高级测试工程师认证

测试分析师（CTAL-TA）

教学大纲

版本：3.1.0

国际软件测试认证委员会

版权声明

中文版权声明

修订历史

更多细节，请参见发布说明。

版本	日期	说明
V3.1.0	2021年3月	进行了较小的更新，包括重写了第3.2.3节并改进了措辞
2019 V3.0	2019年10月19日	重大更新，整体修订并缩小范围
2012 V2.0	2012年10月19日	作为单独的AL-TA教学大纲的第一个版本

目录

致谢

本文档由国际软件测试认证委员会高级工作组的测试分析师核心团队编制：Mette Bruhn-Pedersen（工作组主席）；Matthias Hamburg（产品所有者）；Wim Decoutere,、István Forgács,、Adam Roman、Jan Sabak、Marc-Florian Wendland（作者）。

核心团队感谢Paul Weymouth 和Richard Green进行技术编辑，Gary Mogyorodi进行术语一致性检查，各分会提供了与已发布的2019版教学大纲相关的评审意见和修改建议。

以下人员参加了对本教学大纲的评审和讨论：

Gery Ágnecz，Armin Born，Chenyifan，Klaudia Dussa-Zieger，Chen Geng（Kevin），Istvan Gercsák，Richard Green，Ole Chr。 Hansen，Zsolt Hargitai，Andreas Hetz，Tobias Horn，Joan Killeen，Attila Kovacs，Rik Marselis，Marton Matyas，Blair Mo，Gary Mogyorodi，IngvarNordström，Tal Pe'er，Palma Polyak，Nishan Portoyan，Meile Posthuma，Stuart Reid，Murian Song，Péter Sótér，Lucjan Stapp，Benjamin Timmermans，Chris van Bael，Stephanie van Dijck，Paul Weymouth。

ISTQB®于2021年2月23日发布本大纲英文版。

本文档的2019版本由国际软件测试认证委员会高级工作组的核心团队编制：Graham Bath、Judy McKay、Mike Smith。

以下专家参加了本教学大纲的评审、讨论和表决：

Laura Albert, Markus Beck, Henriett Braunné Bokor, Francisca Cano Ortiz, Guo Chaonian, Wim Decoutere, Milena Donato, Klaudia Dussa-Zieger, Melinda Eckrich-Brajer, Péter Földházi Jr, David Frei, Chen Geng, Matthias Hamburg, Zsolt Hargitai, Zhai Hongbao, Tobias Horn, Ágota Horváth, Beata Karpinska, Attila Kovács, József Kreisz, Dietrich Leimsner, Ren Liang, Claire Lohr, Ramit Manohar Kaul, Rik Marselis, Marton Matyas, Don Mills, Blair Mo, Gary Mogyorodi, Ingvar Nordström, Tal Peer, Pálma Polyák, Meile Posthuma, Lloyd Roden, Adam Roman, Abhishek Sharma, Péter Sótér, Lucjan Stapp, Andrea Szabó, Jan te Kock, Benjamin Timmermans, Chris Van Bael, Erik van Veenendaal, Jan Versmissen, Carsten Weise, Robert Werkhoven, Paul Weymouth.

本文档的2012年版本由国际软件测试认证委员会高级子工作组（高级测试分析师）的核心团队编制：Judy McKay（主席），Mike Smith，Erik van Veenendaal。

在高级教学大纲完成时，高级工作组有以下成员（按字母顺序排列）：

Graham Bath，Rex Black，Maria Clara Choucair，Debra Friedenberg，BernardHomès（副主席），Paul Jorgensen，Judy McKay，Jamie Mitchell，Thomas Mueller，Klaus Olsen，Kenji Onishi，Meile Posthuma，Eric Riou du Cosquer，Jan Sabak，Hans Schaefer，Mike Smith（主席），Geoff Thompson，Erik van Veenendaal和Yumoto Tsuyoshi。

以下专家参加了2012版教学大纲的评审、讨论和表决：

Graham Bath，Arne Becher，Rex Black，Piet de Roo，Frans Dijkman，Mats Grindal，Kobi Halperin，BernardHomès，MariaJönsson，Junfei Ma，Eli Margolin，Rik Marselis，Don Mills，Gary Mogyorodi，Stefan Mohacsi，Reto Mueller，Thomas Mueller，Ingvar Nordstrom，Tal Pe'er，Raluca Madalina Popescu，Stuart Reid，Jan Sabak，Hans Schaefer，Marco Sogliani，Yaron Tsubery，Hans Weiberg，Paul Weymouth，Chris van Bael，Jurian van der Laar，Stephanie van Dijk，Erik van Veenendaal，Wenqiang zheng，Debi Zylbermann。

0.教学大纲引言

0.1 本教学大纲编写目的

本教学大纲是国际软件测试认证-高级测试分析师的中文版教学大纲。ISTQB®提供本教学大纲用于：

1. 各个分会，翻译成本国语言并且授权给培训机构。分会可以根据他们特定的语言调整教学大纲，以及引用当地出版物。
2. 考试机构，根据教学大纲的学习目标，编制当地语言的考试题。
3. 培训机构，编制课件和决定相应的授课方法。
4. 认证考试应试者，准备认证考试（认证考试可以是作为培训课程一部分或独立准备）。
5. 国际软件和系统工程界，促进软件和系统测试的专业化，并且作为书籍和文章的基础之一。

ISTQB®允许其他组织为其他目的使用本教学大纲，只要他们事先获得ISTQB®书面许可。

0.2 软件测试的高级认证测试人员

高级认证包括3份独立的教学大纲，分别对应下面角色：

- 测试经理
- 测试分析师
- 技术测试分析师

ISTQB®高级教学大纲概述文档2019 [ISTQB_AL_OVIEW]是一份单独的文档，包括下面内容：

- 每份大纲的商业价值
- 商业价值和学习目标的追溯矩阵
- 每份大纲的概要
- 各个大纲之间的关系

0.3 可考核的学习目标和知识认知级别

学习目标支撑商业价值，并且用于编制高级测试分析师考题。

本教学大纲中每章开头都会给出相应的学习目标和认知级别，要求如下：

- K2：理解
- K3：应用
- K4：分析

章节标题下面列出的所有术语的定义作为关键词需要牢记（K1），即使在学习目标中没有明显提及。

0.4 高级测试分析师考试

高级测试分析师考试的内容将基于本教学大纲的内容。考题的答案，可能会涉及本教学大纲的一个甚至多个章节知识点。考试的范围覆盖本教学大纲除简介和附录外的所有章节。标准、书籍和其他ISTQB®教学大纲可供参考，其内容不直接作为考试内容，但本教学大纲中从这些参考资料总结出的内容是考试范围。

考试形式是多选题，共有40题。必须至少获得总分的65%才算通过考试。

考试可以作为认证培训课程的一部分，也可以单独参加考试（例如在考试中心或公开考试）。完成认证培训课程不作为考试的先决条件。

0.5 参加考试的要求

参加高级测试分析师考试前，必须取得基础级测试工程师认证证书。

0.6 经验要求

高级测试分析师的学习目标不要求任何特定经验。

0.7 课程的授权

ISTQB®分会可以授权那些遵照本教学大纲编写课程材料的培训机构。培训机构应从分会或开展授权的机构获取授权指南。被授权课程需要遵循本教学大纲，并且允许将ISTQB®考试作为课程的一部分。

0.8 本教学大纲的详细程度

本教学大纲的详细程度是考虑了在全球范围内采取一致的教学和考核。为了达到这个目标，本教学大纲由下面几部分组成：

- 总体教学目标，描述了高级测试分析师的目标
- 列出了考生需要能够记忆的术语
- 各个知识域的学习目标，描述要获得的认知学习成果
- 关键概念描述，包括参考来源，例如已认可的文献或标准。

本教学大纲并没有包含整个知识领域，只是提供了高级培训课程需要覆盖的详细程度。本教学大纲关注的是能应用到所有软件项目（包括敏捷软件开发）的资料。本教学大纲不包含与任何特定软件开发生命周期相关的学习目标，但会讨论这些概念如何应用到敏捷软件开发、其他迭代和增量生命周期模型，以及顺序生命周期模型。

0.9 本教学大纲的结构

本教学大纲由包含考试内容的六大章节组成。每一章的一级标题给出本章的最低教学和练习时间，一级标题以下不再列出时间限制。对于经过认可的培训课程，本教学大纲要求至少20.5小时的授课时间。下面是六大章节的课时分布：

- 第一章：（150分钟）测试过程中测试分析师的任务
- 第二章：（60分钟） 基于风险的测试中测试分析师的任务
- 第三章：（630分钟）测试技术
- 第四章：（180分钟）测试软件质量特性
- 第五章：（120分钟）评审
- 第六章：（90分钟） 测试工具和自动化

1.测试过程中测试分析师的任务–150分钟

关键词

出口准则（exit criteria）、概要测试用例（high-level test case）、详细测试用例（low-level test case）、测试（test）、测试分析（test analysis）、测试依据（test basis）、测试条件（test condition）、测试数据（test data）、测试设计（test design）、测试执行（test execution）、测试执行进度表（test execution schedule）、测试实施（test implementation）、测试规程（test procedure）、测试套件（test suite）。

测试过程中测试分析师的任务的学习目标

1.1 引言

无学习目标

1.2 软件开发生命周期中的测试

TA-1.2.1 (K2) 解释当使用不同的软件开发生命周期模型时，测试分析师的介入时间和介入程度有何不同，为什么会有不同。

1.3 测试分析

TA-1.3.1 (K2) 总结测试分析活动中测试分析师的任务。

1.4 测试设计

TA-1.4.1 (K2) 解释为什么利益相关方需要理解测试条件。

TA-1.4.2 (K4) 根据给定项目场景，为测试用例选择合适的设计级别（概要测试用例或详细测试用例）。

TA-1.4.3 (K2) 解释测试用例设计中需要考虑的问题。

1.5 测试实施

TA-1.5.1 (K2) 总结测试实施活动中测试分析师的任务。

1.6 测试执行

TA-1.6.1 (K2) 总结测试执行活动中测试分析师的任务。

1.1 引言

在ISTQB®基础级教学大纲中，测试过程包括了下面活动：

- 测试计划
- 测试监督与控制
- 测试分析
- 测试设计
- 测试实施
- 测试执行
- 测试结束

在本高级教学大纲中，会进一步说明与测试分析师有特定相关性的活动。从而细化测试过程，更好地适应不同的软件开发生命周期模型。

测试分析师的主要关注领域包括确定、设计、实施和执行适当的测试。理解测试过程中的其他步骤也很重要，但测试分析师的主要工作通常聚焦于下面活动：

- 测试分析
- 测试设计
- 测试实施
- 测试执行

测试过程中的其他活动已经在基础级中充分讨论，本教学大纲中不再做进一步展开。

1.2 软件开发生命周期中的测试

在制定测试策略时，应考虑整个软件开发生命周期。对于不同的生命周期，测试分析师的介入时间点是不同的；参与的程度、所需的时间、可用的信息和期望也可能有很大的不同。测试分析师必须要知道提供给其他相关组织角色的信息类型，例如：

- 需求工程和管理 –需求评审反馈
- 项目管理 –进度输入
- 配置和变更管理 –每个构建的验证测试结果、版本控制信息
- 软件开发 –通报发现的缺陷
- 软件维护 –有关缺陷、缺陷移除效率和确认测试的报告
- 技术支持 –准确记录已知问题和应急方法的文档
- 技术文档的编制（如数据库设计规格说明、测试环境文档） –这些技术文档的输入和技术评审

测试活动必须与选定的软件开发生命周期相匹配。生命周期的本质可以是顺序、迭代、增量或者几者的混合。例如在顺序V模型中，系统测试级别的测试过程可以这样规划：

- 系统测试计划与项目计划同时进行，测试监督与控制贯穿测试全过程。这会影响测试分析师为项目管理的目标达成而提供的进度输入信息。
- 系统测试分析和设计，与系统需求规格说明、系统架构（概要）设计规格说明、组件（详细）设计规格说明等文档保持一致。
- 系统测试环境的实现可能在系统设计期间就开始，尽管大部分工作通常和编码与组件测试同时进行，系统测试实现活动的工作通常会持续到系统测试执行开始几天前才完成。
- 当满足入口准则或，如有必要，豁免入口准则时（这通常意味着组件测试和组件集成测试至少满足了相应的出口准则），开始执行系统测试。系统测试持续执行到满足相应的系统测试出口准则为止。
- 系统测试结束活动在系统测试出口准则满足后开始。

迭代和增量模型可能不会遵循相同的活动顺序，也可能不包括某些活动。例如，迭代模型可能在每轮迭代中使用一个裁剪过的测试活动集合。每轮迭代可能都会有测试分析、设计、实施、执行活动，但概要计划只在项目开始时执行，结束活动只在最后执行。

在敏捷软件开发中，通常使用不太正式的过程，并与项目干系人建立更紧密的工作关系，从而使变更更容易在项目中发生。该过程可能没有明确、清晰地定义测试分析师角色，也没有比较全面完整的测试文档，沟通也更短暂、更频繁。

敏捷软件开发从一开始就涉及测试工作。在产品开发工作启动时即开发人员开始进行架构和设计工作时，测试就要开始介入。评审可以是非正式的，随着软件的发展持续进行。测试活动需要贯穿整个项目生命周期，测试分析师的工作可由团队协作完成。

迭代和增量模型的范围从敏捷软件开发（在敏捷软件开发中，会随着客户需求的演变发生变化）到混合模型，例如迭代/增量开发模型与V模型相结合。在这种混合模型中，测试分析师应该参与顺序活动中的计划和设计工作，然后在迭代/增量活动中扮演更具交互性的角色。

无论使用什么样的软件开发生命周期，测试分析师都必须了解参与目的和参与时间。通过调整其参与特定软件开发生命周期模型的活动和参与时间，测试分析师可以为软件质量做出有效贡献，而不必拘泥于预先定义的角色模型。

1.3 测试分析

在测试计划阶段，定义了测试项目的范围。在测试分析阶段，测试分析师使用此范围定义来：

- 分析测试依据
- 识别测试依据中的不同缺陷类型
- 识别需要测试的测试条件和特征（功能），并设置优先级
- 获取测试依据和相应测试条件之间的双向可追溯性
- 执行与基于风险的测试相关的任务（见第2章）

为了让测试分析师有效地进行测试分析，需要满足以下入口准则：

- 有一个描述测试对象的知识体系（例如需求、用户故事），可以作为测试依据（参见[ISTQB FL基础级教学大纲] 1.4.2和2.2或其他测试依据来源列表）
- 该测试依据已通过评审，结果合理，并在评审后根据需要进行了更新。注意，如果要定义概要测试用例（见第1.4.1节），测试依据可以不用充分定义。在敏捷软件开发中，会在每次迭代开始时细化用户故事，因此这个评审周期将是迭代的。
- 有已经批准的预算和时间表用于完成测试对象中的剩余测试任务。

测试条件通常通过分析测试依据和测试目标（如测试计划中的定义）来确定。在某些情况下，文档可能是旧的或不存在时，测试条件可以通过与利益相关方的讨论来确定（例如，在工作会议或迭代计划期间）。在敏捷软件开发中，测试设计的依据可以来自用户故事中定义的部分验收标准。

虽然测试条件通常依赖于要测试的条目，但是测试分析师需关注一些通用的注意事项：

- 通常建议定义不同详细程度的测试条件。最初，确定概要条件以定义测试的通用目标，例如“screen x的功能”。随后，识别更详细的条件作为具体测试用例的基础，例如“screen x拒绝比正确长度短一位数的帐号”。使用分层的方法来定义测试条件，有助于保证概要条目覆盖的充分性。这种方法还允许测试分析师开始为尚未细化的用户故事定义概要测试条件。
- 如果已经定义了产品风险，那么测试条件必须保证识别每个产品风险并追溯到风险条目。

测试技术的应用（如测试策略和/或测试计划中所定义的）有助于测试分析活动，并可用于支持以下目标：

- 识别测试条件
- 减少遗漏重要测试条件的可能性
- 定义更精确的测试条件
- 在测试条件细化和完善后，可以与利益相关方一起进行评审，以保证充分理解需求，以及测试与项目目标保持一致。

在某一特定领域（例如特定功能）的测试分析活动结束时，测试分析师应该知道必须为该领域设计哪些特定测试。

1.4 测试设计

随着测试过程的进行，测试分析师按照测试计划确定的范围，设计将要实施和执行的测试。测试设计包括以下活动：

- 确定哪些测试区域适合详细测试用例或概要测试用例
- 确定可以实现必要覆盖率的测试技术。可用的技术需在测试计划期间确定
- 使用测试技术设计测试用例和用例集合，覆盖已识别的测试条件
- 确定必要的测试数据以支持测试条件和测试用例
- 设计测试环境，确定必要的基础设施，包括工具
- 获取双向可追溯性（例如在测试依据、测试条件和测试用例之间）

从分析和设计到实施和执行的整个过程，均应使用在风险分析和测试计划中识别的优先级准则。

根据所设计的测试类型，测试设计的入口准则之一可以是在设计工作期间所用工具的可用性。

在测试设计阶段，测试分析师至少要考虑下列几点：

- 有些测试条目，最好只定义测试条件，而不是定义出脚本化的测试执行序列。在这种情况下，应将测试条件定义为无脚本化的测试指南。
- 应该明确识别通过/失败准则。
- 测试的设计应使其它测试人员（而不仅仅是作者）可以理解。如果作者不是执行测试的人，则其他测试人员需要阅读并理解之前定义的测试，以便理解测试目标和测试的相对重要性。
- 其他利益相关方也必须理解测试，如开发人员（他们可能会评审测试），审计员（他们可能会需要批准测试）。
- 测试应涵盖与测试对象交互的所有类型，不应仅局限于通过用户可见界面进行的交互。例如，它们还可以包括与其他系统以及技术或物理事件的交互（详见[IREB_CPRE]）。
- 测试应设计为测试各种测试对象之间的接口，以及对象本身的行为。
- 测试设计需要优化和平衡风险级别与商业价值的一致性。

1.4.1 详细/概要测试用例

测试分析师的主要工作之一就是针对给定的场景确定最佳的测试用例粗略程度。详细和概要测试用例在[基础级教学大纲]中有介绍。使用这些方法的优缺点如下：

使用详细测试用例有如下优点：

- 缺乏经验的测试人员可以依赖项目内提供的详细信息。详细测试用例提供测试人员执行测试用例（包括任何数据需求）和验证实际结果所需的所有特定信息和过程。
- 不同测试人员可重复执行测试，并且应该得到相同结果。
- 可以发现测试依据中不明显的缺陷。
- 可根据需要对测试进行独立验证，比如审计时。

- 可以减少在自动化测试用例实现上花费的时间。

使用详细测试用例有如下的缺点：

- 创建和维护需要大量的投入。
- 限制了测试人员在执行过程中的创造性。
- 要求明确定义测试依据。
- 相比概要测试用例，建立与测试条件的可追溯性会花费更多精力。

使用概要测试用例有如下优点：

- 为需要测试的内容提供指导，并允许测试分析师在执行测试时更改实际数据，甚至更改所遵循的规程。
- 可能比详细测试用例提供更好的风险覆盖率，因为每次执行时都会有所不同。
- 可以在需求阶段的早期定义。
- 在执行测试时，可以利用测试分析师在测试和测试对象方面的经验。
- 在没有详细和正式文档的情况下也可以定义。
- 当每轮使用不同测试数据时，它们更适合在不同的测试周期中复用。

使用概要测试用例有如下缺点：

- 可重现性（重现相同结果）较差，使得验证更加困难。因为缺少了详细测试用例中的细节描述。
- 需要更有经验的人员来执行。
- 以概要测试用例为基础进行自动化时，由于缺少细节可能会导致验证了错误的实际结果，或者漏测了应该确认的内容。

当需求变得更加明确和稳定时，可以使用概要测试用例来开发详细测试用例。这种情况下，测试用例的创建是按顺序完成的，从概要到详细，只有详细测试用例用于执行。

1.4.2 设计测试用例

测试用例是通过使用测试技术逐步细化已识别测试条件来设计的（见第3章）。测试用例应该是可重复的、可验证的，并且可以追溯到测试依据（例如需求）。

测试设计需要识别下面内容：

- 目标（即测试执行中可观测、可度量的目标）
- 前置条件，例如项目或本地测试环境需求及其交付计划，或测试执行前系统的状态等
- 测试数据需求（包括测试用例的输入数据，以及为了执行测试用例，系统中必须存在的数据）
- 具有明确的通过/失败准则的预期结果
- 后置条件，如受到影响的数据，测试执行后系统的状态，后续处理的触发等

定义测试预期结果是个特别的挑战。手动计算通常很乏味且容易出错；如果可能的话，最好是找到或创建一个自动化的测试结果参照物。在识别预期结果时，测试人员不仅要关注屏幕上的输出，还要关注数据和环境后置条件。如果明确定义了测试依据，则在理论上确定正确的结果应该很简单。然而，测试依据文档可能是模糊的、矛盾的、缺少对关键领域的覆盖，或者完全缺失。在这种情况下，测试分析师必须具有专业的（或者能够获得）技能。此外，即使测试依据是明确定义的，各种影响因素和系统相互之间复杂的交互也会使预期结果的定义变得困难；因此，测试结果参照物是必不可少的。在敏捷软件开发中，测试结果参照物可能来自产品负责人。假如没有任何方法可以确定测试实际结果的正确性，则其执行可能变得毫无意义，通常会生成无效的测试报告或对系统的错误信心。

上述活动可适用于所有测试级别，尽管测试依据会有所不同。在分析和设计测试时，记住测试级别和相应的测试目的是很重要的。这有助于确定所需的详细程度以及可能需要的工具（例如，组件测试级别的驱动器和桩）。

在开发测试条件和测试用例的过程中，通常会创建一些文档，从而生成测试工作产品。在实践中，测试工作产品的文档记录程度差异很大。这可能会受到以下因素的影响：

- 项目风险（什么必须/不能被记录到文档）
- 文档带给项目的附加价值
- 需要遵循的标准或法规
- 使用的软件开发生命周期或方法（例如敏捷方法只要“刚好够用”的文档）
- 测试分析和设计过程中，对测试依据的可追溯性的要求

根据测试范围，测试分析和设计会关注测试对象相关的质量特性。ISO25010标准[ISO25010]提供了有用的参考。在测试硬件/软件系统时，可能会需要额外的质量特性。

测试分析和测试设计活动可以通过将其与评审和静态分析相结合而得到加强。事实上，进行测试分析和测试设计通常是静态测试的一种形式，因为通过这个活动可能会在测试依据文档中发现问题。基于需求规格说明书进行测试分析和测试设计是准备需求评审会议的一种很好的方法。阅读需求并用其创建测试，需要理解需求并确定用来评价需求是否实现的一种方法。此活动通常会发现漏掉的需求和不清楚、不可测试或未定义验收准则的需求。类似地，测试工作产品（如测试用例、风险分析和测试计划）也可以接受评审。

在测试设计过程中，可以定义所需的详细测试基础设施需求，尽管实际上测试实施之前可能都不会确定这些需求。必须记住，测试基础设施不仅仅包括测试对象和测试件。例如，基础设施需求可以包括房间、设备、人员、软件、工具、外围设备、通信设备、用户授权以及运行测试所需的所有其他项。

测试分析和测试设计的出口准则将根据项目参数而有所不同，但这两个章节中讨论的所有条目均应考虑纳入到定义的出口准则中。重要的是，出口准则是可度量的，并且提供了后继步骤所需的所有信息，并且已执行了所有必需的准备工作。

1.5 测试实施

测试实施基于测试分析和设计来准备测试执行所需的测试件。它包括以下活动：

- 开发测试规程，可能的话，创建自动测试脚本
- 将测试规程和自动测试脚本（如果有）纳入测试套件，以在特定测试运行中执行
- 咨询测试经理，确定执行测试用例和测试套件的优先级
- 创建包括资源分配的测试执行进度表，以便可以开始执行测试（参见基础级教学大纲5.2.4）
- 完成测试数据和测试环境的准备
- 更新测试依据和测试件之间的可追溯性，例如测试条件、测试用例、测试规程、测试脚本和测试套件

在测试实施期间，测试分析师会确定测试用例的高效执行顺序并依此创建测试规程。定义测试规程需要仔细识别可能影响测试执行顺序的约束和依赖关系。测试规程记录了所有初始前置条件（例如，从数据存储库中加载测试数据）以及后置的任何活动（例如，重置系统状态）。

测试分析师找到可以进行分类组合的测试规程和自动测试脚本（例如，它们都与特定业务流程的测试相关），将它们组织并纳入到测试套件中。这使得相关的测试用例能够一起执行。

测试分析师以高效的测试执行为目标，在测试执行计划中对测试套件进行排序。如果使用基于风险的测试策略，则风险级别将是确定测试用例执行顺序的主要考虑因素。可能还有其他因素决定测试用例的执行顺序，例如合适的人员、设备、数据和被测试功能的可用性。

将代码分段发布并不罕见，但测试工作量必须与可发布软件的测试顺序相协调。特别是在迭代和增量开发模型中，测试分析师必须与开发团队协调，确保软件以可测试的顺序发布并进行测试是非常重要的。

测试条件和测试用例的详细程度会影响测试实施期间所做工作的详细程度和复杂性。在某些情况下会涉及到法律法规，测试工作产品应提供符合相关标准的证据，如美国标准DO-178C（在欧洲，ED 12C）。[RTCA DO-178C/ED-12C]。

如上所述，大多数测试都需要测试数据，在某些情况下，这些数据集可能非常大。在实施过程中，测试分析师会创建输入和环境数据以加载到数据库和其他类似的存储库。这些数据必须“匹配目标”才能够发现缺陷。测试分析师还可以创建用于数据驱动和关键字驱动的测试数据（见6.2节）以及手工测试的数据。

测试实施也与测试环境有关。在测试实施阶段，应在测试执行之前完成设置并验证好环境。“匹配目标”的测试环境是必不可少的，即测试环境能够有利于通过可控的测试发现相应的缺陷；在没有失效发生时能正常运行，并在需要时可以精确重现生产或最终用户环境以进行更高级别的测试。如遇到非预期的变更、测试结果或者其它因素，则需在测试执行期间变更测试环境。如果在测试执行期间确实发生了环境变更，则评估变更对已运行的测试的影响将非常重要。

在测试实施过程中，测试分析师必须确认负责创建和维护测试环境的人员是否已知并可用，同时保证所有的测试件、测试支持工具和相关的流程就绪。这包括配置管理、缺陷管理、测试日志记录和管理。此外，对用于根据出口准则和测试结果报告来评价当前状态的数据，测试分析师必须对数据收集流程进行验证。

制定测试计划时，使用平衡的策略做测试实施是明智的。例如，基于风险的分析型测试策略通常与应对型测试策略相结合。在这种情况下，部分测试实施的工作可以使用不需遵循预定脚本（非脚本化）的测试。

非脚本的测试不应是随机的或无目的的，因为这可能无法预测持续时间和覆盖范围，且导致缺陷发现率低。相反，它应该在规定时间的会话中进行，每次都有个测试章程作为初步指引，但如果在会话期间发现可能更有成效的测试机会，则可以自由偏离章程的约定。多年来，测试人员开发了多种基于经验的测试技术，例如缺陷攻击[Whittaker03]、错误猜测[Myers11]和探索性测试[Whittaker09]。测试分析、测试设计和测试实施仍然存在，但它们主要发生在测试执行过程中。

当遵循这种应对型测试策略时，每个测试的结果都会影响后续测试的分析、设计和实施。虽然这些策略是轻量级的，并且通常能够有效地发现缺陷，但也有一些缺点，包括：

- 测试分析师需要具有专业能力
- 测试时长难以预估
- 覆盖范围难以跟踪
- 如果没有良好的文档化或工具支持，则可重复性可能较差

1.6 测试执行

测试执行需要根据测试执行计划进行，包括以下任务（参见ISTQB基础级教学大纲）：

- 执行手工测试，包括探索性测试；
- 执行自动化测试；
- 比较测试结果和预期结果；
- 分析异常，找出可能的原因；
- 基于观察到的失效报告缺陷；
- 记录测试执行的实际结果；
- 根据测试结果，更新测试依据和测试件之间的可追溯性；
- 执行回归测试；

上述测试执行任务可以由测试人员或者测试分析师执行。

以下是测试分析师可能执行的典型附加任务：

- 识别缺陷集群，通常这表示软件的特定部分需要更多测试；

- 根据探索性测试的结果，为以后的探索性测试会话提出建议；
- 根据执行测试任务时获取的信息，识别新的风险点；
- 对改进测试实施活动中的任何工作产品提出建议（例如，改进测试规程）。

2.测试分析师在基于风险的测试中的任务–60分钟

关键字

产品风险（product risk）、风险识别（risk identification）、风险缓解（risk mitigation）、基于风险的测试（risk-based testing）。

测试分析师在基于风险的测试中的任务和学习目标

测试分析师在基于风险的测试中的任务

TA -2.1.1 (K3) 针对给定情况，参与风险识别，进行风险评估，并提出适当的风险缓解建议。

2.1 简介

测试经理通常全面负责建立和管理基于风险的测试策略。他们通常会要求测试分析师投入其中，以确保基于风险的方法得到正确实施。

测试分析师应积极参与以下基于风险的测试任务：

- 风险识别
- 风险评估
- 风险缓解

这些任务在软件开发生命周期内反复执行，以应对新出现的风险、不断变化的优先级，并定期评估和沟通风险状态(详见[vanVeenendaal12]和[Black02])。在敏捷软件开发中，这三个任务通常融合在所谓的风险会话中，关注焦点放在迭代或发布上。

测试分析师应该在测试经理为项目建立的基于风险的测试框架内工作。他们应提供项目中固有的与业务领域风险相关的知识，例如与功能安全、商业和经济问题、政治因素等相关的风险。

2.2 风险识别

在风险识别过程中通过广泛地访谈具有代表性的利益相关方，将有最大可能识别出尽可能多的、有重大意义的风险。

测试分析师通常拥有与被测系统的特定业务领域相关的独特知识。这意味着他们特别适合执行下列任务：

- 与领域专家和用户进行专业的访谈
- 进行独立评估
- 使用风险模板
- 参与风险研讨会
- 参与与潜在用户和现有用户的头脑风暴会议
- 定义测试检查表
- 借鉴过去类似系统或项目的经验

特别要强调的是，测试分析师应该与用户和其他领域专家(例如，需求工程师，业务分析师)密切合作，以确定在测试期间应该处理的业务风险领域。在敏捷软件开发中，与利益相关方的密切联系使得风险识别能够定期进行，比如在迭代计划会议期间。

项目中可能识别的风险示例包括：

- 功能正确性问题，如计算错误
- 易用性问题，例如键盘快捷键不足
- 可移植性问题，例如，无法在特定平台上安装应用程序

2.3 风险评估

风险识别是指尽可能多地识别相关风险，而风险评估则是对这些已识别风险的研究。具体地说，即对每个风险进行归类，并确定风险级别。

确定风险级别通常涉及对每个风险项评估风险可能性和风险影响。风险可能性通常被解释为被测系统投产后，存在潜在问题且问题可被发现的可能性。技术测试分析师应有助于发现和理解每个风险项的潜在可能性，而测试分析师应有助于理解问题发生时的潜在业务影响(在敏捷软件开发中，这种基于角色的区别可能不那么明显)。

风险影响通常被解释为对用户、客户或其他利益相关方影响的严重性。换言之，它源于商业风险。测试分析师应有助于识别和评估每个风险项对潜在业务领域或用户的影响。影响商业风险的因素包括：

- 受影响功能的使用频率
- 商业损失
- 经济损失
- 生态或社会损失或责任
- 民事或刑事法律制裁
- 功能安全问题
- 罚款，吊销执照
- 如造成人们无法继续工作，且没有合理的变通方案
- 功能的可见性
- 可见的失效导致负面报道和潜在的形象损害
- 客户流失

给定可用的风险信息，测试分析师需要根据测试经理提供的指导方针来建立业务风险级别。可以使用排序列表（使用数字或低/中/高）或交通信号灯颜色进行分类。确定风险可能性和风险影响后，测试经理使用这些值来确定每个风险项的风险级别。 然后，使用该风险级别来确定风险缓解活动的优先级。

2.4 风险缓解

在项目过程中，测试分析师应设法做到以下几点：

- 通过设计有效的测试用例证明测试是通过还是失败，并通过参与软件工作产品(如需求、设计和用户文档)的评审来降低产品风险
- 实施测试策略和测试计划中确定的适当的风险缓解活动(例如，使用特定的测试技术测试高风险的业务流程)
- 根据项目展开时收集的额外信息重新评估已知风险，适当调整风险可能性、风险影响或两者兼有
- 在运行测试获得的信息中识别新的风险

当人们在谈论一项产品风险时，测试对降低此类风险做出了重要贡献。通过发现缺陷，测试人员通过提供缺陷的信息和在发布版本前处理缺陷的机会来降低风险。如果测试人员没有发现缺陷，那么测试通过提供证据证明在特定的条件下(例如，已测试的条件下)，系统可正常运行，从而降低风险。测试分析师通过调查来收集准确的测试数据、创建和测试真实的用户场景、执行或监督易用性研究等方法来帮助确定风险缓解方案。

2.4.1 确定测试优先级

风险级别可用于确定测试的优先级。测试分析师可能需查明在一个统计系统中业务精确性部分是否存在高风险。因此，为了降低风险，测试人员可能会与其他业务领域专家合作，一起收集足以处理和验证数据精确性的代表数据。类似地，针对新测试对象，测试分析师可能会确定易用性问题是一个重大风险。与其等待用户验收测试发现问题，测试分析师则可以优先进行以原型为基础的早期易用性测试，以便在用户验收测试之前识别并解决易用性设计问题。这种优先级必须在计划阶段尽早考虑，以便安排在时间工作进度表计划内，从而能够在必要的时间安排必要的测试。

在某些情况下，风险最高的测试先运行，低风险的测试后运行，测试严格按照风险顺序运行(称为“深度优先”)；在某些情况下，通过在所有已识别的风险域中抽取测试样本，按照风险级别加权重的方式选择测试，同时确保每个风险至少覆盖一次(称为“广度优先”)。

无论基于风险的测试是深度优先还是广度优先，都有可能没有足够的时间执行完所有的测试任务。基于风险的测试允许测试人员在此时向管理层报告剩余的风险级别，并允许管理层决定是否延长测试或将剩余的风险转移到用户、客户、帮助台（help desk）/技术支持和/或操作人员身上。

2.4.2为未来的测试周期调整测试

风险评估不是在实施测试前进行的一次性活动，而是一个持续的过程。每个未来的测试周期规划都应该进行新的风险分析，并考虑以下因素：

- 任何新的或发生显著变化的产品风险
- 在测试中发现不稳定或容易出错的区域
- 来自已经修复缺陷的风险
- 测试中发现的典型缺陷
- 测试不足的区域（对需求覆盖率低）

3.测试技术–630分钟

关键字

黑盒测试技术（black-box test technique）、边界值分析（boundary value analysis）、基于检查表的测试（checklist-based testing）、分类树技术（classification tree technique）、判定表测试（decision table testing）、缺陷分类（defect taxonomy）、基于缺陷的测试技术（defect-based test technique）、等价类划分（equivalence partitioning）、错误猜测（error guessing）、基于经验的测试（experience-based testing）、基于经验的测试技术（experience-based test technique）、探索性测试（exploratory testing）、结对测试（pairwise testing）、状态转换测试（state transition testing）、测试章程（test charter）、用例测试（use case testing）。

测试技术的学习目标

3.1 简介

无学习目标

3.2 黑盒测试技术

TA-3.2.1 (K4) 通过应用等价类划分来分析给定的规格说明项并设计测试用例。

TA-3.2.2 (K4) 通过应用边界值分析来分析给定的规格说明项并设计测试用例。

TA-3.2.3 (K4) 通过应用判定表测试来分析给定的规格说明项并设计测试用例。

TA-3.2.4 (K4) 通过应用状态转换测试分析给定的规格说明项并设计测试用例。

TA-3.2.5 (K2) 解释分类树图是如何支持测试技术的。

TA-3.2.6 (K4) 通过应用结对测试分析给定的规格说明项并设计测试用例。

TA-3.2.7 (K4) 通过应用用例测试分析给定的规格说明项并设计测试用例。

TA-3.2.8 (K4) 分析系统，或其需求规格说明，以确定预期的缺陷类型，并选择适当的黑盒测试技术。

3.3 基于经验的检测技术

TA-3.3.1 (K2) 解释基于经验测试技术的原理，以及其与黑盒测试、基于缺陷的测试技术相比的优缺点。

TA-3.3.2 (K3) 从给定的场景中识别探索性测试。

TA-3.3.3 (K2) 描述基于缺陷的测试技术的应用，并与黑盒测试技术相区分。

3.4 使用最恰当的测试技术

TA-3.4.1 (K4)根据给定的项目情况，确定应该应用哪些黑盒测试技术或基于经验的测试技术来实现特定目标。

3.1 简介

本章中涉及到的测试技术分为以下几类：

- 黑盒
- 基于经验

这些技术是互补的，在任何测试活动、测试级别均适用。

注意，这两类技术都可用于测试功能性和非功能性质量特性。下一章将讨论测试软件特性。

在这些章节中讨论的测试技术主要集中于确定最优的测试数据(例如，来自等价类区间)或生成测试规程(例如，来自状态模型)。通常结合多种技术来创建完整的测试用例。

3.2 黑盒测试技术

黑盒测试技术在ISTQB®基础级教学大纲[ISTQB_FL_SYL]中有介绍。

黑盒测试技术的共同特点包括：

- 在测试设计过程中根据测试技术创建模型，例如状态转换图和判定表
- 从这些模型中系统地推导出测试条件

测试技术通常提供可以用于度量测试设计和测试执行活动的覆盖准则。完全满足覆盖准则并不意味着整个测试集是完整的，而是该模型不再建议增加基于该技术的测试来提高覆盖。

黑盒测试通常基于某种形式的规格说明文档，例如系统需求规格说明或用户故事。由于规格说明文档应该描述系统行为，特别是在功能适用性领域，从需求推导测试通常是测试系统行为的一部分。在某些情况下，可能没有规格说明文档，但是有隐含的需求，比如替换遗留系统的功能适用性。

黑盒测试技术有很多种。这些技术针对不同类型的软件和场景。以下章节展示了每种技术的适用范围、测试分析师可能遇到的限制和难点、度量覆盖率的方法以及目标缺陷类型。

详情请参阅[ISO29119-4]、[Bath14]、[Beizer95]、[Black07]、[Black09]、[Copeland04]、[Craig02]、[Forgács19]、[Koomen06]和[Myers11]。

3.2.1等价类划分

等价类划分(EP)是一种用于精简所需测试用例数量的技术，可有效处理测试输入、输出、内部值以及

与时间有关的值。分类用来创建等价类，等价类是以相同方式处理值集合。通过从一个（等价类）分类中选择一个代表值，就可以假定覆盖了同一分类中的所有值。

通常，测试对象的行为由几个参数决定。当通过组合不同参数的等价类生成测试用例时，可以应用多种技术。

适用性

此技术适用于任何测试级别，当程序会采用同样方式处理待测数据集合中的任何一个数据，而这些数据无交互的时候尤其适用。等价类可以是任意非空值集合，例如：有序、无序、离散、连续、无限、有限甚至单例。值集的选择适用于有效和无效等价类(即，被测软件视为无效的值的分区)。

当与边界值分析结合使用时，测试数据涵盖了分类的边界，使用等价类划分（EP）的作用是最大的。在新构建或新版本的冒烟测试中，等价类划分（使用有效分区的值）是一项常规技术，因为它可以快速确定基本功能是否能正常工作。

限制/难点

如果假设不成立，如等价类中的值没有以完全相同的方式处理，则此技术可能会遗漏缺陷。所以仔细挑选等价类非常重要。例如，输入可为正数，也可为负数的时候，最好将正数、负数划分为两个不同的等价类，因为处理方式可能会不一样。根据是否允许为零，考虑是否增加一个等价类。对测试分析师来说，为了确定值的最佳等价类划分，理解底层处理是很重要的，这可能需要代码设计方面的支持。

测试分析师还应考虑不同参数之间等价类划分可能存在的依赖关系。例如，在航班预订系统中，参数“成人陪同”只能与年龄段“儿童”绑定使用。

覆盖率

覆盖率由已测试值的等价类个数除以识别出的等价类总数测算。等价类划分覆盖率以百分比表示。测试单个等价类的多个代表值不会增加测试覆盖率。

如果测试对象的行为依赖于单个参数，则每个等价类（无论有效还是无效）都应至少被覆盖一次。

如果有多个参数，则测试分析师应根据风险[Offutt16]选择一个简单的或组合的覆盖类型。因此，区分仅包含有效等价类的组合和包含一个或多个无效等价类的组合至关重要。对于仅包含有效等价类的组合，最低要求是简单覆盖所有参数的全部有效类。假设参数彼此无关，则此类测试套件中所需的最小测试用例数量等于一个参数的有效等价类的最大数量。与组合技术有关的更全面的覆盖类型包括结对覆盖（请参阅下面的3.2.6节）或有效等价类的任何组合的完整覆盖。无效的等价类至少应单独测试，即与其他参数的有效等价类结合使用，以避免缺陷屏蔽。因此，每个无效等价类都会为简单覆盖的测试套件贡献一个测试用例。在高风险的情况下，可以将其他组合添加到测试套件中，例如仅由无效等价类或结对的无效等价类组成。

典型的缺陷

测试分析师使用此技术来发现处理各种数据值的缺陷。

3.2.2 边界值分析

边界值分析(BVA)用于对有序划分的等价类在边界上的值的正确处理进行测试。边界值分析有两种常用测试方法：二值边界法或三值边界法。通过二值边界法测试，使用边界值(边界上的)和边界外的值(根据要求的准确度，以尽可能小的精度)。例如，对于具有两位小数的货币金额，如果等价类包括从1到10的值，则二值法的上边界测试值将为10和10.01。下边界测试值将为1和0.99。边界由等价类划分中定义的最大值和最小值确定。

对于三值边界法测试，取一个小于边界、在边界上和一个大于边界的值。在前面的示例中，上边界测试值将包括9.99、10和10.01。下边界测试值将包括0.99、1和1.01。采用二值边界法还是三值边界法取决于被测试项的风险大小，风险高的采用三值边界法。

适用性

该技术适用于任何测试级别，尤其适用于存在有序等价类划分的情况。因此，边界值分析技术通常与等价类划分技术结合使用。由于存在边界上和边界外的概念，因此需要等价类有序划分。例如，一组数字是有序的等价类。由一些文本字符串组成的等价类也可以排序，例如按其字典顺序排序，但如果从业务逻辑而言，该排序并无意义，则不应关注边界值。除了数值范围外，可以应用边界值分析的等价类还包括：

- 非数值变量的数值属性(如长度)
- 循环执行周期的数量，包括状态转换图中的循环
- 在存储的数据结构（如数组）中迭代元素的数量
- 物理对象的大小（例如内存）
- 活动持续时间

限制/难点

由于该技术正确识别边界值的准确性取决于等价类划分的准确性，因此其限制和难点与等价类划分相仿。测试分析师还应该了解有效值和无效值的精度，以便能够准确地确定要测试的值。只有有序的等价类可以用于边界值分析，但并不限于有效输入的范围。例如，在测试电子表格支持的单元数时，有一个等价类包含最大允许的单元数(边界)，另一个等价类以超出最大允许的单元数(超出边界)开始。

覆盖率

覆盖率是由被测试的边界条件数除以识别的边界条件总数确定的(使用二值边界法或三值边界法)。覆盖率用百分比表示。

典型的缺陷

边界值分析可以准确地发现边界的偏移或遗漏，而且可能会发现边界额外的情况。这种技术有助于发现关于边界值处理的缺陷，尤其是小于和大于逻辑值（即偏移）的错误。它还可以用于查找非功能性缺陷，例如，系统支持10000个并发用户，而不是10001个。

3.2.3判定表测试

判定表由一组条件和相关动作的表格组成，表示为一组条件值[OMG-DMN]在确定的规则下将发生的动作。测试分析师可以使用判定表来分析适用于被测软件的规则，并设计测试以覆盖这些规则。

测试对象的条件和动作构成判定表的行，通常条件在顶部，动作在底部。表的第一列分别对条件和动作进行描述。下面的列称为规则，分别包含条件值和相应的动作值。

条件为布尔值且取值仅为“ 真”和“ 假”的判定表称为有限条目判定表。例如，条件是“用户的收入<1000”。扩展条目判定表允许条件具有多个值，这些值可以表示离散元素或元素集。例如，“用户收入”的条件可以取三个可能的值之一：“低于1000”，“ 1000至2000之间”和“高于2000”。

简化动作采用布尔值“ 真”和“ 假”（例如，“ 允许的折扣 = 20%”这一动作，如果应该发生则值为“真”，用“ X”表示；如果不应发生，则值为“ 假”，用“-”表示）。就像条件一样，动作也可能从其他领域取值。例如，“允许的折扣”动作可以采用以下五个可能的值之一：0%，10%，20%，35%和50%。

判定表测试从基于规格说明的判定表设计开始。包含无法执行的条件值组合列被排除或标记为“无法执行”。接下来，测试分析师应与其他利益相关方一起评审判定表。测试分析师应确保表中的规则是一致的（即规则不重叠）、完整的（即包含了针对条件值的每种可行组合的规则）以及正确的（即对预期的行为进行了建模）。

判定表测试的基本原则是，规则是从测试条件中得来的。

当设计一个测试用例以覆盖给定规则时，测试分析师应意识到该测试用例的输入值可能与判定表的条件值不同。例如，条件“年收入> 100,000？”的“ 真”值，可能不能直接使用，但可能需要测试人员定义一个在给定会计年度中信用额度大于100,000的客户账户。同样，测试用例的预期结果可能与判定表中的动作不同。

准备好判定表后，需要通过选择满足条件和动作的测试输入值（和预期结果），将规则作为测试用例实施。

优化判定表

当根据条件尝试测试每个可能的输入组合时，判定表可能会变得非常大。具有n个条件的完整的有限条目判定表有2^n 条规则。一种系统地减少组合数量的技术称为优化判定表测试[Mosley93]。当使用此技术时，如果一组规则中的某些条件的取值与该规则下的动作无关，且当所有其他条件保持不变时，这组规则具有相同的动作集合，则可以将这组规则合并（优化）为一个规则。在这条新规则中，无关条件的值表示为“不关心项”，通常用短划线“-”标记。对于具有“不关心项”值的条件，测试分析师可以为测试实施指定任意有效值。

优化规则适用的另一种情况是，条件值不适用于某些其他条件值的组合，或者两个或多个条件具有冲突值。例如，在用于卡支付的判定表中，如果条件“卡有效”为假，则条件“ PIN码正确”不适用。

优化判定表上的规则可能比完整的判定表上的规则少很多，这样所需的测试用例和工作量会更少。如果给定规则包含“不关心项”条目，并且只有一个测试用例覆盖了该规则，则其只测试了符合规则的几个可能值之一，因此可能无法测试出涉及其他值的缺陷。因此，对于高风险级别，测试分析师应与测试经理协商一致，为单个条件值的每种可能组合定义单独的规则，而不是优化判定表。

适用性

判定表测试通常应用在集成、系统和验收测试级别。如果组件负责处理一组判定逻辑时，它也可适用于组件测试。当以流程图或业务规则表的形式描述测试对象时，此技术特别有用。

判定表也是一种需求定义技术，有时候需求规格说明可能已经以这种格式定义了。测试分析师应参与评审判定表，并在开始测试设计之前，对其进行分析。

限制/难点

在考虑条件组合时，要找到所有相互作用的条件是有挑战性的，特别是在需求没有很好的定义或不存在时。在判定表中选择条件时必须谨慎，以便使这些条件的组合数量保持可控。在最坏的情况下，规则的数量将呈指数增长。

覆盖率

该技术的通用覆盖准则是，判定表里的每个规则都有一个对应的测试用例。覆盖率以测试套件覆盖的规则数量和与可运行规则总数的百分比来衡量。

边界值分析和等价类划分可以与判定表技术结合使用，尤其是在扩展判定表输入条件的情况下。如果条件包含完全有序的等价类，则边界值可以生成额外的输入条件和测试用例。

缺陷类型

典型缺陷包括基于特定条件组合导致的错误的逻辑关系处理，从而引发非预期结果的缺陷。在判定表的创建过程中，可能会在规格说明文档中发现缺陷。一组条件中的一个或多个规则的预期结果未描述的情况并不少见。最常见的缺陷类型是动作遗漏(即特定情况下未描述预期结果)和矛盾。

3.2.4 状态转换测试

状态转换测试用于测试被测对象通过有效转换，进入和退出已定义状态的能力，以及尝试进入无效状态或覆盖无效转换的能力。事件导致测试对象从一个状态转换到另一个状态，并执行操作。事件可以通过影响转换路径的条件(有时称为守卫条件或转换监控)来限定。例如，使用有效用户名/密码的登录事件与使用无效密码的登录事件会产生不同的转换。该信息会在状态转换图或状态转换表中表示(也可能包括状态之间潜在的无效转换)。

适用性

状态转换测试适用于任何有状态定义并让状态之间因事件发生转换的软件(例如，屏幕变化)。状态转换测试可以在任何测试级别上使用。嵌入式软件、web软件和任何状态转换类软件都适合进行此类测试。控制系统，例如交通灯控制器，也是这类测试很好的候选者。

限制/难点

确定状态通常是定义状态转换图或状态转换表最困难的部分。当测试对象具有用户界面时，屏幕对用户的不同显示通常用状态来代表。对于嵌入式软件，状态可能依赖于硬件的状态。

除了状态本身之外，状态转换测试的基本单元是单个转换。简单地测试所有的单个转换将发现一些状态转换缺陷，但是通过测试状态转换序列可能会发现更多的缺陷。单个转换被称为0-切换（0-switch），两次连续的转换序列称为1-切换（1-switch）；三次连续的转换序列称为2-切换（2-switch），以此类推。通常，N切换（N-switch）表示N + 1次连续切换[Chow1978]。随着N的增加，N切换的数量增长非常快，从而难以通过合理的少量测试来实现N切换的覆盖。

覆盖率

与其他类型的测试技术一样，状态转换覆盖率有一个层次结构。可接受的最低覆盖率是指到达过每个状态和遍历过每个转换至少一次。100%的转换覆盖（又称100%的0-切换覆盖率）将保证到达过每个状态和遍历过每个状态转换，除非系统设计或状态转换模型(图或表)有缺陷。根据状态和转换之间的关系，为了执行某个转换一次，可能需要多次遍历某些转换。

术语“N-切换覆盖率”是指长度为N+1的转换所覆盖的数量，占该长度的转换总数的百分比。例如，要实现100%的1-切换覆盖率，每个有效序列需要至少测试两次连续的转换。这个测试可能会触发某些类型的失效，如只达到100%的0-切换覆盖率将会漏掉的失效。

“往返覆盖”适用于转换序列形成循环的情况。实现100%往返覆盖意味着已经测试了从任何状态出发又回到原来相同状态的所有循环。此循环不包含任何特定状态(不包括初始/最终状态)[Offutt16]的多次出现。

对于上述的任何一种方法，当尝试包括状态转换表中标识的所有无效转换时，可以得到更高的覆盖率。状态转换测试的覆盖需求和覆盖集必须确定是否包括无效的转换。

状态转换图或状态转换表可以支持设计测试用例，以便让特定测试对象的覆盖率达到期望值，这些信息也可以用能够显示特定值“N”[Black09]的N-切换状态表表示。

可以利用手工过程来识别需要的项(例如，转换、状态或N-切换)。建议的方法是打印状态转换图和状态转换表，并使用钢笔或铅笔标记所需覆盖的项，直到达到所需的覆盖率[Black09]。对于更复杂的状态

转换图和状态转换表来说，这种方法会变得太耗时。因此，应该使用工具来支持状态转换测试。

缺陷类型

典型缺陷包括（参见[Beizer95]）：

- 不正确的事件类型或值
- 不正确的操作类型或值
- 不正确的初始状态
- 无法达到某种退出状态
- 无法进入所需状态
- 额外的(不必要的)状态
- 无法正确执行某些有效的转换
- 能够执行无效的转换
- 错误的守卫条件

在状态转换模型的创建过程中，可能会在规格说明文档中发现缺陷。最常见的缺陷类型是遗漏(即没有关于在特定情况下实际应该发生什么的信息)和矛盾。

3.2.5 分类树技术

分类树通过对测试对象的数据空间以图形化的展现方式来支持某些黑盒测试技术。

数据按如下方式分成分类和类：

- 分类：代表测试对象数据空间中的参数。如输入参数（可以进一步包含环境状态和前置条件）和输出参数。例如，如应用程序可以以多种不同的方式配置，则分类可能包括客户端、浏览器、语言和操作系统。
- 类：每个分类可以有任意数量的类和子类用来描述当前参数。每个类或等价类都是分类中的一个特定值。在上面的例子中，语言分类可包括英语、法语和西班牙语的等价类。

分类树允许测试分析师输入他们认为合适的组合。例如，这包括结对组合(见3.2.6节)、三元（three-wise）组合和单一组合。

[Bath14]和[Black09]中提供有关使用分类树技术的更多信息 。

适用性

分类树的创建可以帮助测试分析师识别感兴趣的参数（分类）和对应的等价类 (类)。

对分类树图的进一步分析可以确定可能的边界值，并识别特定的输入组合，这些组合要么是特别感兴趣的，要么是不重要的(例如，因为它们相互不兼容)。由此产生的分类树可以用来支持等价类划分、边界值分析或结对测试(见3.2.6节)。

限制/难点

随着分类和/或类数量的增加，图表会变得更大，更不容易使用。此外，分类树技术不创建完整的测试用例，而是创建测试数据组合。测试分析师必须为每个测试组合提供结果，以生成完整的测试用例。

覆盖率

测试用例设计需达到覆盖率要求，例如，最小的类覆盖率(即，分类中的所有值至少测试一次)。测试分析师也可能决定使用成对组合进行覆盖，或使用其他类型的组合测试如三元组合。

缺陷类型

发现的缺陷类型取决于分类树支持的技术(例如，等价类划分、边界值分析或结对测试)。

3.2.6结对测试

当所测试的软件中存在多个输入参数，每个参数又有多个可能的值，而且需要测试它们的组合时，就可使用结对测试，从而能在允许的时间内测试更多的组合。输入参数可能是独立的，任何参数的任何可选值（即任一输入参数的任何选定值）可以与任何其他参数的任何可选值组合，然而实际情况并不总是如此（见下面特征模型的说明）。特定参数与该参数的特定值的组合称为“参数值对”(例如，如果“color”是具有包含“red”在内的七个允许值的参数，那么“color = red”可能是一个“参数值对”)。

结对测试使用组合技术来确保每个“参数值对”与其他参数的每个“参数值对”至少测试一次(即，测试任意两个不同参数的“参数值对”的“所有对”)，同时避免测试“参数值对”的所有组合。如果测试分析师使用手工方法，首先将构建一个表，其中的行代表测试用例，每一列对应一个参数。然后测试分析师用（参数）值填充表，这样可以在表中识别所有的值对(参见[Black09])。表格中的任何留空白的条目都可以由测试分析师使用他们自己的领域知识填充值。

有许多工具可以帮助测试分析师完成此任务(参见www.pairwise.org获取示例)。它们需要参数及其值的列表作为输入，并从每个参数生成一组合适的值组合，覆盖所有“参数值对”。工具的输出可以作为测试用例的输入。请注意，测试分析师必须为工具创建的每个组合提供预期的结果。

分类树(见3.2.5节)经常与结对测试一起使用[Bath14]。分类树设计通常由工具支持，可以使参数及其值的组合可视化(一些工具提供增强型结对)。这有助于识别下列信息：

- 结对测试技术使用的输入值。
- 感兴趣的特定组合(例如，经常使用或常见的缺陷来源)。
- 不兼容的特定组合。这并不意味着组合的参数不会相互影响；它们很可能会相互影响，但应该会以可接受的方式相互影响。

- 变量之间的逻辑关系。例如，“如果变量1 = x，那么变量2不能是y”。能捕获这些关系的分类树称为“特征模型”。

适用性

至少在如下与测试相关的两种不同的情况下会产生参数值组合过多的问题。有些测试项涉及多个参数，每个参数都有多个可能的值，例如具有多个输入字段的屏幕。在这种情况下，参数值的组合构成了测试用例的输入数据。此外，一些系统可以在多个维度上进行配置，从而可能导致较大的配置空间。在这两种情况下，可以使用结对测试来确定可管理和可接受的组合子集。

对于具有多值的参数，可以先对每个参数分别应用等价类划分或其他选择机制，以减少每个参数值的数量，然后再应用结对测试，以降低组合结果集的规模。在分类树中捕获参数及其值能支持此活动。

这些技术通常应用于组件集成、系统和系统集成测试级别。

限制/难点

这些技术的主要限制是假设了少数测试的结果可以代表所有测试的结果，并且这些少数测试代表了预期的使用方法。如果某些变量之间存在非预期的交互，假如没有测试这些特定的组合，则该测试技术可能无法测试到这些非预期交互。这些技术很难向非技术受众解释，因为他们可能不理解用逻辑简化的测试。任何这样的解释都应该通过提供实证研究[Kuhn16]的结果来加以佐证，在医疗器械研究领域，该实证结果表明，66%的失效是由单一变量触发的，97%是由一个变量或两个变量交互触发的。结对测试无法发现由三个或更多的变量交互触发的系统失效，这是一个残余风险。

有时很难确定参数及其各自的值，因此应该尽可能借助于分类树的支持来执行这项任务(见3.2.5节)。找到一组最小的组合来满足一定程度的覆盖是很难手工完成的，但可以借助工具寻找最小的组合。有些工具支持强制将某些特定组合包括在最终选择的组合中或从最终选择的组合中排除。测试分析师可以根据领域知识或产品使用信息使用此功能来强化或弱化某些组合。

覆盖率

100%结对的覆盖率要求任意一组结对的参数值至少包含在一个组合中。

缺陷类型

使用这种测试技术发现的最常见的缺陷类型是与两个参数的组合值相关的缺陷。

3.2.7用例测试

用例测试提供事务性的、基于业务场景的测试，它应该可以通过用例模拟特定的组件或系统的预期用途。用例是根据参与者和实现某个目标的组件或系统之间的交互来定义的。参与者可以是人类用户、外部硬件或其他组件或系统。

[OMG-UML]中提供了用例的通用标准。

适用性

用例测试通常应用在系统测试和验收测试中。如果可以用用例描述组件或系统的行为，则也可以在集成测试中使用。用例通常也是性能测试的基础，因为它们描述了系统的实际使用情况。用例中描述的场景可以分配给虚拟用户，以便在系统上创建一个真实的负载(只要在用例中已经定义了负载和性能的需求)。

限制/难点

为了有效，用例必须表达真实的用户业务。用例规格说明是系统设计的一种形式。描述用户需要完成什么的需求应该来自用户或用户代表，并且在设计相应的用例之前应该对照组织级需求进行检查。如果用例不能真实反映用户和组织级需求，或者阻碍而不是协助用户完成任务，那么用例的价值就会降低。

对异常、可替代性和错误处理机制的准确定义对于达到全面的覆盖率非常重要。用例应该是一个指南，而不应该是对测试内容的完整定义，因为它可能并不能提供所有需求的清晰定义。从用例描述中创建其他模型（如流程图和/或判定表）对于提高测试的准确性和验证用例本身都是有益的。与其他形式的规格说明相似，这有机会暴露出用例规格说明中可能存在的逻辑异常。

覆盖率

用例测试的最小可接受覆盖率是：每个基本行为（基本流）有一个测试用例，另有足够多的附加测试用例能覆盖每个备选行为（备选流）和错误处理机制。如果需要一个最精简的测试套件，则可以合并多个备选行为到一个测试用例中，前提是它们可以相互兼容。如果需要更好的诊断能力(例如，帮助隔离缺陷)，可以为每个备选行为设计一个额外的测试用例，尽管嵌套的备选行为仍然需要将一些行为合并到单个测试用例中(例如，“重试”异常操作中的终止与非终止的可替代操作)。

缺陷类型

典型的缺陷包括对定义的行为处理不当、遗漏的替代行为、对当前条件的错误处理以及执行不力或不正确的错误提示。

3.2.8技术的组合

有时会结合使用多种技术创建测试用例。例如，可以对判定表中标识的条件进行等价类划分，以便找到满足条件的多种方式。测试用例不仅可以覆盖每种条件组合，而且可以生成额外的测试用例来覆盖那些被划分的等价类条件。在选择使用一项具体的技术时，测试分析师应该考虑该技术的适用性、限制或难点，以及测试要达成的覆盖率目标和缺陷发现目标。本章对这些个别的技术进行了描述。对于某种情况，可能没有单一的“最佳”技术。如果有足够的时间和技能来正确地应用这些技术，组合技术通常会提供最完整的覆盖。

3.3 基于经验测试技术

基于经验的测试是利用测试人员的技能和直觉，以及他们对类似应用程序或技术的经验，以达到更好发现缺陷的测试目标。这些测试技术的适用范围从测试人员不用预先做正式测试计划的“快速测试”，到使用测试章程制定预先计划的测试会话，再到脚本化的测试会话。基于经验的测试几乎总是有用的，但是如果可以实现如下所列优点的话，它们将更具有特殊的价值。

基于经验的测试具有以下优点：

- 在缺乏系统文档的情况下，它可能是一种替代结构化方法的好方案。
- 可在测试时间受到严格限制的情况下使用。
- 它能够把该领域的专业知识和技术应用在测试上，它可以将那些不参与测试的人纳入进来，例如，业务分析师、客户或当事人。
- 它可以向开发人员提供早期反馈。
- 它有助于团队在软件生产过程中熟悉软件。
- 可有效分析操作故障。
- 能够应用于多种测试技术。

基于经验的测试有以下缺点：

- 可能不适用于需要详细测试文档的系统。
- 难以实现高水平的可重复性。
- 准确评估覆盖率的能力有限。
- 测试不太适合后续的自动化。

当使用反应式和启发式方法时，测试人员通常使用基于经验的测试，这比预先计划好的测试方法更能有效应对事件。此外，执行和评估是并行任务。一些基于结构的测试方法相对于基于经验的方法而言并不完全是动态的，即并不是完全在创建测试的同时执行测试。例如，在测试执行之前使用错误猜测来定位测试对象的特定问题，可能就是这种情况。

请注意，尽管这里讨论的技术提供了一些关于覆盖率的想法，但是基于经验的测试技术没有正式的覆盖率标准。

3.3.1 错误猜测

当使用错误猜测技术时，测试分析师使用经验来猜测在设计和开发代码时可能出现的潜在错误。当识别出预期的错误后，测试分析师就确定了用来发现产生缺陷的最佳方法。例如，如果测试分析师预期输入无效的密码时，该软件将产生失效，那他将在运行测试时输入各种不同的值来验证这个错误是否真实发生了，从而导致了在运行测试时可视为失效的缺陷。

除了作为一种测试技术使用之外，错误猜测在风险分析中也很有用，可以识别潜在的失效模式(Myers11)。

适用性

错误猜测主要在集成和系统测试期间进行，但可以用于任何测试级别。此技术经常与其他技术一起使用，并有助于扩展现有测试用例的范围。在测试软件的新版本时，也可以有效地使用错误猜测，以便在开始执行更严格的和脚本化的测试之前测试出常见的缺陷。

限制/难点

错误猜测存在以下限制和难点：

- 覆盖率很难评估，并且随着测试分析员的能力和经验不同而有很大差异。
- 最好由熟悉这类被测代码中常见缺陷类型的，经验丰富的测试人员使用。
- 它常常被使用，但又经常没有记录，因此与其他形式的测试相比更不具有可重复性。
- 测试用例可能会被文档化，但只有作者才能理解和重现。

覆盖率

当使用缺陷分类时，覆盖率是通过将已测试的分类项数量除以分类项的总数得到的，并以百分比的形式说明。如果没有缺陷分类，覆盖率将受限于测试人员的经验、知识以及可用时间。通过这种技术发现缺陷的数量会由于测试人员针对问题区域的能力高低而有所不同。

缺陷类型

典型的缺陷通常是那些在缺陷分类中定义的或者由测试分析师“猜测”的缺陷，这些缺陷可能在黑盒测试中没有被发现。

3.3.2 基于检查表的测试

当应用基于检查表的测试技术时，经验丰富的测试分析师会使用一个概要的、广义的清单，罗列出需关注、检查或记住的事项，或使用针对测试对象必须验证的一组规则或准则。这些检查表是基于一组标准、经验和其他需考虑因素构建的。例如，一个用户界面标准检查表可以用作测试应用程序的基础。在敏捷软件开发中，可以根据用户故事的验收标准构建检查表。

适用性

如果项目的测试团队经验丰富，且熟悉被测软件或检查表所覆盖的领域，则使用基于检查表的测试是最有效的（例如，成功应用用户界面检查表，测试分析师可能只熟悉用户界面测试，但不熟悉特定的被测系统）。因为检查表是概要的，而且往往缺乏测试用例和测试规程中常见的详细步骤，所以测试人员的知识可以用来填补空白。通过删除详细的步骤，维护检查表是低成本的，可以在多个类似的版本中使用。

检查表非常适合于软件发布和变更迅速的项目。这有助于减少测试文档的准备和维护时间。它们可以用于任何测试级别，也可以用于回归测试和冒烟测试。

限制/难点

检查表天然的概要性会影响测试结果的可重复性。不同的测试人员可能会对检查表有不同的诠释，并采用不同的方法来完成检查项。即使使用完全一样的检查表，也可能导致不同的测试结果。尽管这会提升覆盖率，但有时会牺牲可重复性。由于实际测试效果取决于测试人员的主观判断，因此检查表可能导致对达到的覆盖率水平过度自信。检查表可以从更详细的测试用例或列表导出，并随着时间的推移而增长。检查表需要维护以确保覆盖了所测试软件的重要方面。

覆盖率

覆盖率可以通过将已测试的检查项数量除以检查项总数来确定，并以百分比形式表示。覆盖率高低依赖于检查表的质量，但是，由于检查表天然的概要性，结果会因执行检查表的测试分析师不同而不同。

缺陷类型

使用该技术可以发现在测试过程中由于数据、步骤序列或一般工作流变化导致失效发生的典型缺陷。

3.3.3探索性测试

探索性测试的特点是测试人员了解测试对象和发现缺陷同时完成，即同步进行测试的计划、设计和执行，并报告结果。测试人员在执行中动态调整测试目标，并且只准备轻量级的文档。(Whittaker09)

适用性

好的探索性测试是有计划的、交互式的和创造性的。它几乎不需要与被测试系统有关的文档，经常被用于文档不可用或不适合使用其他测试技术的情况。探索性测试常作为其他测试技术的补充，并作为开发附加的测试用例的基础。在敏捷软件开发中经常使用探索性测试，目的是用很少的文档就能灵活、快速地完成用户故事测试。而且该技术也可用于使用顺序开发模型的项目中。

限制/难点

探索性测试的覆盖范围可能是非连续的，而且可重复性较差。使用测试章程（test charters）标注需要在测试会话（test session）中覆盖的区域，使用时间盒（Time-Boxing）标注允许测试的时长，用这些技术来管理探索性测试。在一个测试会话或一组测试会话结束时，测试经理可以通过主持一个事后说明会来收集测试结果并确定下一测试会话的测试章程。

在测试管理系统中准确地跟踪探索性测试会话是另一个难点，有时是通过创建实际上是探索性测试会话的测试用例来实现的。这允许把分配给探索性测试的时间和计划的覆盖范围与其它测试工作量一起进行跟踪。

由于探索性测试的可重复性较差，当需要重现失效步骤时可能会遇到问题。一些组织使用具有捕获/回放功能的测试自动化工具来记录探索性测试人员的操作步骤。这提供了探索性测试会话(或任何基于经验的测试会话)期间所有活动的完整记录。分析细节找到引起失效的真实原因可能很繁琐，但至少有涉及的所有步骤的记录。

其他工具可能用于捕获探索性测试会话，但是这些工具不会记录预期结果，因为它们不会捕获图形用户界面（GUI）的交互。在这种情况下，必须记录预期的结果，以便在需要时可以对缺陷进行适当的分析。一般而言，建议在执行探索性测试时进行记录，以便在需要时支持可重复性。

覆盖率

测试章程可以针对特定的任务、目标和可交付成果而设计，然后计划探索性测试会话来达到这些要求。章程也可以确定测试工作的重点、测试会话包含和不包含的内容、承诺完成计划的测试所需的资源。测试会话可聚焦在用于发现特定的缺陷类型和其他有潜在问题的区域，而这些区域不需要正式的脚本化的测试就可以处理。

缺陷类型

使用探索性测试发现的典型缺陷有：在脚本化的功能适合性测试中遗漏的基于场景的问题、介于功能边界之间的问题，以及与工作流程相关的问题。在探索性测试期间，有时也能发现性能和安全问题。

3.3.4基于缺陷的检测技术

基于缺陷的测试技术是将所寻找的缺陷类型作为测试设计的基础，从已知的缺陷类型中系统地推导出测试的技术。与从测试依据中导出测试的黑盒测试不同，基于缺陷的测试从关注缺陷的列表中导出测试。通常，列表可由缺陷类型、根本原因、失效症状和其他缺陷相关的数据组成。标准列表适用于多种类型的软件，并不针对特定产品。使用这些列表有助于利用行业标准知识来导出特定的测试。通过遵循行业特定的列表，可以对缺陷的发生进行跨项目甚至跨组织的跟踪度量。最常见的缺陷列表是那些使用了特定的专业知识和经验的组织或项目的列表。

基于缺陷的测试也可以使用已识别的风险和风险场景的列表作为目标测试的基础。该测试技术让测试分析师可以针对某个特定的缺陷类型，也可以利用已知和常见的特定缺陷类型列表系统地进行测试。通过这些信息，测试分析师创建能导致缺陷显现出来的（如果缺陷存在）测试条件和测试用例。

适用性

基于缺陷的测试可以应用于任何测试级别，但最常用于系统测试。

限制/难点

存在多种缺陷分类方法，可能更侧重于特定类型的测试，例如易用性。选择一个适用于被测软件的分类法（如果有）是很重要的。例如，对于创新软件可能没有任何现成可用的分类。有些组织编制了适合自身的可能或常见缺陷分类。无论使用什么缺陷分类法，在开始测试之前定义预期的覆盖率是很重要的。

覆盖率

该技术提供了用于确定所有有效的测试用例何时被识别出的覆盖准则。依据缺陷列表，覆盖项可以是结构元素、规格说明元素、使用场景，或者这些元素的任何组合。实际上，与黑盒测试技术相比，基于缺陷的测试技术的覆盖准则往往不那么系统，因为它只给出了覆盖的一般规则，且关于由什么可以构成有效

覆盖范围的具体描述是酌情作出决定的。与其他技术一样，有覆盖准则并不意味着整个测试集是完整的，更确切地说，基于该技术无法提供更多有用的测试，用于发现期望的缺陷。

缺陷类型

发现的缺陷类型通常取决于使用的缺陷分类法。例如，如果使用了用户界面缺陷列表，那么发现的大多数缺陷可能与用户界面相关，除此之外发现的其它缺陷可以作为特定测试的副产品。

3.4应用最合适的技术

黑盒测试技术和基于经验的测试技术结合使用时最有效。基于经验的测试技术填补了黑盒测试技术中存在的系统性弱点而导致的覆盖空白。

没有一种适用于所有情况的完美技术。对于测试分析师而言，最重要的是了解每种技术的优缺点，在考虑到项目类型、进度、信息获取、测试人员的技能和其他可以影响选择因素的情况下，能选择出最佳技术或技术组合。

在每个黑盒和基于经验的测试技术的讨论中（分别见第3.2节和第3.3节），在“适用性”、“限制/难点”和“覆盖率”中提供的信息可以指导测试分析师选择最合适的测试技术。

4.测试软件质量特性–180分钟

关键字

易访问性（accessibility*）、兼容性（compatibility*）、功能适合性（functional appropriateness*）、功能完备性（functional completeness*）、功能正确性（functional correctness*）、功能性（functional suitability*）、互操作性（interoperability*）、易学性（learnability*）、易操作性（operability*）、软件易用性度量调查表（Software Usability Measurement Inventory (SUMI)），易用性（usability*）、用户差错防御性（user error protection*）、用户体验（user experience）、用户界面舒适性（user interface aesthetics*）、网站分析和度量调查表（Website Analysis and MeasureMent Inventory（WAMMI））。

说明：*表示是按“中华人民共和国国家标准 GB/T 25000.10-2016” 翻译。

软件测试质量特性的学习目标

4.1 简介

无学习目标

4.2 业务领域测试的质量特性

TA-4.2.1（K2）解释适用于测试功能完备性、功能正确性和功能适合性的测试技术。

TA-4.2.2（K2）定义针对功能完备性、功能正确性和功能适合性特性的典型缺陷。

TA-4.2.3（K2）定义了在软件开发生命周期中进行功能完备性、功能正确性和功能适合性的测试时机。

TA-4.2.4（K2）解释能够同时验证和确认实现易用性需求并满足用户期望的适用方法。

TA-4.2.5（K2）解释测试分析师在互操作性测试中的角色，包括识别目标缺陷。

TA-4.2.6（K2）解释测试分析师在可移植性测试中的角色，包括识别目标缺陷。

TA-4.2.7（K4）对于一组给定的需求，测试分析师在圈定的范围内，确定可以验证功能和/或非功能质量特性的测试条件。

4.1介绍

虽然上一章描述了测试人员可用的特定技术，但本章讨论的是，在评估用于描述软件应用程序或系统的质量的特性时，如何应用这些技术。

本教学大纲讨论了可由测试分析师评估的质量特性。技术测试分析师所要评估的属性包含在《高级技术测试分析师教学大纲》[CTAL-TTA]中。

使用ISO25010（GB/T 25000.10）中提供的产品质量特性描述作为指导性的描述。ISO软件质量模型将产品质量分为不同的产品质量特性，每个特性都可能有子特性。这些特性如下表所示，并说明了测试分析师和技术测试分析师教学大纲中覆盖的特性和子特性。

特性	子特性	测试分析师	技术测试分析师
功能性	功能正确性、功能适合性、功能完备性	X	
可靠性	成熟性、容错性、易恢复性、可用性		X
易用性	适当性和可辨识性、易学性、易操作性、用户界面舒适性、用户差错防御性、易访问性	X	
性能效率	时间特性、资源利用性、容量		X
维护性	易分析性、易修改性、易测试性、模块化、可重用性		X
可移植性	适应性、易安装性、易替换性	X	X
信息安全性	保密性、完整性、抗抵赖性、可核查性、真实性		X
兼容性	共存性		X
	互操作性	X	

虽然不同的组织中工作的安排可能会有所变化，但在相关的ISTQB®教学大纲中我们遵照以上描述。

对于本节讨论的所有质量特性和子特性，必须识别出典型的风险，这样才能形成恰当的测试策略并记录下来。对质量特性的测试需要特别注意软件生命周期的时间安排、所需工具、软件和文档可用性以及技术专长。如果没有处理每个特性及其独特测试需求的策略，测试人员可能无法在进度表中安排充足的时间来进行计划、提升该特性和执行测试[Bath14]。其中一些测试，比如易用性测试，可能需要分配专门的人力资源、大量的计划、专用的实验室、特定的工具、特定的测试技能，并且在大多数情况下，还需要大量的时间。在某些情况下，易用性测试可能由一个独立的易用性专家组或用户体验专家组执行。

虽然测试分析师可能不用负责测试那些需要运用更多技术方法才能进行测试的质量特性，但是了解其他的质量特性并且理解他们和测试分析师所关注的质量特性在测试中的重叠部分仍然是十分重要的。例如在性能测试中失败的测试对象，如果其反应速度太慢而使用户无法有效使用，那么该测试对象也有可能在易用性测试中失败。类似的，与某些组件有互操作性问题的测试对象很可能也没有准备好做可移植性测试，因为当环境变化后可能会掩盖一些更基本的问题。

4.2业务领域测试的质量特性

功能性测试是测试分析师主要的关注点，主要关注测试对象做“什么”。功能性测试的测试依据通常是需求、规格说明、特定领域的专业知识或者隐含需求。功能性测试可能会受到软件开发生命周期的影响，也会根据不同的测试级别有所变化。例如，在集成测试期间进行的功能性测试，将会在定义实现了单一功能的接口组件上进行。而在系统测试级别，功能性测试则包括测试整个系统的功能性。在综合系统中，功能性测试主要集中于跨集成系统的端到端测试。在功能性测试中，采用了多种测试技术（参见第3章）。

在敏捷软件开发中，功能性测试通常包括：

- 测试在特定迭代中计划实现的具体功能（如：用户故事）
- 对所有未更改的功能进行回归测试

除了本章节覆盖到的功能性测试外，还有一些属于测试分析师职责范围的质量特性被视为非功能（专注于测试对象“如何”交付功能）的测试范畴。

功能性测试的三个子特性

4.2.1功能正确性测试

功能正确性测试包括验证应用程序是否符合规定的或隐含的需求，还可能包括计算准确性。功能正确性测试采用了第3章中介绍的多种测试技术，并且通常使用规格说明或原有系统作为测试结果参照物（test oracle）。功能正确性测试可以在任何测试级别上进行，主要针对数据或场景的不正确处理。

4.2.2功能适合性测试

功能适合性测试包括评估和确认，一组功能对其预期的特定任务的适合性。这种测试可以基于功能设计（例如：用例和/或用户故事）。功能适合性测试通常在系统测试期间进行，但也可能在集成测试后期阶段进行。在功能适合性测试中发现缺陷，则表明系统将无法以用户可接受的方式来满足用户需求。

4.2.3功能完备性测试

功能完备性测试是为了确定所实现的功能是否覆盖了特定的任务和用户目标。规格说明里的条目（如需求、用户故事、用例）和已实现的功能（如功能、组件、工作流）之间的可追溯性对于确定所需的功能完备性至关重要。度量功能完备性的方法可根据特定的测试级别和/或使用的不同的软件生命周期。例如，敏捷软件开发的功能完备性可能基于已实现的用户故事和功能。系统集成测试的功能完备性可能

集中在对高级别业务过程的覆盖上。

如果测试分析师维护测试用例和功能规格说明项之间的可追溯性，测试管理工具通常支持确定功能完备性。

功能完备性低于预期水平，表明该系统尚未完全实现。

4.2.4互操作性测试

互操作性测试验证了两个或多个系统或组件之间的信息交换。测试的重点是交换信息以及后续使用交换信息的能力。测试应覆盖所有预期的目标环境（包括不同的硬件、软件、中间件、操作系统等），以保证数据交换能够正常工作。在实际情况中，可能只在相对较少的环境中可行。在这种情况下，可能互操作性测试仅能在具代表性的一组环境中进行。详述互操作性测试需要对预期目标环境的组合进行识别、配置，并可供测试团队使用。选择可以使用环境中各种数据交换点的功能性测试用例来对这些环境进行测试。

互操作性与不同的组件和软件系统如何相互交互有关。具有良好互操作性特性的软件可以与其它许多系统集成，而无需进行重大变更或不会对非功能性行为产生重大影响。变更的数量以及需要实现和测试这些变更所需的工作量可以作为对互操作性的度量。

例如，软件互操作性测试可能集中于以下设计特性：

- 使用行业范围的通讯标准，如XML
- 具有自动检测与之交互系统的通讯需求，并作出相应调整的能力

互操作性测试对以下情况可能特别重要：

- 商业现货软件产品和工具
- 基于综合系统（system of systems）的应用
- 基于物联网的系统
- 与其他系统连接的网络服务（web services）

这类测试在组件集成和系统集成测试阶段执行。在系统集成级别，这类测试用来确定开发完毕的系统与其它系统的交互效果如何。因为系统可能在多个级别上进行交互，测试分析师必须理解这些交互并能够创造条件来执行各种交互。例如，如果两个系统将交换数据，测试分析师必须能够生成必要的数据以及执行数据交换所需要的事务。重要的是要记住，在需求文档中可能不会清晰的描述所有的交互。相反，许多交互只会在系统架构和设计文档中被定义。测试分析师必须具备检查这些文档的能力并做好相应准备，找出系统间和系统与它的环境间的信息交换点，并确保都被测试到。在互操作性测试中，等价类划分、边界值分析、判定表、状态转换图、用例和结对测试等技术都适用。能发现的典型缺陷包括交互组件之间不正确的数据交换。

4.2.5易用性评估

测试分析师通常负责协调和支持易用性评估。这可能包括详述易用性测试，或者作为主持人与用户一起进行测试。为了有效地完成这项工作，测试分析师必须了解这类测试所涉及的主要方面、目标和方法。请参考ISTQB®易用性测试专业教学大纲[ISTQB_UT_SYL]，了解本节所提供内容以外的细节。

理解为什么用户可能会难以使用系统或没有良好的用户体验（UX）是很重要的（例如，使用软件进行娱乐活动）。要理解这一点，首先必须要意识到“用户”可以指各种不同类型的角色，可以从IT专家、儿童，再到残疾人。

4.2.5.1易用性方面

以下是本节考虑的三个方面的内容：

- 易用性
- 用户体验(UX)
- 易访问性

易用性

易用性测试的目标是发现影响用户通过用户界面执行任务的能力的软件缺陷。这些缺陷可能会影响到用户有效、高效或满意地实现其目标的能力。易用性问题可能会导致用户困惑、错误、延迟或完全无法完成某些任务。

以下是易用性的子特性[ISO 25010 / GB/T 25000.10-2016]；相关定义，请参见[ISTQB_GLOSSARY]：

- 可辨识性（即可理解性）
- 易学性
- 易操作性
- 用户界面舒适性（即吸引力）
- 用户差错防御性
- 易访问性（见下文）

用户体验（UX）

用户体验评估涉及到对测试对象的完整用户体验，不仅仅是直接的交互。这对于测试对象来说尤为重要，因为在测试对象中，愉悦程度和用户满意度等因素对业务的成功至关重要。

影响用户体验的典型因素包括以下几点：

- 品牌形象(即用户对厂家的信任度)
- 互动行为
- 测试对象的帮助性，包括帮助系统、支持和培训等

易访问性

为包括残疾人在内，有特殊使用需求或限制的用户考虑软件的易访问性是十分重要的。易访问性测试应当考虑相关的标准，例如《Web内容无障碍指南》（WCAG）和立法，如《残疾歧视法》（北爱尔兰、澳大利亚）、《2010年平等法》（英格兰、苏格兰、威尔士）和508条款（美国）。易访问性同易用性一样，必须在设计阶段就被考虑。易访问性测试经常开始于集成测试级别，贯穿整个系统测试级别直至验收测试级别。通常在软件无法满足指定规则、标准和法律时，则视为存在缺陷。

改善易访问性的典型措施主要是为残疾用户提供与应用程序互动的机会。这些措施包括以下几点：

- 语音识别输入
- 确保呈现给用户的非文本内容有一个与文本等效的替代方案
- 允许在不丢失内容或功能的情况下调整文本的大小

易访问性指南为测试分析师提供了可用于测试的信息来源和检查表（易访问性指南的例子见[ISTQB_UT_SYL]）。此外，还提供了一些工具和浏览器插件来帮助测试人员识别易访问性问题，例如网页中的颜色选择不当导致违反了色盲指南。

4.2.5.2易用性评估方法

易用性、用户体验和易访问性可通过以下一种或多种方法进行测试：

- 易用性测试
- 易用性评审
- 易用性调查问卷

易用性测试

易用性测试是评估用户在特定环境中使用或学习使用系统以达成指定目标的难易程度。易用性测试的目的是为了度量以下几点：

- 有效性 – 指测试对象在特定使用环境中准确和完备地实现指定目标的能力
- 效率 – 指测试对象在特定使用环境中，消耗一定的资源获得有效性的能力
- 满意度 – 指测试对象在指定的使用环境中让用户满意的能力

值得注意的是，设计和确定易用性测试通常是由测试分析师与具有易用性测试特长的测试人员，以及理解以人为本的设计过程的易用性设计工程师合作进行的（详见[ISTQB_UT_SYL]）。

易用性评审

审查和评审是一种从易用性角度进行的测试，有助于提高用户的参与度。在软件开发生命周期早期发现需求规格说明和设计中的易用性问题是非常有价值的。启发式评估（对用户界面的易用性设计进行系统化审查）可以用来发现设计中的易用性问题，因此它可以作为迭代设计过程的一部分来处理。这包

括让一小部分评估者检查界面并判断其是否符合公认的易用性原则（"启发式"）。用户界面的可视化程度越高，评审越有效。例如，通过屏幕快照展现的样例通常比仅通过对特定屏幕的功能描述更容易理解和诠释。可视化对于对文档进行充分的易用性评审很重要。

易用性调查和调查问卷

调查和调查问卷技术可用于收集有关对系统用户行为的观察和反馈数据。标准化和公开可用的调查，诸如SUMI（软件易用性度量清单）和WAMMI（网站分析和度量清单），允许根据以往的易用性度量数据进行基准分析。此外，由于SUMI提供了具体的易用性测量标准，因此可以提供一组完成/验收的标准。

4.2.6可移植性测试

可移植性测试与软件组件或系统作为一个新的安装或从现有环境迁移到预定境中的难易程度有关。

ISO 25010产品质量特性分类包括以下几个可移植性相关的子特性：

- 易安装性
- 适应性
- 易替换性

测试分析师和技术测试分析师共同承担识别风险和设计可移植性特性测试的任务（见[ISTQB_ALTTA_SYL]第4.7节）。

4.2.6.1易安装性测试

对软件进行易安装性测试，需要使用文档化的规程在目标环境中进行安装和卸载。

测试分析师关注的典型测试目标包括：

- 验证软件的不同配置是否可以成功安装。在可能配置大量参数的情况下，测试分析师可以使用结对技术设计测试，以减少测试的参数组合数量，并将重点放在特定配置上（例如，经常使用的配置）。
- 测试安装和卸载步骤的功能正确性。
- 在安装或卸载后进行功能性测试，以检测可能引入的任何缺陷（如配置不正确、功能不可用等）。
- 识别安装和卸载步骤中的易用性问题（例如，确认在执行操作步骤时向用户提供了可理解的说明和反馈/错误信息）

4.2.6.2适应性测试

适应性测试检查给定的应用程序是否能够有效且高效地在所有预期的目标环境（硬件、软件、中间件、操作系统、云等）中正常运行。测试分析师通过识别预期的目标环境（例如，支持的不同移动操作系统的版本，可能使用的浏览器的不同版本），并设计覆盖这些环境组合的测试来支持适应性测试。然后使用选定的功能性测试用例对这些目标环境进行测试，这些功能性测试用例使用了存在于环境中的各

种组件。

4.6.2.3易替换性测试

易替换性测试侧重于系统内的软件组件或版本与其他组件或版本进行交换的能力。这对于基于物联网的系统架构尤其重要，其中不同硬件设备和/或软件安装的交换是经常发生的。例如，仓库中用于登记和控制库存水平的硬件设备可以被更先进的硬件设备(例如，使用更好的扫描器)取代，或者安装的软件可以被升级为新的版本，以使库存后备订单能够自动发布到供应商的系统中。

如果有多个可替代组件可供集成到完整的系统中，易替换性测试可由测试分析师在功能集成测试级别并行执行。

5.评审–120分钟

关键字

基于检查表的评审（checklist-based reviewing）。

评审的学习目标

5.1简介

无学习目标。

5.2在评审中使用检查表

TA-5.2.1（K3）根据教学大纲中提供的检查表信息识别需求规格说明中的问题。

TA-5.2.2（K3）根据教学大纲中提供的检查表信息识别用户故事中的问题。

5.1简介

测试分析师必须是评审过程的积极参与者，并提出自己独特的见解。如果恰当地进行了评审工作，评审有可能是对整体交付质量贡献最大、最划算的因素。

5.2在评审中使用检查表

基于检查表的评审是测试分析师在评审测试依据时最常用的技术。检查表可用来提醒参与者在评审过程中具体的检查点。它们还可以帮助消除评审的个性化（例如，“这是我们每次评审都会用到的检查表。并不是仅针对你的工作产品。”）

基于检查表的评审一般可以应用于所有评审，也可以针对特定的质量特性、领域或文档类型。例如，通用检查表可以核查文档的通用属性，如具有唯一的标识符、没有标记为“待确定”的参考资料、正确的格式和类似一致性的项。需求文档的专用检查表甚至可能包括对“shall（必须）”和“should（应该）”这类词汇使用恰当与否的检查，检查每项需求的可测试性等等。

可以通过需求的格式来确定要使用的检查表类型。采用描述性文本格式的需求文档与基于图表的需求文档将会使用不同的评审标准。

检查表也可以面向某一特定方面，如：

- 程序员/架构师技能集合或测试员技能集合-对于测试分析师来说，测试工程师技能集合检查表将是最合适的
- 一定的风险级别（例如，在安全关键系统中）- 检查表通常包括风险级别所需的具体信息
- 特定的测试技术 - 检查表将重点关注特定技术所需的信息（例如，在判定表中要表示的规则）
- 特定规格说明中的项，如需求、用例或用户故事 - 这些将在下面的章节中讨论，通常与技术测试分析师用于评审代码或架构的关注点不同。

5.2.1需求评审

下面的示例展示了面向需求的检查表可能包含的检查项：

- 需求的来源（例如，人、部门）
- 每个需求的可测试性
- 每个需求的优先级
- 每个需求的验收标准
- 适用时，用例调用结构的可用性
- 每个需求/用例/用户故事的唯一标识
- 每个需求/用例/用户故事的版本控制
- 从业务/市场需求到每个需求的可追溯性
- 需求和/或用例之间的可追溯性（适用时）

- 术语使用的一致性（例如，使用术语表）

重要的是要记住，如果一条需求是不可测试的，或者说测试分析师根据需求的定义无法确定如何对其进行测试，那么这条需求就存在缺陷。例如，一条需求的描述为“软件应该是对用户非常友好的”这是无法测试的。测试分析师如何判断软件是否对用户友好，甚至是非常友好？相反，如果需求中写着“软件必须符合易用性标准文档xxx版中规定的易用性标准”，如果易用性标准文档存在，那么这个需求具备可测试性。而这条需求又是一个全局需求，因为该需求适用于界面中的所有元素。在这种情况下，这条需求在一个重要的应用上就可以很轻易的派生出大量独立的测试用例。建立从这个需求，或者从易用性标准文档到测试用例之间的追溯关系同样非常重要，因为一旦关联的易用性标准发生了变动，测试用例可能需要随之被重新评审或更新。

如果测试人员无法确定测试是否通过，或者无法构建出能够执行通过或者失败的测试，那与该测试有关的需求同样也不具备可测试性。例如，“系统应该全年365（或366）天，每周7天，每天24小时，均100%可用”就是不可测试的。

针对用例评审的简化检查表可能包括以下问题：

- 是否清晰的定义了基本流（路径）？
- 是否识别了所有的备选流（路径），并完成了错误处理？
- 是否定义了用户界面信息？
- 是否只有一个基本流（应该只有一个，否则应拆分为多个用例）？
- 每条路径是否均可测试？

5.2.2用户故事评审

在敏捷软件开发中，需求通常采用用户故事的形式表述。这些故事可作为验证功能的小单元。与用例可以是跨越多个功能部分的事务不同，用户故事更加独立并且通常根据开发时间来限定范围。用于用户故事的检查表[1]可能包含：

- 故事是否适合迭代/冲刺的目标？
- 故事是从需求提出者的角度编写的吗？
- 验收标准是否已定义并可测试？
- 特征是否被清晰定义并可区分？
- 这个故事是否独立于其他故事？
- 故事是否设定了优先级？
- 故事是否遵循常用格式：作为一个<用户类型>，我希望<一些目标>，以便<一些原因>[Cohn04]

如果故事定义了新的界面，那么除了使用通用的故事检查表（例如上面的这个）以外，还应当使用一个详细的用户界面检查表。

5.2.3 裁剪检查表

可根据以下情况裁剪一份检查表：

- 组织（例如，参考公司政策、标准、惯例、法律限制）
- 项目/开发工作(例如，关注点、技术标准、风险)
- 被评审的工作产品的类型（例如，代码评审可能需对特定的编程语言进行裁剪）
- 被评审的工作产品的风险级别
- 使用的测试技术

好的检查表会发现问题，还有助于就检查表中可能没有被特别提及的项展开讨论。组合运用不同的检查表是确保通过评审成功获得最高质量工作产品的重要途径。将基于检查表的评审与标准检查表（例如基础级教学大纲中引用的检查表）结合使用，并编写组织特有的检查表（如上述所示的检查表）将有助于测试分析师进行有效的评审。

更多关于评审和审查的信息可以参考 [Gilb93] 和 [Wiegers03]。更多检查表的例子可以从第7.4节的参考资料中获得。

6.测试工具和自动化–90分钟

关键字

关键字驱动测试（keyword-driven testing）、测试数据准备（test data preparation）、测试设计（test design）、测试执行（test execution）、测试脚本（test script）。

测试工具和自动化的学习目标

6.1简介

无学习目标

6.2关键字驱动自动化

TA-6.2.1(K3)对于给定的场景，确定测试分析师在关键字驱动的测试项目中的适当活动。

6.3测试工具的类型

TA-6.3.1(K2)解释在测试设计、测试数据准备和测试执行中应用的测试工具的用途和类型。

6.1简介

测试工具可以极大提高测试的效率和准确性。本章将介绍测试分析师所使用的测试工具和自动化方法。需要注意的是，测试分析师与开发人员、测试自动化工程师和技术测试分析师一起工作，共同创建测试自动化解决方案。关键字驱动的自动化尤其涉及到测试分析师，以便利用他们在业务和系统功能性方面的经验。

关于测试自动化的话题和测试自动化工程师角色的更多信息，请参见ISTQB®高级测试自动化工程师教学大纲[ISTQB_TAE_SYL]。

6.2关键字驱动测试

关键字驱动测试是主要的测试自动化方法之一，测试分析师需要提供主要的输入：关键字和数据。

关键字（有时也称为行为词）经常（但不局限于）用于表示系统的高级别业务交互（例如“取消订单”）。每个关键字通常用来表示操作者和被测系统之间大量详细的交互。关键字序列（包括相关的测试数据）用于阐述测试用例。[Buwalda02]

在测试自动化中，关键字被实现为一个或多个可执行的测试脚本。工具读入用关键字序列编写的测试用例，调用相应的测试脚本来实现关键字功能。这些脚本是以高度模块化的方式实现的，能够更容易映射到特定关键字。当然，实现这些模块化脚本是需要一定的编程技能的。

以下是关键字驱动测试的主要优点：

- 与特定应用或业务领域关联的关键字可由领域专家定义。可以使得编制测试用例规格说明的任务更有效。
- 具备主要领域专业知识的人员也能从自动化的测试用例执行（一旦关键字被实现为脚本）中获益，而无需了解底层的自动化代码。
- 当功能和被测软件的接口发生变化时，使用模块化编写技术使测试自动化工程师可以有效地维护测试用例[Bath14]。
- 测试用例规格说明独立于它们的实现。

测试分析师通常会创建和维护关键字/行为词数据。他们必须认识到，脚本开发的任务对于实现关键字仍然是必需的。一旦定义了要使用的关键字和数据，测试自动化人员（例如，技术测试分析师或测试自动化工程师）就会把业务流程关键字和低级别动作翻译成自动化测试脚本。

虽然关键字驱动测试通常在系统测试阶段进行，但其代码开发可能早在测试设计之初就开始了。在迭代环境中，特别是使用持续集成/持续部署时，测试自动化开发是一个持续的过程。

一旦创建了输入关键字和数据，测试分析师通常负责执行关键字驱动测试用例，并且分析任何可能发

生的失效。

当发现异常时，测试分析师应该协助调查失效的原因，以确定缺陷是与关键字、输入数据、测试自动化脚本本身，还是与被测试的应用有关。通常失效排除步骤的第一步是人工使用相同的数据执行相同的测试，观察失效是否发生在应用本身。如果这步没有显示失效，测试分析师应检查导致失效的测试序列，以确定问题是否发生在前一步（可能引入了不正确的输入数据导致），但缺陷直到后面的处理过程才显露。如果测试分析师还是不能确定失效的原因，则应将故障排除信息传递给技术测试分析师或开发人员做进一步分析。

6.3测试工具的类型

测试分析师的大部分工作都需要有效地使用工具。以下几点可以增强工具使用的有效性：

- 了解应该使用哪些工具
- 了解工具可以提高测试工作的效率（例如，在允许的时间内提供更好的覆盖率）。

6.3.1测试设计工具

测试设计工具用来帮助生成测试所需的测试用例和测试数据。这些工具可以针对特定格式的需求文档、模型（例如UML）或测试分析师提供的输入进行处理，测试设计工具通常被设计和构造为可以与特定格式和特定工具协同工作，例如特定的需求管理工具。

测试设计工具能为测试分析师提供信息，以便确定为达到特定目标级别的覆盖率、对系统的信心或产品风险缓解活动所需使用的测试类型。例如，分类树工具基于选定的覆盖准则能产生（并且显示）达到完全覆盖所需的一组组合，测试分析师可以利用这些信息确定必须执行的测试用例。

6.3.2测试数据准备工具

测试数据准备工具提供以下几方面优势：

- 分析诸如需求文档甚至源代码之类的文档，以确定测试过程中达到一定覆盖率级别所需要的数据。
- 从生产系统中提取一个数据集，并对其进行“清洗”或匿名化，以删除任何个人信息，同时仍然保持数据的内部完整性。然后，经过清洗的数据可以用于测试，而不会有安全泄露或滥用个人信息的风险。这在需要大量真实数据，且存在安全和数据隐私风险的情况下尤为重要。
- 从给定的输入参数集生成合成测试数据（例如，用于随机测试）。其中一些工具将分析数据库结构，以确定需要测试分析师提供哪些输入。

6.3.3自动化测试执行工具

在所有测试级别中，测试分析师主要使用测试执行工具运行自动化测试和检查实际结果。使用测试执行工具的目的有以下几种：

- 为了降低成本（在工作和/或时间方面）
- 为了运行更多的测试

- 为了在多种环境中运行相同的测试
- 为了使测试执行更具可重复性
- 为了运行那些靠人工无法运行的测试（例如，大规模数据确认测试）

这些目标通常与提高覆盖率的同时降低成本的主目标重叠。

在自动化回归测试中，测试执行工具的投资回报通常是最高的，主要是因为回归测试只需要低级别维护并且被重复执行。自动化冒烟测试中也可以有效地使用自动化，因为会频繁的使用冒烟测试并且需要快速地获得测试结果，尽管维护成本可能更高，但能通过自动化途径来评估持续集成环境中的新构建。

测试执行工具通常用于系统和集成测试级别中。有些工具也可能用于组件测试，特别是API测试工具。凭借最适用的工具将有助于提高投资回报率。

7.参考文献

7.1 标准

[ISO25010] ISO/IEC 25010 (2011)系统与软件工程-系统与软件质量要求和评价（SQuaRE）系统与软件质量模型第4章。（中文版 GB/T 25000.10-2016）

[ISO29119-4] ISO/IEC/IEEE 29119-4 《软件和系统工程-软件测试-第4部分，测试技术》，2015年。

[OMG-DMN]对象管理组织：OMG®决策模型和标记™， 1.3版，2019年12月；网址：www.omg.org/spec/DMN/，第8章

[OMG-DMN]对象管理组织：OMG®统一建模语言TM， 2.5.1版，2017年12月；网址：url: www.omg.org/spec/UML/

[RTCA DO-178C/ED-12C]：机载系统和设备审定中的软件要求， RTCA/EUROCAE ED12C，2013，第1章。

7.2 ISTQB®和IREB 文档

[IREB_CPRE] IREB Certified Professional for Requirements Engineering Foundation Level Syllabus， Version 2.2.2， 2017

国际需求工程委员会 需求工程基础级专业认证教学大纲 版本2.2.2，2017

[ISTQB_AL_OVIEW] ISTQB® Advanced Level Overview， Version2.0

高级教学大纲-概述，版本2.0

[ISTQB_ALTTA_SYL] ISTQB® Advanced Level Technical Test Analyst Syllabus，Version2019

ISTQB® 高级技术测试分析师教学大纲 版本2019

[ISTQB_FL_SYL] ISTQB® Foundation Level Syllabus， Version 2018

ISTQB®基础级教学大纲 版本2018

[ISTQB_GLOSSARY] Standard glossary of terms used in Software Testing软件测试中使用的标准术语表

url: https://glossary.istqb.org/

[ISTQB_TAE_SYL] ISTQB® Advanced Level Test Automation Engineer Syllabus, Version 2017

ISTQB®高级测试自动化工程师教学大纲，版本2017

[ISTQB_UT_SYL] ISTQB® Foundation Level Specialist Syllabus Usability Testing, Version 2018

ISTQB®基础级专家教学大纲易用性测试，版本2018

7.3 文献和文章

[Bath14] Graham Bath, Judy McKay, “The Software Test Engineer’s Handbook (2nd Edition)”, Rocky Nook, 2014, ISBN 978-1-933952-24-6

[Beizer95] Boris Beizer, “Black-box Testing”, John Wiley & Sons, 1995, ISBN 0-471-12094-4 [Black02]: RexBlack, “ManagingtheTestingProcess(2nd edition)”, JohnWiley&Sons: NewYork, 2002, ISBN 0-471-22398-0

[Black07]: Rex Black, “Pragmatic software testing: Becoming an effective and efficient test professional”, John Wiley and Sons, 2007, ISBN 978-0-470-12790-2

[Black09]: Rex Black, “Advanced Software Testing, Volume 1”, Rocky Nook, 2009, ISBN 978-1-933- 952-19-2

高级软件测试第1卷，清华大学出版社，2011

[Buwalda02]: Hans Buwalda, “Integrated Test Design and Automation: Using the Test Frame Method”, Addison-Wesley Longman, 2002, ISBN0-201-73725-6

[Chow1978]: T.S. Chow, Testing Software Design Modeled by Finite-State Machines, IEEE Transactions on Software Engineering vol. SE-4, issue 3, May 1978, pp. 178-187

[Cohn04]: Mike Cohn, “User Stories Applied: For Agile Software Development”, Addison-Wesley Professional, 2004, ISBN 0-321-20568-5

用户故事与敏捷方法，清华大学出版社，2010

[Copeland04]: Lee Copeland, “A Practitioner’s Guide to Software Test Design”, Artech House, 2004, ISBN 1-58053-791-X

[Craig02]: Rick David Craig, Stefan P. Jaskiel, “Systematic Software Testing”, Artech House, 2002, ISBN 1-580-53508-9 /系统的软件测试，电子工业出版社，2003

[Forgács19]: István Forgács, Attila Kovács, “Practical Test Design”, BCS, 2019, ISBN 978-1-780-1747-23

[Gilb93]: Tom Gilb, Dorothy Graham, “Software Inspection”, Addison-Wesley, 1993, ISBN 0-201-63181-4

[Koomen06]： Tim Koomen， Leo van der Aalst， Bart Broekman， Michiel Vroon ″TMap NEXT，for result driven testing″， UTN Publishers， 2006， ISBN 90-72194-80-2

[Kuhn16]： D. Richard Kuhn et al， “Introduction to Combinatorial Testing， CRC Press，2016，ISBN 978-0-429-18515-1

[Myers11]： Glenford J. Myers， “The Art of Software Testing 3rd Edition”， John Wiley & Sons， 2011，ISBN： 978-1-118-03196-4

软件测试的艺术（原书第3版）， 机械工业出版社， 2012

[Offutt16]： Jeff Offutt， Paul Ammann， Introduction to Software Testing - 2nd Edition，Cambridge University Press， 2016， ISBN 13： 9781107172012，

软件测试基础（原书第2版），机械工业出版社，2018

[vanVeenendaal12]： Erik van Veenendaal， “Practical risk-based testing.″ Product Risk Management：The PRISMA Method ”， UTN Publishers，2012， ISBN 9789490986070

[Wiegers03]： Karl Wiegers， “Software Requirements 2”， Microsoft Press， 2003， ISBN 0-735-61879-8

[Whittaker03]： James Whittaker， “How to Break Software”， Addison-Wesley， 2003，ISBN 0-201-79619-8

[Whittaker09]： James Whittaker， “Exploratory software testing： tips， tricks， tours，and techniques to guide test design”， Addison-Wesley， 2009， ISBN 0-321-63641-4 /探索式软件测试，清华大学出版社，2010

7.4 其它参考文献

下列参考文献是因特网和其它地方的信息。尽管在发布本高级教学大纲时核查过这些参考文献，但如果现在已经无法获取这些信息，ISTQB®不承担任何责任。

- 第3章

  Czerwonka， Jacek：www.pairwise.org

  Defect taxonomy：www.testingeducation.org/a/bsct2.pdf

Sample defect taxonomy based on Boris Beizer’ s work:
inet.uni2.dk/~vinter/bugtaxst.doc

Good overview of various taxonomies: testingeducation.org/a/bugtax.pdf

Heuristic Risk-Based Testing By James Bach

Exploring Exploratory Testing, Cem Kaner and Andy Tinkham,
www.kaner.com/pdfs/ExploringExploratoryTesting.pdf

Pettichord, Bret, “An Exploratory Testing Workshop Report”,
www.testingcraft.com/exploratorypettichord

- 第5章

http://www.tmap.net/checklists-and-templates

8.附录A

下表来自于ISO 25010（中文版 GB/T 25000.10-2016）中提供的完整表格。它仅侧重于测试分析师教学大纲所涵盖的质量特性，并将ISO 9126(2012年版教学大纲中使用的)和ISO 25010(本版中使用的)中的术语进行了比较。

ISO/IEC 25010	ISO/IEC 9126-1	备注
功能性	**功能性**	
功能完备性		
功能正确性	准确性	
功能适合性	适用性	
	互操作性	该特性已移到兼容性
易用性		
可辩识性	易理解性	新的名字更加准确
易学性	易学性	
易操作性	易操作性	
用户差错防御性		新增的子特性
用户界面舒适性	吸引性	新的名字更加准确
易访问性		新增的子特性
兼容性		新增的特性
互操作性		
共存性		技术测试分析师涵盖的内容

附录-致谢

中文版本	发布日期	本地化专家（按姓氏评审排序）
V3.1.0	2021年7月	贺炘、郑丹丹
2019 V3.0	2019年10月19日	商莉，陶显锋，许爱国，羊婷婷，杨婷，于长青，翟宏宝，郑丹丹，郑文强（组长） 周震漪（终审）
2012 V2.0	2012年10月19日	柴阿峰、杜庆峰、马均飞、沈建雄、熊晓虹、徐文叶、郑文强、周震漪（组长）