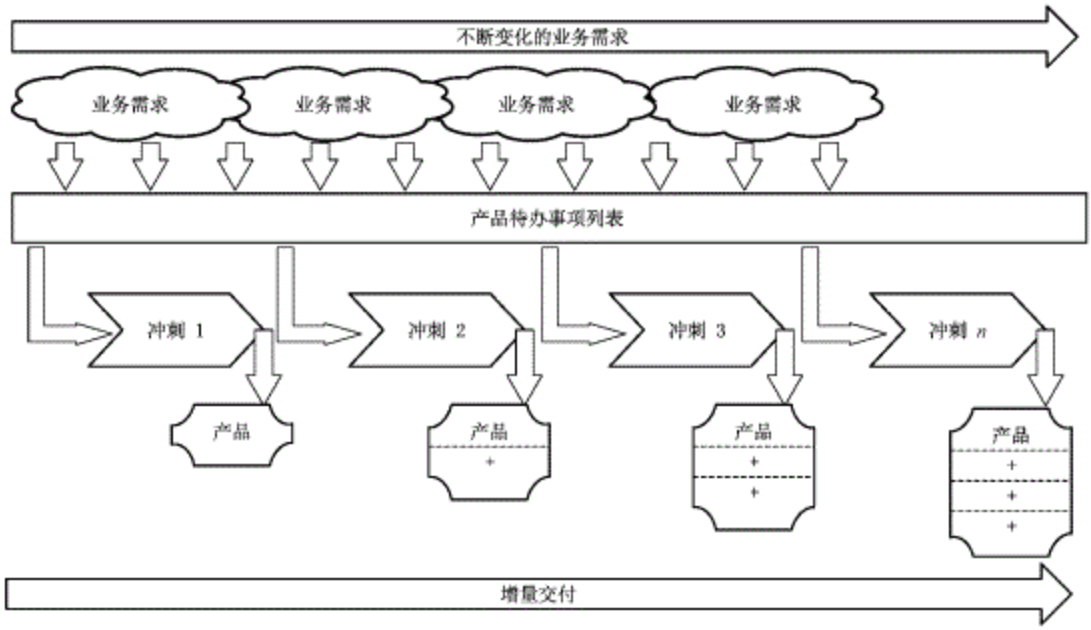

图 C.1　Scrum(敏捷)项目生存周期示例

本标准中定义的测试过程模型可以应用于遵循敏捷开发模型项目中执行的测试。

C.2.2　敏捷开发中的测试管理

组织级测试策略宜反映开发遵循敏捷开发模型这一事实。在使用包括敏捷在内的不同项目的混合开发模型进行开发的组织中,通常会有一个涵盖敏捷开发的特定组织级测试策略。使用敏捷开发的项目的组织级测试策略应该使用特定的敏捷词汇表,包括待办事项列表、冲刺和每日例会的概念,但除此之外,任何组织级测试策略的内容主要取决于涵盖的项目和产品的风险概况(尽管所使用的开发模型的类型可能会产生测试策略应解决的其他类型的风险)。

敏捷项目通常由项目经理(PM)管理,而冲刺由敏捷教练(Scrum Master)负责(这些角色可能由同一人执行)。敏捷项目中的测试管理是作为产品待办事项列表、单人冲刺和每日站会管理的一个集成部分来执行。

在冲刺开始时,Scrum 团队和客户(PO,产品负责人)同意应该在此冲刺中实现产品待办事项列表的哪些用户故事。选定的用户故事组成了冲刺-待办事项列表。然后,团队通过安排开发和测试活动以及分配团队成员的角色和职责来计划冲刺。敏捷在所有产品开发和测试中遵循相同的通用过程,Scrum 冲刺示例如图 C.2 所示。

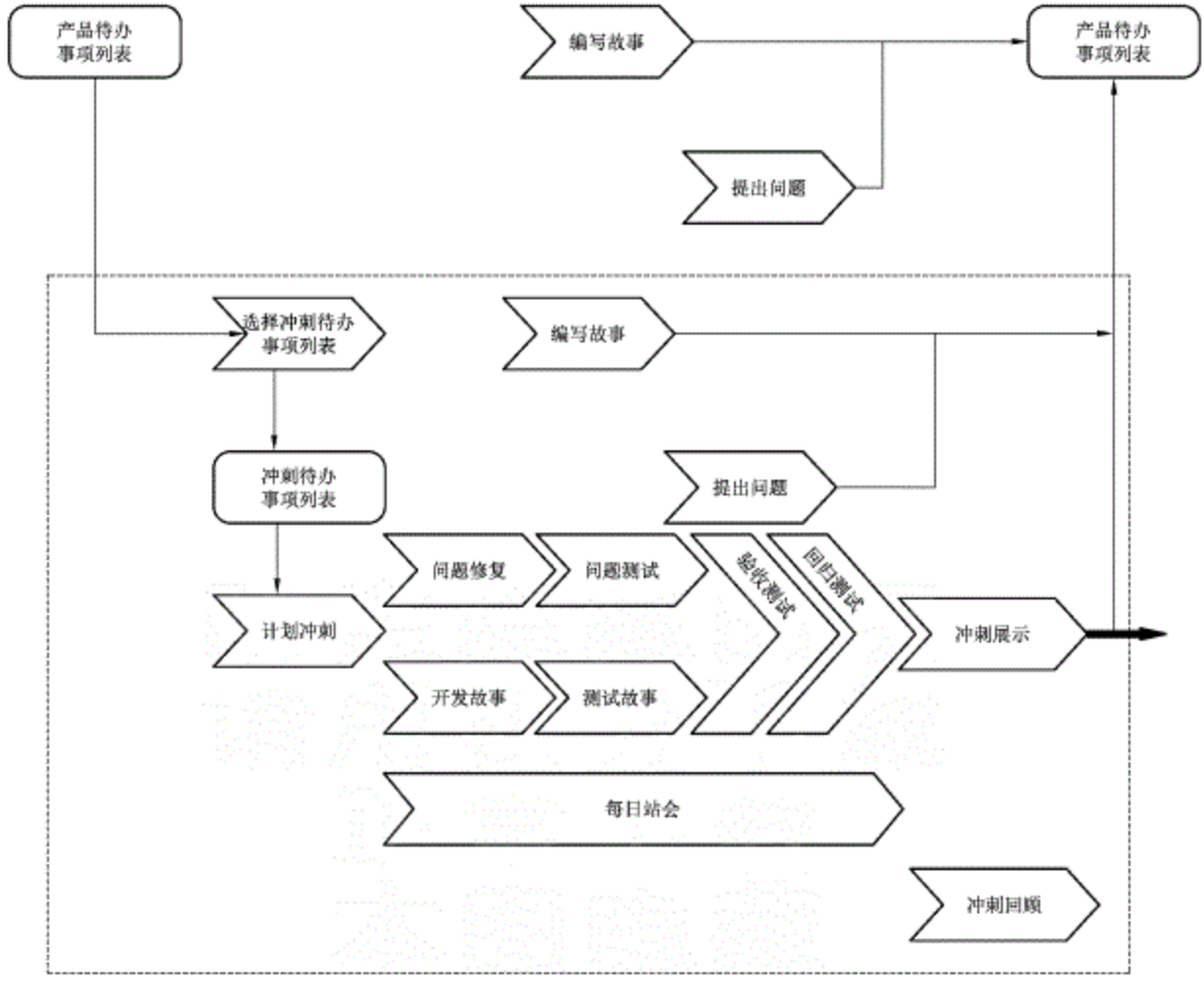

图 C.2　敏捷冲刺示例

在冲刺展示会上向客户展示了冲刺的可交付成果，团队有机会向利益相关方展示他们所创建的内容。冲刺的最后一项活动是冲刺回顾，团队评审他们的表现如何，并确定在未来的冲刺中可以进行哪些改进。因此在 Scrum 框架内集成了过程改进。

在整个冲刺期间，每天开始时都会举行每日站会，以确保整个团队都了解：已经做了什么，当天将做什么，以及团队面临的任何问题。他们还可以让敏捷教练决定需要移除哪些障碍以使团队更高效地前进。

敏捷项目中的考虑的关键因素是管理回归缺陷的风险（因为每个冲刺都基于之前的冲刺），并管理需求变化的本质及其对测试工件的影响。测试自动化通常用于管理回归风险，而探索性测试可用于管理缺乏详细需求的影响。

C.2.3　敏捷开发中的测试子过程

测试活动是敏捷开发项目的一个集成部分，如图 C.2 所示，测试在整个冲刺中持续进行。可以使用本标准中的过程来定义和执行测试子过程，以测试用户故事和正在交付的演进系统。

冲刺团队使用的典型测试实践有：

——测试驱动开发（TDD），这是在编写代码之前为代码编写测试。测试是基于用户故事，可由测试人员和开发人员共同开发。这些测试通常使用自动化组件测试工具来实现。这可能导致

TDD 被视为一种编程形式。

——自动化构建和持续集成的测试，它用在当系统在签入代码时不断更新和回归测试。用于确保及时识别和纠正集成和回归问题。

——所有质量特性(即功能和非功能)的系统测试都是针对用户故事和现有的任何高级别需求进行的。系统测试之后通常会进行验收测试，验收测试时应包括最终用户的参与，以确保交付的功能满足他们的需求。

——通常需要回归测试来确定当前冲刺中的任何变化对产品的现有功能和特征没有负面影响。

在“理想”的冲刺结束时，这些特征已经准备好供用户使用，这意味着上述测试都在冲刺中执行(如图 C.2 所示)。实践发现，在许多项目中要做到这点很困难，这导致会采用折衷的替代方法。例如，以并行但又互补的方式执行测试，或在一个临时的、以测试为主要目标的冲刺中执行测试。

C.3 顺序开发和测试

C.3.1 顺序开发原则

顺序生存周期模型存在时间最长，并且仍然被广泛使用。基本(和原始的)顺序模型称为瀑布模型，它由开发阶段描述，这些开发阶段按顺序排列在测试阶段之前，以最终的操作阶段结束。

顺序生存周期模型的特点是，除了后续阶段的反馈绝对必要之外，不包括明确的重复的阶段。

本标准中定义的测试过程模型可以应用于遵循顺序生存周期模型开发中的测试。

C.3.2 顺序开发的测试管理

组织级测试策略应该反映出开发遵循顺序开发模型。在使用包括顺序在内的不同项目的混合开发模型进行开发的组织中，通常会有一个或多个特定的组织级测试策略，涵盖所使用的开发模型。组织级测试策略应该使用它所涵盖的项目类型所使用的词汇表；此外，组织级测试策略的内容取决于其涵盖的项目和产品的风险概况，而不是所使用的开发模型。

顺序项目由项目经理管理。在大多数项目中，还定义了开发经理和测试经理的角色。根据项目的规模，这些角色可以由不同的人担任，或者两个或所有角色可以由同一人担任。

虽然测试计划应该在项目过程中制定，但在顺序开发中的这些计划是为整个项目而制定的。顺序开发项目可能规模相对较大，并且在开始时可能无法为整个项目以相同详细级别进行计划。此外，还应制定一个项目测试计划，这通常以单个文档形式出现，但可能是整体项目计划的一部分。如果在项目测试计划中指定要执行的测试子过程，可以生成单独的测试子过程计划。这些计划通常在顺序开发中要正式记录。

C.3.3 顺序开发中的测试子过程

C.3 描述的用于演化开发的测试子过程也与顺序项目中的测试相关，虽然它们只执行一次(只有一次通过顺序开发模型)。

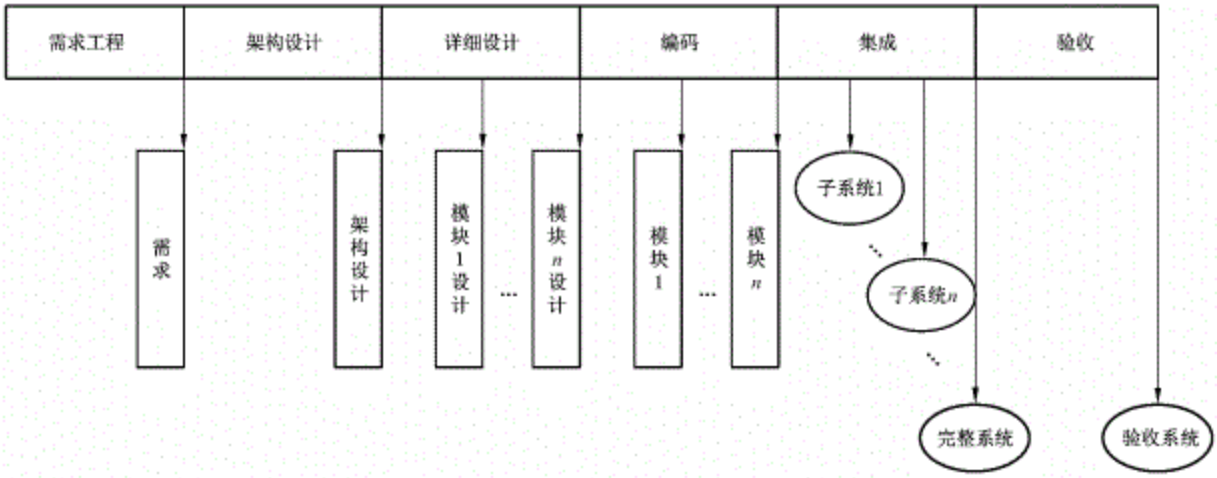

图 C.3　顺序开发生存周期模型中测试子过程示例

注意，图 C.3 未按比例绘制；所示的测试子过程的相对规模与测试子过程的实际规模无关。例如，所用时间、工作量或成本。

定义了架构设计测试、详细设计测试和代码测试。对于其中每一项，相应的开发阶段可以根据已完成的测试结果签署关闭，即当所有详细的开发项都单独进行了测试。

对于编码阶段，有两个不同的测试子过程。第一个是源代码测试子过程，由源代码的静态测试构成。第二个是组件测试，由可执行代码或可解释代码的动态测试构成。这些测试子过程可以结合，组件的动态测试依赖于成功的源代码静态测试。

集成阶段的最终结果是一个完整的系统。此时，假设系统测试的执行活动已经根据需求进行了计划和准备，那么就可以开始系统测试。

这里所提的测试子过程的示例将在附录 D 中进一步说明。

C.4　演化开发和测试

C.4.1　演化开发原则

演化开发基于迭代和增量交付的两个基本原则。迭代允许开发人员将系统开发为一系列较小的部分，然后他们可以根据对开发的产品以及开发实践的反馈，来改进下一次迭代。与传统的顺序周期相比，迭代能够更早地处理更高的风险，从而增加了处理风险的时间。随着增量开发的进行，每次迭代的结果都会发布给客户，这意味着用户可以在一系列的交付中获得了系统，并且系统具有越来越多的功能。

迭代由所有或部分标准开发活动组成。如果迭代的结果交付给用户，迭代将包括验收阶段，否则验收阶段通常只在最后一次迭代中执行。

图 C.3 显示了顺序生存周期。演化开发生存周期可以表示为许多独立的顺序生存周期，每个生存周期都为正在开发的系统增加了更多的能力。

本标准中的测试过程模型可以应用于遵循演化开发模型开发中的测试。

C.4.2　演化开发的测试管理

组织级测试策略应该反映开发遵循演化开发模型这一事实。在使用包括演化在内的不同项目的混合开发模型进行开发的组织中，通常会有一个或多个特定的组织级测试策略，涵盖所使用的开发模型。

组织级测试策略应使用其涵盖的项目类型使用的词汇表，但除此之外，任何组织级测试策略的内容主要取决于其涵盖项目和产品的风险概况(尽管所使用的开发模型的类型可能会产生测试策略应解决的其他类型的风险)。

演化开发项目由项目经理管理。在大多数项目中，还定义了开发经理和测试经理的角色。根据项目的规模，这些角色可以由不同的人担任，或者由同一人担任两个或所有角色。

演化开发中为整个项目和每次迭代制定计划。宜制定项目测试计划，可以采用单个文档的形式，也可以作为整个项目计划的一部分。还可以生成较小的迭代测试计划，指定要在迭代中执行的测试。计划可能在演化开发过程中或多或少地正式记录，但它们通常会被保存在文档和/或计划工具中。迭代测试计划和相关子过程测试计划通常会被重复使用，并在迭代和迭代之间进行必要的修正。

在迭代过程中监测测试进度，持续采取必要的纠正措施，并将更新的测试计划传达给利益相关方。

C.4.3 演化开发中的测试子过程

在每次迭代中，可以测试工作产品和完成的系统。可以使用本标准中的过程来定义和执行测试子过程，以测试可执行的工作产品和正在生产的系统。

本例中第一个开发阶段是需求工程，其中指定了业务、功能/非功能以及系统需求。图 C.3 只显示了与第一阶段(需求测试阶段)相关的测试子过程，以避免图像混乱。被指定的需求测试可以看作由静态测试组成的子过程。这个测试子过程的形式取决于产品的风险概况，但由于需求是迭代工作的基础，因此应该倾向于选择更正式的静态测试设计技术。

可以为两个设计阶段以及编码阶段定义类似的测试子过程。图 C.3 所示的开发模型示例可能包括：

——架构设计测试；

——详细设计测试；

——源代码测试。

这些测试子过程通常对应延伸到相关的开发阶段的大部分过程，它们集中于一种类型的测试项，例如需求，并包括测试项的不同测试方法，例如走查和技术评审。在根据测试子过程计划，在测试所有开发的需求之前，需求测试子过程不会完成。通常有一个或多或少的正式里程碑标志着需求工程阶段的结束，这通常取决于需求测试子过程的结果。

类似地，设计和源代码测试子过程直到根据测试子过程计划对所有开发的测试项进行了测试，才可能结束，并且相应的开发里程碑将取决于测试子过程的结果。

验收测试子过程如图 C.3 所示。如果在本次迭代中，需求发生重大变化的风险低于启动测试子过程的益处，则验收测试子过程中的计划和准备活动可以启动。测试设计将揭示需求中的缺陷，从而有助于提供关于需求的信息。对涉及动态测试的其他开发里程碑，可以定义类似的测试子过程。对于图 C.3所示的开发模型示例，这可能包括以下与验收子过程类似的子过程：

——组件测试；

——集成测试；

——系统测试。

在图 C.3 的示例中，还定义了性能测试。特定的测试子过程可能被定义为涵盖需求的特定类别或需求描述的系统的特定区域，通常取决于系统的风险概况。这种特定的测试子过程可能延伸到许多开发阶段，因此可能有各种测试项和相关的测试依据、测试职责、技术和环境，以及测试目标、完成标准和计划，尽管测试子过程只有一个重点，在这种情况下，性能需求以及它们如何在系统中描述、设计和实现。

上面描述的所有测试都可以在后续迭代中执行。

由于产品不断扩展，因此在每次迭代中应对之前已接受的内容进行广泛的回归测试。应为每次迭

代定义回归测试子过程。回归测试子过程可以包括迭代中扩展的所有项的回归测试，包括需求、设计、源代码和系统，或者可以根据风险概况只包含其中的一部分。对一个项的所有扩展类型，还可以单独定义回归测试子过程，以便从迭代到迭代的回归测试中获得更大的灵活性。

附 录 D
（资料性附录）
详细的测试子过程示例

D.1 概述

本附录列举了测试子过程的示例。这些示例以字母表的顺序排列。该列表只显示了一些示例；在特定的测试项目中可能需要许多其他的特定子过程。

示例如下：

——验收测试；
——详细设计测试；
——集成测试；
——性能测试；
——回归测试；
——复测；
——故事测试；
——系统测试；
——组件测试。

需要注意，回归测试和复测已经被包含在特定的测试子过程中。这些可以适当地包括在任何其他测试子过程中。

每个示例的测试子过程的描述包括：

——测试子过程的目标；
——计划的测试子过程内容，要执行的静态和动态测试过程。

对于计划用于测试子过程的每个静态或动态测试过程，描述包括：

——测试目标；
——测试项；
——测试依据；
——适用的测试过程；
——建议的测试设计技术（如适用）。

请注意，示例只供参考。在实际情况下，应根据产品的风险情况进行选择。

D.2 验收测试子过程

这些示例表示与开发生存周期的验收阶段相关联的测试子过程。

测试子过程目标：向客户演示最终系统在其指定要求方面的可接受性。

计划的测试子过程内容：动态测试 1：预演，
动态测试 2：演示。

动态测试 1：预演

测试目标：确保顾客在场的情况下最终执行成功。

测试项：已完成的系统。

测试依据：用户需求，用户手册，业务流程文档。

动态测试过程:测试设计和实现、测试环境建立和维护、测试执行、测试事件报告。

测试设计技术:用例测试,其他取决于需求性质的技术。

一旦用户需求稳定,测试用例的设计和实现就可以开始。虽然验收测试的假设是系统可以运行,但大多数组织在客户到场见证系统的正式演示之前就准备并执行测试,这便是人们熟知的预演。可以计划复测和回归测试过程以满足由于该测试而导致的任何"最后一分钟"的缺陷排除。

动态测试 2:演示

测试目标:向客户演示最终系统。

测试项:已完成的系统。

测试依据:不适用。

动态测试过程:测试环境建立和维护、测试执行和测试事件报告。

测试设计技术:不适用。

在最后的预演后需要重新设置环境,否则测试执行将按照预演中准备的规程和环境进行。

D.3 详细设计测试子过程

该示例表示与开发生存周期的详细设计阶段相关的测试子过程。

测试子过程目标:提供详细设计质量的信息。

计划的测试子过程的内容:静态测试 1:详细设计项文档,

静态测试 2:详细设计项易用性(不是系统的易用性),

静态测试 3:详细设计项完整性。

在详细设计阶段,根据架构设计会开发一些设计项。每项都可以接受为这个测试子过程定义的静态测试。因此,可以计划静态测试的实例,其中测试项被定义为特定的详细设计项。静态测试的实例可以与测试项目定义为一个具体的详细设计项目。详细设计测试子过程仅在所有计划的测试完成后(或根据实际情况放弃)完成。

静态测试 1:详细设计项文件

测试目标:提供关于详细设计项记录方式的信息。

测试项:详细设计项。

测试依据:内部和/或外部检查表,指定详细的设计文档规则。

测试设计技术:技术评审或审查。

静态测试 2:详细设计项可用性

测试目标:提供有关详细设计项目可用性的信息,例如编码或测试。

测试项:详细设计项。

测试依据:不适用。

测试设计技术:程序员或测试人员对示例的走查。

静态测试 3:详细设计项完整性

测试目标:提供详细设计项的完整性信息。

测试项:详细设计项。

测试依据:更高层次设计和/或需求的可追溯信息。

测试设计技术:技术评审。

D.4 集成测试子过程

这些示例表示与开发生存周期中的集成阶段相关的测试子过程,其中(被测试的)组件正在逐步

集成。

在集成阶段，产生了许多类似的测试项，即正在集成的两个组件。本示例中的动态测试是通用的，可以对正在集成的任何两个组件执行，从最初的两个组件集成到一个组件，到最后两个组件集成到完整的系统。集成测试子过程仅在所有计划的测试完成后(或根据实际情况放弃)完成。

集成可以在不同的层次进行。它可以是将源代码组件集成到更大的组件，直至一个完整的系统，不同类型的子系统(硬件、软件、数据、培训材料等)集成到一个完整的系统，或完整的系统被集成到一个(更大的)系统组件中。虽然形式取决于风险概况，但原则是相同的。

测试子过程目标：提供关于集成组件交互的信息。

计划的测试子过程内容：静态测试：直接接口，

动态测试 1：直接接口，

动态测试 2：间接接口，

动态测试 3：共存。

静态测试：直接接口

测试目标：提供有关两个集成组件之间直接接口的信息，例如以参数列表的形式。

测试项：正在集成的组件之间的接口的源代码。

测试依据：架构设计。

详细的测试过程：测试设计和实现、测试环境的建立和维护、测试执行、测试事件报告。

测试设计技术：根据风险概况的技术评审或审查。

动态测试 1：直接接口

测试目标：提供关于两个集成组件的直接接口信息，例如以参数列表的形式。

测试项：正在集成的组件之间的接口。

测试依据：架构设计。

具体测试过程：测试设计和实现、测试环境建立和维护、测试执行、测试事件报告。

测试设计技术：适当的技术。

动态测试 2：间接接口

测试目标：提供关于两个集成组件的间接接口信息，例如通过数据库。

测试项：集成组件。

测试依据：架构设计。

具体测试过程：测试设计和实现、测试环境建立和维护、测试执行、测试事件报告。可以从早期(即组件)子过程重用测试过程，如果是这种情况，则可以最小化或省略测试设计和实现。

测试设计技术：适当的技术。

动态测试 3：共存

测试目标：提供有关集成组件(或完整系统)与环境中其他现有系统共存的信息。

测试项：集成组件(或完整系统)和环境中的现有系统。

测试依据：架构设计。

具体测试过程：测试设计和实现、测试环境建立和维护、测试执行、测试事件报告。可以重用来自其他测试的测试过程，从而减少测试设计和实现的需求。

测试设计技术：如适用，可以辅以基于经验和/或探索性的测试。

D.5 性能测试子过程

本示例展示了关注系统性能的测试子过程。

测试子过程目标：提供有关满足系统性能需求的信息。

计划的测试子过程内容：静态测试 1：性能需求文档，
静态测试 2：性能需求完整性，
静态测试 3：性能架构设计，
静态测试 4：性能详细设计，
动态测试 1：性能适用子系统，
动态测试 2：完整的系统性能。

该测试子过程未与开发生存周期中特定的阶段相关联。可以计划在开发适用的测试项时执行测试。对于静态测试，一旦计划开发测试项就可以开始准备，并且一旦测试项宣布已准备好进行静态测试，就可以进行检查和跟踪。对于动态测试，一旦测试依据稳定，就可以开始设计和实现，并且一旦测试项被宣布准备就可以执行。

本条是关于质量属性的可能测试子过程的示例。类似的测试过程可以定义为功能性、易操作性和可移植性。

静态测试 1：性能需求文档

测试目标：提供有关性能需求质量的信息。

测试项：性能需求的集合。

测试依据：内部和/或外部检查表，用于记录性能需求，例如易测试性。

测试技术：技术评审和审查(审查之前应进行非正式或技术评审，以确保要审查的需求有一定的成熟度)。

静态测试 2：性能需求完整性

测试目标：提供有关性能需求完整性的信息(所有功能需求都应该有相关的性能需求)。

测试项：所有需求。

测试依据：功能需求和性能需求之间的可验证追溯信息。

测试技术：技术评审。

静态测试 3：性能架构设计

测试目标：提供有关如何将性能需求整合到架构设计中的信息。

测试项：架构设计。

测试依据：性能需求及相关检查清单。

测试技术：走查、技术评审或审查(审查之前应进行非正式或技术评审，以确保要审查的需求有一定的成熟度)。

静态测试 4：性能详细设计

测试目标：提供有关如何将性能需求整合到详细设计的信息。

测试项：适用的一个或多个详细设计项。

测试依据：性能需求及相关检查清单。

测试技术：走查、技术评审或审查(审查之前应进行非正式或技术评审，以确保要审查的需求有一定的成熟度)。

动态测试 1：性能适用子系统

测试目标：提供关于特定子系统性能的信息。

测试项：一个或多个相关子系统。

测试依据：性能需求和可能定义的性能风险。

具体测试过程：测试设计和实现、测试环境建立和维护、测试执行、测试事件报告。可以重用一些集成测试的测试程序。

测试技术：适用的技术。

动态测试 2:完整的系统性能

测试目标:提供完整的集成系统的性能信息。

测试项:完整的系统。

测试依据:性能需求和可能定义的性能风险。

具体测试过程:测试设计和实现、测试环境建立和维护、测试执行、测试事件报告。系统测试的一些测试程序可以重用。

测试技术:适用的技术。

该测试通常在系统测试子过程中执行。

D.6 回归测试子过程

该示例表示在与测试项相关的项目中实现的更改和/或对运行测试项的环境的更改之后要执行的通用测试子过程。测试子过程将用于先前已通过一个或多个测试的测试项,并且评价为不受所实施更改的影响。

测试子过程可以作为任何其他测试子过程和任何测试项的一部分来执行。测试项的选择取决于变更的性质和风险概况。测试子过程所需的回归测试的数量取决于测试项的初始质量和测试完成准则。

测试子过程目标:在实现变更(是否与测试项相关)后,提供关于测试项状态的信息。

计划测试子过程的内容:适当地重新执行先前的静态检查或者执行动态测试。

回归测试:

测试目标:提供关于未更改测试项上已更改测试项质量的信息。

测试项:有问题的测试项。

具体测试过程:测试环境建立和维护、测试执行、测试事件报告。不需要测试设计和实现,因为该测试是使用已经通过的选定测试程序执行的。

测试设计技术:适用的技术。

静态测试 1:需求文档

测试目标:提供关于需求记录方式的信息。

测试项:所选的需求(全部或一组)。

测试依据:内部和/或外部检查列表,例如关于特定需求文档样式,和/或关于相关信息,例如唯一标识、优先级和发起者。

测试设计技术:技术评审或审查(审查之前应进行非正式或技术评审,以确保要审查的需求有一定的成熟度)。

静态测试 2:需求的可用性

测试目标:提供有关设计或测试需求的可用性信息。

测试项:所选的需求。

测试依据:不适用。

测试设计技术:与开发人员或测试人员走查。

静态测试 3:需求完整性

测试目标:提供关于一组需求完整性的信息。

测试项:所选的需求(全部或一组)。

测试依据:检查表和/或更高层次需求的可追溯信息。

测试设计技术:技术评审。

D.7 复测子过程

该示例表示以前测试执行中发现的缺陷所导致项目实现变更之后要执行的通用测试子过程，例如，对于以前测试失败的测试项目。

测试子过程可以作为任一其他测试子过程的一部分和任何测试项执行。

测试子过程目标：提供关于上报为已排除的缺陷信息。

计划测试子过程内容：重新执行以前失败的静态测试或执行动态测试。

复测：

测试目标：提供有关已更改测试项质量的信息。

测试项：有问题的测试项。

测试依据：如适用。

详细的测试过程：测试设计和实现、测试环境的建立和维护、测试执行、测试事件报告。不需要测试设计和实现，因为该测试是使用先前失败的测试程序执行的。

测试设计技术：如适用。

D.8 故事集测试子过程

该示例表示一个测试子过程，该子过程与根据当前项目待办事项列表在特定 sprint 中处理的待办事项列表的选择相关。

测试子过程目标：提供关于在特定的 sprint 中工作的故事集质量的信息。

计划的测试子过程内容：静态测试：故事的可行性。

静态测试：故事的可行性

测试目标：提供关于选择的故事集的信息。

测试项：选择的故事。

测试依据：检查列表，例如对故事的理解和估计的开发时间，以及故事之间的依赖性和一致性。

测试设计技术：非正式的评审或技术评审。

D.9 故事测试子过程

该示例表示一个测试子过程，该子过程与故事实现的完成相关，然后将故事包含在日常构建中(或者按照确定的生成新构建的速度)。

在开发冲刺待办事项列表中的故事时，会产生许多类似测试项，即故事的实现。因此，整个故事测试子过程可以覆盖一个或多个故事的测试，这取决于在构建之前实现了多少个故事。实施的故事以类似的方式单独测试。本例中的动态测试是通用的，可以对任何故事进行测试。故事测试的子过程只有在所有计划的测试完成(或根据情况放弃)时才完成。

测试子过程目标：在构建中包含实现的故事之前，提供关于它的信息。

计划测试子过程内容：静态测试：源代码质量属性。

动态测试：故事测试、探索性测试。

静态测试：源代码质量属性

测试目标：提供关于源代码质量的信息。

测试项：由故事实现创建或影响的源代码。

测试依据：内部和/或外部检查列表，例如关于特定代码编写风格，和/或编码异常，例如在声明变量

之前使用变量。

测试设计技术：技术评审或静态分析。

动态测试：故事测试

测试目标：提供关于故事实现质量的信息。

测试项：与公共构建隔绝的环境中实现的故事。

测试依据：故事。

具体测试过程：测试设计和实现、测试环境建立和维护、测试执行、测试事件报告。

测试设计技术：适用的技术，补充适当的技术以实现所需的覆盖范围。

D.10 系统测试子过程

该示例表示与系统阶段的完成相关的测试子过程。它可能与定义为成熟阶段的阶段相关，然后才能宣布系统准备好进行验收测试。

测试子过程目标：提供关于整个系统质量的信息。

计划的测试子过程内容：动态测试、系统测试。

动态测试：系统测试

测试目标：评价集成后整个系统的质量。

测试依据：系统需求。

具体测试过程：测试计划和实现、测试环境建立和维护、测试执行、测试事件报告。

测试设计技术：等价类划分、边界值分析、状态转移方法和分类树方法。

系统测试的目的是与软件系统需求上定义的功能相比，发现实际系统的缺陷。

应该预测到的是，为达到系统测试的完成准则，需要一些复测子过程和回归测试子过程。

D.11 组件测试子过程

本示例表示与开发生存周期的编码阶段相关的测试子过程。

在编码阶段，产生了许多类似的测试项，即源代码组件，它们可以直接解释组件，也可以编译并链接到可执行组件中。因此，整个组件测试子过程可以用类似的方法分别覆盖所有或部分组件的测试。本示例中的动态测试是通用的，可以应用到任何组件。组件测试子过程只有在所有计划的测试完成(或根据实际情况放弃)时才完成。

测试子过程目标：提供有关组件质量的信息。

计划的测试子过程的内容：动态测试、组件测试。

动态测试：组件测试

测试目标：提供有关组件质量的信息。

测试项：与系统中其他组件隔离的组件(可能需要驱动程序和桩)。

测试依据：组件的详细设计，包括需求的可追溯性。

具体测试过程：测试计划和实现、测试环境建立和维护、测试执行、测试事件报告。

测试设计技术：适当的技术，补充适当的技术以实现所需的覆盖范围。

附 录 E
（资料性附录）
测试角色和职责

E.1 测试角色

测试行业内的不同角色有不同的名称，因此本标准不提供一个完整的列表，列出旨在代表测试职业的不同的角色和责任。以下角色是这样描述的：担任该角色的人员可能将负责完成本标准中描述的测试过程的某个方面。每个角色可能不止一个人。

测试策略者

建立并确保符合组织级测试过程。

测试经理

开发、管理并确保符合测试管理过程。测试经理还策划和控制动态测试过程。

测试人员

开发测试可交付成果，并完成与动态测试过程相关的过程。

实际工作中，本标准中所列出的过程很可能由一系列职位不同的人完成。

E.2 测试交流

测试人员需要与组织中适当级别的人员以及与外部组织的利益相关方（如：测试项目的开发人员、产品赞助商、支持团队以及销售和营销人员）进行沟通。测试状态需要根据项目进度及时沟通。沟通可以基于书面文档，例如组织级测试策略、测试计划、测试状态报告和测试完成报告。书面文档可以附有口头陈述，就像在更正式的顺序或演化开发中出现的情况一样。在更加非正式的开发机制中，例如敏捷开发，沟通可能主要以口头方式进行，并根据需要提供书面文档。

E.3 独立测试

测试应尽可能客观。测试分析人员与测试项目的人员越接近，就越难以做到客观。一般认为，开发者发现自己工作中的缺陷比独立测试人员找到相同缺陷更困难。产品独立性评价在许多行业中都很常见，例如出版业，由编辑进行评价；有质量控制的制造业，由建筑检查员检查房屋建设质量。

开发者和测试人员的独立性按照如下顺序递增：

a) 开发者测试自己的产品；
b) 测试由不是开发本产品的开发者设计和执行，如：本产品开发者同一组织单位的另一位向同一经理汇报的开发者；
c) 测试由与开发者同一组织单位的测试人员设计和执行，并向同一经理汇报；
d) 测试由独立于开发组织单位的测试人员设计和执行，但仍然是内部的；
e) 测试由外部组织（顾问）雇用的测试人员设计和执行，但与开发者在同一组织中工作；
f) 测试由外部组织中的测试人员设计和执行（第三方测试）。

独立测试目的是在项目的时间、预算、质量和风险限制范围内，在设计测试和生产测试项的人员之间获得尽可能多的独立性。组织级测试策略应确定组织中必要的测试独立性程度，并在项目测试计划和当前测试子过程计划中继承。风险较高的情况通常会导致更高的独立性。IEEE 1012:2014 中提出

了验证和确认活动(包括测试)中独立性的概念。

对于不同的测试子过程,独立性的程度通常是不同的。在动态组件测试中,经常可以看到最低程度的独立性(即没有独立性),即使在同行评审中应用相同程度的独立性(开发者进行的静态测试)通常也不被认为是可接受的。

如果在敏捷项目中进行测试,那么由开发人员和测试人员组成的团队通常意味着很难有高级别的独立性。在这种情况下,应注意确保尽可能多的独立性。

附 录 F
（资料性附录）
本部分与 ISO/IEC/IEEE 29119-1:2013 相比的结构变化情况

本部分与 ISO/IEC/IEEE 29119-1:2013 相比在结构上有较多调整，具体章条编号对照情况见表 F.1。

表 F.1 本部分与 ISO/IEC/IEEE 29119-1:2013 的章条编号对照情况

本部分章条编号	对应 ISO/IEC/IEEE 29119-1:2013 章条编号
—	3
3	4
4	5
4.1	5.1
4.1.1	—
4.1.2	5.1.1
4.1.3	5.1.2
4.1.4	5.1.3
4.2	5.2
4.2.1	—
4.2.2	5.2.1
4.3	5.3
4.3.1	—
4.3.2	5.3.1
4.3.3	5.3.2
4.3.4	5.3.3
4.3.4.1	—
4.3.4.2	5.3.3.1
4.3.4.3	5.3.3.2
4.3.4.4	5.3.3.3
4.3.4.5	5.3.3.4
4.4	5.4
4.4.1	—
4.4.2	5.4.1
4.4.3	5.4.2
4.4.4	5.4.3
4.5	5.5
4.5.1	—

表 F.1（续）

本部分章条编号	对应 ISO/IEC/IEEE 29119-1:2013 章条编号
4.5.2	5.5.1
4.5.3	5.5.2
4.5.4	5.5.3
4.5.5	5.5.4
4.5.6	5.5.5
4.5.7	5.5.6
4.5.7.1	—
4.5.7.2	5.5.6.1
4.5.7.3	5.5.6.2
4.6	5.6
4.6.1	5.6.1
4.6.2	5.6.2
4.6.3	5.6.3
4.6.4	5.6.4
4.6.5	5.6.5
4.6.6	5.6.6
4.7	5.7
4.8	5.8
附录 F	—

参 考 文 献

[1] GB/T 25000.1—2010 软件工程 软件产品质量要求与评价(SQuaRE) SQuaRE 指南

[2] GB/T 25000.10—2016 系统与软件工程 系统与软件质量要求和评价(SQuaRE) 第10部分:系统与软件质量模型

[3] GB/T 25000.51—2016 系统与软件工程 系统与软件质量要求和评价(SQuaRE) 第51部分:就绪可用软件产品(RUSP)的质量要求和测试细则

[4] GB/T 32421—2015 软件工程 软件评审与审核

[5] GB/T 32422—2015 软件工程 软件异常分类指南

[6] GB/T 32423—2015 系统与软件工程 验证与确认

[7] ISO/IEC/IEEE 12207 Systems and software engineering—Software life cycle processes

[8] ISO/IEC 15026-3 Information technology—System and software integrity levels

[9] ISO/IEC 16085 Systems and software engineering—Life cycle processes—Risk management

[10] ISO/IEC 20000 Information technology—Service management

[11] ISO/IEC/IEEE 24765 Systems and software engineering—Vocabulary

[12] IEC 60300-3-9:1995 Risk analysis of technological systems

[13] IEEE Std 610.12-1995 IEEE Standard Glossary of Software Engineering Terminology

[14] IEEE Std 829-2008 IEEE Standard for Software and System Test Documentation

[15] IEEE Std 1008-1987 IEEE Standard for Software Unit Testing

[16] BS 7925-1:1998 Software testing—Vocabulary

[17] BS 7925-2:1998 Software testing—Software component testing

[18] International Software Testing Qualifications Board (ISTQB), Standard glossary of terms used in Software Testing [online]. 2010. Updated 1 April 2010 [viewed 11 April 2011].

[19] CRISPIN, L. and GREGORY, J. 2009. Agile Testing: A Practical Guide for Testers and Agile Teams. Pearson Education.

[20] KOEN, B. V., 1985. Definition of the Engineering Method. American Society for Engineering Education.