文档编号：SY00010_ISTQB_CTAL_TA_EN4.0_CN1.0

认证测试工程师
高级测试分析师
大纲

版本：EN4.0_CN1.0

发布日期：2025年11月28日

国际软件测试认证委员会

中文版的翻译、编辑和出版统一由ISTQB®授权的CSTQB®负责

版权声明

英文版权声明

2025年版的v4.0教学大纲的作者：Armin Born, Filipe Carlos, Wim Decoutere, István Forgács, Matthias Hamburg (Product Owner), Attila Kovács, Sandy Liu, François Martin, Stuart Reid, Adam Roman, Jan Sabak, Murian Song, Tanja Tremmel, Marc-Florian Wendland, Tao Xian Feng.

2021-2022年版的 v3.1.0 修正版3.1.1和修正版3.1.2 教学大纲的作者（修改者）：Wim Decoutere, István Forgács, Matthias Hamburg, Adam Roman, Jan Sabak, Marc-Florian Wendland.

2019年版教学大纲的作者（修改者）：Graham Bath, Judy McKay, Jan Sabak, Erik van Veenendaal.

2012年版教学大纲的作者：Judy McKay, Mike Smith, Erik van Veenendaal.

中文版权声明

修订历史

版本	日期	说明
EN4.0_CN1.0	2025/11/28	CTAL-TA v4.0中文版常规发布版本
v4.0	2025/05/02	全面修订与范围调整
v3.1.2	2022/01/31	修正格式、语法及措辞细节问题
v3.1.1	2021/05/15	版权声明已适配现行ISTQB®标准
v3.1	2021/03/03	重写第3.2.3节并优化多处措辞
v3.0 (2019)	2019/10/19	全面修订并缩减范围
v2.0 (2012)	2012/10/19	首次以独立《认证测试分析师大纲》形式发布

目录

致谢

ISTQB委员会于2025年5月2日正式发布本文档。

本文档由国际软件测试资格认证委员会的工作组编制完成，工作组成员包括：Armin Born、Filipe Carlos、Wim Decoutere、István Forgács、Matthias Hamburg（产品负责人）、Attila Kovács、Sandy Liu、François Martin、Stuart Reid、Adam Roman、Jan Sabak、Murian Song、Tanja Tremmel、Marc-Florian Wendland、Tao Xian Feng。

工作组感谢Julia Sabatine的校对，Gary Mogyorodi的技术评审，Daniel Pol'an持续的技术支持，以及评审团队和各成员委员会提出的建议和意见。

以下人员参与了本大纲的评审、评论和投票：

Current edition (v4.0): 当前版本（v4.0）：Gergely Ágnecz, Laura Albert, Dani Almog, Somying Atiporntham, Andre Baumann, Lars Bjørstrup, Ralf Bongard, Earl Burba, Filipe Carlos, Renzo Cerquozzi, Kanitta Chantaramanon, Alessandro Collino, Nicola De Rosa, Aneta Derkova, Dingguofu, Ole Chr. Hansen, Zheng Dandan,Karol Frühauf, Dinu Gamage, Chen Geng, Sabine Gschwandtner, Ole Chr. Hansen, Zsolt Hargitai, Tharushi Hettiarachchi, Ágota Horváth, Arnika Hryszko, Caroline Julien, Beata Karpinska, Ramit Manohar Kaul, Mattijs Kemmink, László Kvintovics, Thomas Letzkus, Marek Majerník, Donald Marcotte, Dénes Medzihradszky, Imre Mészáros, Krisztián Miskó, Gary Mogyorodi, Ebbe Munk, Maysinee Nakmanee, Ingvar Nordström, Tal Pe'er, Kunlanan Peetijade, Sahani Pinidiya, Lukáš Piška, Nishan Portoyan, Meile Posthuma, Yasassri Ratnayake, Randall Rice, Adam Roman, Marc Rutz, Jan Sabak, Vimukthi Saranga, Sumuduni Sasikala, Gil Shekel, Radoslaw Smilgin, Péter Sótér , Helder Sousa, Richard Taylor, Benjamin Timmermans, Giancarlo Tomasig, Robert Treffny, Shun Tsunoda, Stephanie Ulrich, François Vaillancourt, Linda Vreeswijk, Carsten Weise, Marc-Florian Wendland, Paul Weymouth, Elżbieta Wiśniewska, John Young, Claude Zhang.

Edition 2021-2022 (v3.1): 2021-2022版（v3.1）：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, Ingvar Nordström, 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.

Subsequently, Tal Pe'er, Stuart Reid, Marc-Florian Wendland, and Matthias Hamburg suggested formal, grammar, and wording improvements that have been implemented and published in Errata 3.1.1 and 3.1.2.

Edition 2019 (v3.0): 2019版（v3.0）：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.

Edition 2012 (v2.0): 2012版（v2.0）：Graham Bath, Arne Becher, Rex Black, Piet de Roo, Frans Dijkman, Mats Grindal, Kobi Halperin, Bernard Homès, Maria Jönsson, 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.

ISTQB® CTAL-TA 高级测试分析师大纲4.0版本及附属文档，由ISTQB®中国成员国委员会CSTQB®组织专家团队统一进行了本地化工作，并于2025年11月28日正式发布。在此感谢参加此次ISTQB®高级测试分析师中文本地化工作组专家成员。参加本次中文翻译、评审参与者（按姓氏拼音排序）：白宇、曹翔、刘晓更、吴衍刚（组长）、许根、朱云娜， QA评审专家（按姓氏拼音排序）：郑文强、郑丹丹。

0. 简介

0.1 本大纲的目的

本大纲是国际软件测试资格认证高级测试分析师级别考试的基础。ISTQB® 提供此大纲的目的如下：

1. 供各成员委员会将其翻译成本地语言，并授权培训提供商。成员委员会可根据其特定的语言需求调整大纲内容，并可修改参考文献以适应本地的出版物。
2. 供认证机构根据本大纲的学习目标，推导出以其本地语言命制的考试题目。
3. 供培训提供商制作培训材料并确定合适的教学方法。
4. 供认证考生（无论是作为培训课程的一部分还是独立学习）为认证考试做准备。
5. 供国际软件和系统工程界用以推进软件和系统测试专业的发展，并作为书籍和文章的根据。

0.2 软件测试中的高级测试分析师

ISTQB®高级测试分析师认证旨在提供在整个软件开发生存周期中执行结构化且彻底的软件测试所需的技能。它详细说明了测试分析师在标准测试过程每一步中的角色和职责，并深入阐述了重要的测试技术。高级测试分析师认证面向已获得基础级别证书，并希望进一步提升其在测试分析和测试技术方面专业知识的人士。

在本大纲中，测试分析师被理解为一个角色，该角色：

- 更侧重于客户的业务需求，而非测试的技术层面。
- 主要执行功能测试，同时也参与以用户为中心的非功能测试，例如可用性、适应性、可安装性或互操作性测试。
- 使用黑盒测试技术和基于经验的测试，而非白盒测试技术。
- 通过使用缺陷预防技术来提高测试的有效性。

0.3 测试人员的职业路径

ISTQB®认证体系为测试专业人员在其职业生涯的各个阶段提供支持。获得 ISTQB® 高级测试分析师认

证的个人可能也会对其他核心高级级别（高级技术测试分析师和高级测试管理）感兴趣，并可在此后进阶专家级别（测试管理或改进测试过程）任何希望提升在敏捷环境下测试实践技能的人士，可以考虑“敏捷技术测试工程师”或“大规模敏捷测试领导力”认证。此外，专业方向提供专注于特定测试技术和方法、特定质量特性与测试级别，或特定行业领域内测试的认证产品。请访问www.istqb.org获取关于ISTQB®“认证测试人员”计划的最新信息。

0.4 商业价值

本节列出了获得高级测试分析师认证的候选人应具备的商业价值。

获得高级测试分析师认证的测试人员可以：

代码	描述
TA-BO1	按照所遵循的软件生存周期模型，支持并执行适当的测试活动
TA-BO2	应用基于风险的测试原则
TA-BO3	选择并应用适当的测试技术，以支持测试目标的实现
TA-BO4	提供详细程度适当且质量达标的文档
TA-BO5	确定需要执行的功能测试类型
TA-BO6	参与非功能测试
TA-BO7	帮助预防缺陷
TA-BO8	利用工具提高测试过程的效率
TA-BO9	明确测试环境和测试数据的要求

0.5 学习目标和知识认知级别

学习目标支撑商业价值，并用于生成认证测试分析师进阶级考试题。

一般来说，除引言和附录外，本大纲的所有内容都在考试范围内。考试题目将确认 K1 级别（见下文）的关键词知识或相应知识级别的学习目标。

每章开头都会给出具体的学习目标和认知级别，分类如下：

- K2：理解
- K3：应用

- K4：分析

对于章节标题下方列出的所有关键词，即使没有在任何学习目标中明确提及，也应牢记ISTQB®词汇表中的正确名称和定义（K1）。

有关学习目标的更多详情和示例，请参见第7节。

0.6 高级测试分析师认证考试

高级测试分析师认证考试将基于本大纲的内容。试题的答案，可能需要使用本大纲中一个甚至多个章节的知识点。除引言和附录外，本教学大纲的所有章节都在考试范围内。标准和书籍列为参考资料，但除了已在教学大纲中归纳的标准和书籍内容外，其他内容不在考试范围内。

更多详情，请参阅与高级测试分析师 v4.0 适配的考试结构和规则文件。

ISTQB®高级测试分析师认证的报考基本条件是考生对软件测试感兴趣。不过，强烈建议考生同时具备以下条件：

- 至少具备软件开发或软件测试方面的基本背景，如六个月的系统测试员或用户验收测试员，或软件开发人员工作经验
- 参加按照 ISTQB® 标准获得授权的课程（由 ISTQB 认可的分会之一授权）。

报考要求说明：在参加高级测试分析师认证考试之前，必须获得 ISTQB® 基础级证书。

0.7 授权

ISTQB®分会可对课程材料遵循本大纲的培训机构进行授权。培训机构应向分会或执行认可的机构索取授权指南。经授权的课程被视为符合本教学大纲，并允许在课程中包含 ISTQB® 考试。本大纲的授权指南遵循ISTQB®过程管理与合规工作组发布的通用授权指南。

0.8 标准处理

IEEE 和 ISO 等国际标准化组织已经发布了与质量特性和软件测试相关的标准。本大纲参考了这些标准。参考这些文献的目的是提供一个框架（如有关质量特性的 ISO/IEC 25010 参考文献），或在读者需要时提供额外信息的来源。请注意，ISTQB® 教学大纲将标准文件仅作为参考。标准文件并非用于考试。有关标准的更多信息，请参阅第 6 节——参考文献。

0.9 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。
- 列出了学员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。

教学大纲的内容并没有包含软件测试的整个知识领域，只是提供了高级测试分析师培训课程所涵盖的详细程度。

大纲使用ISTQB®术语表（ISTQB-Glossary，2024）中定义的软件测试和质量保证术语（即术语的名称和含义）。

有关相关学科的术语，请参阅各自的术语表：需求工程请参阅IREB-CPRE（IREB-Glossary，2024），软件工程领域请参阅IEEE-Pascal（Computer Society et al.，2024）。

0.10 本教学大纲的结构

本教学大纲由包含在考试范围内的五大章节组成。每一章的一级标题规定了本章的时间分配；一级标题以下不列出时间分配。对于经过认可的培训课程，本教学大纲要求至少20.25小时（1215 分钟）的授课时间。下面是五大章节的课时分布：

- 第 1 章（225 分钟）：测试分析师在测试过程中的任务
  - 学员将学习测试分析师如何参与各种软件开发生存周期。
  - 学员将学习测试分析师如何参与各种测试活动。
  - 学员将学习测试分析师执行的与工作产品相关的任务。
- 第 2 章（90 分钟）测试分析师在基于风险的测试中的任务
  - 学员将学习测试分析师如何参与产品风险分析。
  - 学员将学习如何分析变更的影响，以确定回归测试的范围。

- 第 3 章（615 分钟）测试分析和测试设计
  - 学员将学习基于数据的测试技术，如域测试、组合测试和随机测试。
  - 学员将学习基于行为的测试技术，如 CRUD 测试、状态转移测试和基于场景的测试。
  - 学员将学习基于规则的测试技术，如判定表测试和蜕变测试。
  - 学员将学习基于经验的测试，如基于会话的测试和众测。
  - 学员将学习如何选择适当的测试技术来缓解产品风险。
- 第 4 章（60 分钟）测试质量特性
  - 学员将学习如何执行多种类型的功能测试。
  - 学员将学习如何利用功能方面的具体知识来促进非功能测试类型，如易用性测试、灵活性测试和兼容性测试。
- 第 5 章（225 分钟）软件缺陷预防
  - 学员将学习各种缺陷预防实践。
  - 学员将学习支持阶段遏制的各种方法
  - 学员将学习如何减少缺陷再次发生

1. 测试分析师在测试过程中的任务 - 225 分钟

关键词

概要测试用例（high-level test case）、关键词（keyword）、关键词驱动测试（keyword-driven testing）、详细测试用例（low-level test case）、软件开发生存周期（software development lifecycle）、测试分析（test analysis）、测试分析师（test analyst）、测试用例（test case）、测试条件（test condition）、测试数据（test data）、测试设计（test design）、测试环境（test environment）、测试执行（test execution）、测试实施（test implementation）、测试结果参照物（test oracle）、测试脚本（test script）、测试件（testware）

第1章的学习目标：

1.1 软件开发生存周期中的测试

TA-1.1.1 (K2) 总结测试分析师在不同软件开发生存周期中的参与。

1.2 参与的测试活动

TA-1.2.1 (K2) 总结测试分析师在测试分析中的任务。

TA-1.2.2 (K2) 总结测试分析师在测试设计中的任务。

TA-1.2.3 (K2) 总结测试分析师在测试实施中的任务。

TA-1.2.4 (K2) 总结测试分析师在测试执行中的任务。

1.3 与工作产品相关的任务

TA-1.3.1 (K2) 区分概要测试用例与详细测试用例。

TA-1.3.2 (K2) 解释测试用例的质量准则。

TA-1.3.3 (K2) 举例说明测试环境需求。

TA-1.3.4 (K2) 解释测试结果参照物问题及其潜在解决方案。

TA-1.3.5 (K2) 举例说明测试数据需求。

TA-1.3.6 (K3) 使用关键词驱动测试开发测试脚本。

TA-1.3.7 (K2) 总结用于管理测试件的工具类型。

测试过程中测试分析师任务简介

基础级教学大纲（ISTQB-CTFL, v4.0.1）描述了测试中的两个主要角色：测试管理角色和测试角色。本大纲中，负责测试软件业务方面的测试角色称为测试分析师（TA）。虽然这一职责很少分配给专门的职位或角色，但测试分析师有明确定义的任务和所需能力。在测试级别方面，测试分析师专注于系统测试、验收测试和系统集成测试。在质量特性方面，测试分析师的能力集中在功能性上，同时涵盖面向用户的非功能质量特性，如易用性、适应性、易安装性和互操作性。

1.1 软件开发生存周期中的测试

1.1.1 测试分析师在不同软件开发生存周期中的参与

测试活动的组织方式取决于所采用的软件开发生存周期（SDLC）。因此，测试分析师（TA）在测试活动中的参与会根据所采用的SDLC模型而有所不同。

在**顺序开发模型**中，开发活动分阶段进行，每阶段在前一阶段完成后开始。通常，开发活动间几乎没有重叠。因此，测试分析师的任务会随时间变化。在软件开发生存周期的早期阶段，测试分析师主要支持测试规划。当生成了测试依据时，测试分析师开始进行测试分析。测试设计和测试实施与软件设计和实施并行进行。最后，在SDLC后期阶段，测试分析师执行测试并支持测试完成。

在**增量开发模型**中，把软件划分为较小的、可管理的增量。每个增量是独立开发和测试。因此，测试分析师为每个增量执行相同的活动（即测试分析、测试设计、测试实施、测试执行和测试管理支持）。然而，测试分析师的工作在每个增量中的组织方式可能不同。测试重点在于新增或修改的功能。此外，由于回归风险增加，测试分析师需特别关注代码重构和回归测试套件的构建。

在**迭代开发模型**中，开发过程是循环的。项目经历反复的原型设计、测试、优化和部署循环。测试分析师的角色具有动态性和适应性。测试分析师与开发者和业务代表密切协作，适应不断演变的软件产品。测试分析师根据软件的演变来调整和修改测试条件和测试用例，并在每次迭代中提供反馈以改进测试过程。迭代越频繁，测试分析师对回归测试的持续维护和开发就越重要。

SDLC可能结合多种模型的元素以及特定技术和方法（例如，敏捷软件开发结合了迭代和增量模型的特点）。在这种情况下，测试分析师的参与取决于SDLC的具体特性和组合方式。所有SDLC模型的共同最佳实践是，测试分析师应从SDLC的初始阶段开始参与。

1.2 参与的测试活动

基础级教学大纲（ISTQB-CTFL，v4.0.1）描述了测试过程中的七个活动。测试分析师（TA）主要关注其中四个活动：测试分析、测试设计、测试实施和测试执行。

以下章节详细描述了测试分析师在这四个活动中的任务。

1.2.1 测试分析

在测试分析期间，测试分析师检查测试依据的完整性，并收集与测试相关的额外信息。这不仅包括文档，还包括口头信息，例如在协作用户故事编写中的对话（参见ISTQB-CTFL v4.0.1，第 4.5.1 节）。测试依据的变化可能导致与测试管理协调调整测试范围。

为有效进行测试分析，测试分析师需检查以下入口准则：

- 已完成测试规划，已明确测试范围、测试目的和测试方法。
- 已定义测试依据（如需求或用户故事）。
- 已评估，并在需要时记录识别的产品风险。

测试分析师评估测试依据以识别其可能包含的缺陷并评价其可测试性，从而为产品负责人提供早期反馈。这可能包括根据应用的测试技术对系统行为进行建模（参见第 3 节、第 *5.2.1* 节）。作为过程的一部分，还会应用评审技术（参见第 *5.2.2* 节）。如果测试依据中的缺陷未直接修复，必须予以记录。此外，测试分析师确定所需的测试结果参照物（参见第 *1.3.4* 节）。

测试分析师为测试范围内的每个测试项定义测试条件并确定其优先级排序。测试条件需与测试目的一致（参见ISTQB-CTFL v4.0.1，第 1.1.1 节），并可追溯到测试依据的元素。测试条件的范围和重点需考虑产品风险。在增量或迭代开发模型中，这包括基于影响分析确定回归测试的范围。在敏捷软件开发中，测试条件可以表达为反映用户故事风险的验收准则。

测试分析师可分阶段开展工作，先从概要测试条件（如“屏幕X的功能”）入手。接下来，定义更详细的测试条件，如“屏幕X会拒绝少一位数字的账号”。这种方法支持足够的覆盖率，并使测试设计能够尽早开始，例如针对仍需优化的用户故事。

测试分析师需让利益相关方参与评审测试条件，确保测试依据得到清晰理解，且测试与测试目的保持一致。

1.2.2 测试设计

测试设计描述如何执行测试以实现既定的测试目的，通常通过测试用例完成。测试设计的执行方式取决于多种因素，包括所需覆盖率、测试依据、SDLC、项目约束以及测试人员的知识和经验。

在测试设计期间，测试分析师确定哪些领域适合使用详细测试用例或概要测试用例（参见第 *1.3.1* 节）。无论使用哪种测试用例，测试分析师必须明确通过/不通过准则。测试分析师根据质量准则（参见第 *1.3.2* 节）为新增或更改的测试条件设计测试用例。对于回归测试，通常选择现有概要测试用例或根据优先级调整现有详细测试用例即可。

测试分析师记录测试依据、测试条件和测试用例之间的可追溯性。在基于经验的测试中，测试用例不一定都会形成文档；相反，测试条件（及其他）可能会指导测试执行。测试分析师可基于概要测试目的设计部分测试用例。

此外，测试分析师定义测试环境需求（参见第 1.3.3 节），并识别、创建和指定测试数据的需求（参见第 1.3.5 节）。

测试分析师使用测试规划中定义的出口准则来确定何时设计了足够多的测试用例。然而，其他出口准则，如剩余风险水平或项目约束（例如预算或时间），也能指示测试设计何时可以结束。

测试设计可以借助工具来实现，但应不受工具和技术的限制，以保持独立性。测试设计采用系统化的测试技术方法，或在其他情况下采用临时方法。

测试用例具有沟通作用，应让利益相关方能够理解。由于测试用例可能不由其作者执行，其他测试人员需要理解如何执行这些测试用例、其测试目的（即其底层测试条件）及重要性。测试用例还必须能被开发者理解，他们可能需要实施测试或在失效时重新运行测试，以及被审计人员理解，他们可能需要批准测试用例。

1.2.3 测试实施

在测试实施期间，测试分析师提供测试执行所需的测试件。测试分析师可将测试规程和测试脚本整合成测试套件，或建议测试用例进行自动化。定义测试规程需仔细识别可能影响测试执行顺序的约束和依赖关系。除了测试用例中的步骤外，测试规程还包括设置初始前提条件（例如从数据存储库加载测试数据）、验证期望结果和后置条件，以及执行后重置步骤（例如重置数据库、环境和系统）。

测试分析师根据风险分析和测试规划中确定的优先级标准，对测试执行所需要的测试规程和测试脚本进行优先级排序，并识别应在当前测试对象版本上执行的测试规程或测试脚本。这使得相关测试（例如新功能或回归测试）可在特定测试运行中一起执行。测试分析师更新测试依据与其他测试件（如测试规程、测试脚本和测试套件）之间的可追溯性。

测试分析师可协助测试经理（TM）制定测试执行计划，包括资源分配，通过确定测试执行顺序实现高效的测试执行（参见ISTQB-CTFL v4.0.1，第 5.1.5 节）。

测试分析师创建输入和环境数据，以便加载到数据库和其他存储库中（参见第 *1.3.5* 节）。这些数据必须“符合用途”，以支持特定测试目的。

测试分析师还应验证测试环境是否完整搭建并准备就绪，可进行测试执行（参见第 1.3.3 节）。最佳方法是通过设计和运行冒烟测试进行验证。测试环境应能在测试执行过程中揭示测试对象中的缺陷，在未发生故障时正常运行，并根据需要充分模拟生产或最终用户环境。

测试实施期间所开展工作的细化程度和相关复杂度，可能会受到测试条件和测试用例详细程度的影响。在某些情况下，需遵循监管规则，测试件应提供符合适用标准（例如RTCA DO-178C, 2011）的证据。

1.2.4 测试执行

测试执行按照测试执行计划进行。测试分析师执行的典型任务包括执行测试、比较实际结果与期望结果、分析异常、报告缺陷和记录测试结果。

测试分析师手动执行测试，包括探索性测试、执行测试规程、回归测试和确认测试。对于探索性测试，测试分析师可使用基于会话的测试和测试章程（参见第 *3.4.1* 节）。测试分析师也可运行自动化测试脚本，但在出现失效时，运行和分析自动化测试脚本可能是开发者、测试自动化工程师（TAE）或技术测试分析师（TTA）的任务。

测试分析师分析手动或自动化测试执行中发生的异常以确定其可能原因。异常可能是测试对象中存在缺陷的结果，但也可能由其他原因引起，包括缺少前置条件、测试数据错误、测试脚本或测试环境的缺陷，或对规格说明的误解。测试分析师记录测试执行的实际结果，根据观察到的失效沟通缺陷，并在需要时进行报告。

测试分析师通过考虑测试结果更新测试依据与其他测试件之间的可追溯性。此信息能够将测试结果转化为高层级风险或覆盖率信息，从而使利益相关方做出知情决策。例如，这向利益相关方阐明，与某测试条件相关的测试用例通过或不通过的数量。

除了这些典型任务外，测试分析师还评估测试结果，包括以下任务：

- 识别缺陷集群，这可能表明需要对测试对象的特定部分进行更多测试（参见第5.3.1节）。
- 手动重新执行不通过的自动化测试，以确保测试自动化未产生假阳性结果。
- 根据之前测试中学到的内容，建议开展额外的测试。
- 根据从测试执行中获得的信息，识别新的风险。
- 建议改进测试设计或测试实施（例如改进测试规程）或甚至对被测系统本身也提出改进建议。
- 建议改进回归测试套件，包括重构、范围调整和测试自动化（参见第2.2.1节）。

1.3 与工作产品相关的任务

测试分析师必须确保其负责的工作产品质量，包括测试用例、测试环境、测试数据、测试结果参照物和测试脚本。本节讨论测试分析师与测试件相关的任务以及管理测试件的工具类型。

1.3.1 概要测试用例与详细测试用例

一个概要测试用例（也称为抽象测试用例或逻辑测试用例）描述测试对象在哪些情况下被检查，表明测试用例覆盖了哪些测试条件。概要测试用例（也称为抽象测试用例或逻辑测试用例）通过指明覆盖哪些测试条件，描述了测试对象被测试的场景。因此，概要测试用例适用于确保测试覆盖所有相关测试条件。概要测试用例不包含前置条件、输入数据、预期输出或后置条件的具体信息，这些信息以摘要方式表达（例如，“订购多本图书，且订单金额满足折扣条件；期望结果：给予相应折扣”）。

详细测试用例（也称为具体测试用例或物理测试用例）是对概要测试用例的细化。详细测试用例描述需要准备的数据、测试人员必须执行的操作（如果需要）以及具体的期望结果。详细测试用例包含具体的前置条件、输入数据、期望结果和后置条件（例如，“订购书籍B1（$10）和B2（$20），订单总额$30；期望结果：给予 10%折扣，总价格：$27”）。

测试分析师通常先设计概要测试用例，将其作为开发详细测试用例的基础。一个概要测试用例可转化为一个或多个详细测试用例来实施。有时，测试用例可能保持概要状态，具体信息由测试分析师在测试执行期间确定。例如，在基于会话的测试中，概要测试用例可指导测试分析师在测试章程中创建测试目的，允许测试分析师在测试执行期间细化这些目标（参见第 *3.4.1* 节）。

从概要到详细测试用例的转换不仅是填入具体值，还是从概念到技术的转变。这一过程常从测试设计推迟到测试实施，特别是在需要特定测试数据的情形下。测试分析师必须确保执行详细测试用例所需的一切信息均已知（Koomen et al., 2006）。在实践中，许多测试用例是混合型的，某些方面具体，某些方面概要，这通常是测试用例可维护性与可理解性之间的权衡。

1.3.2 测试用例的质量准则

忽视测试用例质量可能导致许多问题，如高维护性成本、降低可理解性或执行延迟。测试用例的质量准则是实现更可维护测试用例的第一步，包括：

- **正确性：**测试用例必须能够准确验证其基于的测试条件。
- **可行性：**测试用例必须能够被执行。
- **必要性：**每个测试用例应覆盖明确的测试目的，在测试用例的标题或摘要中体现。应避免重复。不应为不需测试的内容设计测试用例。
- **可理解性：**测试用例可能由作者之外的人员评审、修改和执行。测试分析师应使用利益相关方能理解的语言和格式编写测试用例，避免解释显而易见的内容。复杂测试用例应简化或拆分。
- **可追溯性：**测试用例应可追溯到测试条件、需求和风险，以便测试分析师能够保持其更新（参见ISTQB-CTFL v4.0.1，第1.4.4节）。
- **一致性：**语言、格式和结构的一致性使测试用例更易理解和维护。测试分析师可为此使用术语表。

- **精确性：**测试用例应只有一种解释，以避免假阴性和假阳性测试结果。应避免使用模糊术语，如“适合的”、“按需”或“若干”。
- **完整性：**所有必要属性（例如，参见ISO/IEC/IEEE 29119-3, 2021）均应包含，包括所需测试数据（参见第1.3.5节）和明确的期望结果，以避免与实际结果比较时的疑问。
- **简洁性：**测试用例的粒度（即一个包含多个测试动作的大型测试用例与多个较小的测试用例）应与测试依据和测试条件相匹配。较小的、专注于少数覆盖项的测试用例更可取，因其便于查找失效原因，还可灵活组合成测试规程和测试套件，且测试执行期间若发生失效，也不会阻碍后续其他测试的执行。

测试用例的格式和详细程度取决于项目和产品背景，应在测试团队内部达成一致。

1.3.3 测试环境需求

测试环境是手动和自动化测试执行的关键成功因素。测试环境的实施影响可测试性、缺陷检测、总体测试成本和测试结果的可靠性。在测试环境中通过或不通过的测试用例在生产环境中执行时应具有相同结果。理想情况下，测试环境应稳健、可预测，并在需要时与测试自动化框架集成。测试分析师可在测试设计期间基于对以下内容的分析来确定测试环境需求：

- 测试条件、测试用例和测试数据需求，测试环境需求描述设置和维护测试环境所需的条件，以确保测试执行的前置条件得到满足。
- 测试级别和测试类型，影响测试环境灵活性与生产环境相似性之间的权衡。
- 组件和系统的可用性和独立性，可能表明需要使用测试替身（例如，桩或驱动程序）。

测试环境需求描述测试环境项，可分为多种类型，如硬件、中间件、软件、虚拟化服务、网络、接口、工具、安全、配置和场所。对于每个测试环境项，需求应包括以下信息（ISO/IEC/IEEE 29119-3, 2021）：

- 唯一标识符——用于可追溯性目的。
- 描述——足够详细以按需实现。
- 职责——描述谁负责使其可用。
- 所需时间段——确定何时用及需要用多长时间。
- 保真度——该项与生产环境的代表性或偏差程度。

需求还应涵盖测试环境的总体需求，包括设置、备份和恢复、安全需求、调整测试环境的能力以及角色和授权（Koomen et al.,2006）。

测试分析师应清晰、简洁、连贯地记录测试环境需求，可使用示意图或表格。为避免冗余和不必要的文档，测试分析师可引用现有测试环境，并专注于测试级别的特定需求。相关利益相关方（例如开发者、技术测试分析师、测试自动化工程师、业务分析师、项目资助方和产品负责人）也应评审、批准和更新测试环境需求。

1.3.4 确定测试结果参照物

在动态测试中，需要测试结果参照物来确定期望结果。理想情况下，测试依据将提供测试结果参照物（例如，文本或正式规格说明）。人类对测试对象的经验或知识也可作为测试结果参照物。为成本效益考虑，可能需要自动化的测试结果参照物（例如，输入自动生成或人工参照物成本高昂）。

根据测试依据或系统特性的质量和完整性情况，可能无法获得成本效益高的测试结果参照物，这被称为测试结果参照物问题。导致测试结果参照物问题的典型因素包括数据相关复杂性、非确定性（例如，基于AI的系统）、概率行为以及缺失或模糊的需求。

测试结果参照物问题的一些已知解决方案包括（Barr et al.,2014）：

- 伪测试结果参照物：是独立开发的系统，其满足与测试对象（例如，遗留系统、测试对象的简化版本）相同的规格说明。为复杂测试对象开发专用伪测试结果参照物可能成本高昂，但对于关键系统而言，这种做法十分常见。
- 基于模型的测试：将测试结果参照物作为测试模型的一部分形式化，允许生成预期输出并从模型中推导测试。基于行为的测试技术通常涉及实施自动化测试结果参照物（参见第*3.2.2*节和第*3.2.3*节）。
- 基于属性的测试：使用测试对象的指定属性来验证输入与单个测试用例期望结果之间的关系。如果未满足此关系，测试用例不通过。此解决方案适合测试自动化，但其有效性取决于关系，确定关系可能困难。
- 蜕变测试（参见第*3.3.2*节）。
- 人工参照物：利用人类确定期望结果的能力。人力资源可能成本高昂且稀缺。但在某些测试方法中（如探索性测试）人工参照物更受青睐。

断言可嵌入测试自动化代码或测试对象本身以实现自动化测试结果参照物。它们是验证测试对象状态或行为的可执行语句。在嵌入测试对象中时，通常仅验证任务继续所需的条件。

1.3.5 测试数据需求

在测试设计期间，测试分析师识别并请求测试执行所需准备或配置的测试数据，考虑其用途、格式和使用场景。（ISO/IEC/IEEE 29119-3, 2021，第8.5节）也描述了测试数据需求。主要方面包括：

- **与生产数据的相似性：**生产数据反映现实世界数据，但可能缺乏多样性。合成数据可实现可控的多样性，它应反映生产数据模式、分布和异常值的关键方面，但可能忽略仅在现实世界数据中才会出现的缺陷。对于缺乏生产数据的系统，合成数据必须反映真实的业务和技术场景。角色（Personas）通过提供以用户为中心的真实简介，帮助创建反映多样用户场景和行为的数据。
- **保密性：**敏感测试数据（如个人信息）需要予以保护。伪匿名数据用人工标识符替换个人信息。匿名数据移除可识别的个人信息。如有必要，测试分析师必须遵守数据保护法规，如欧盟的GDPR（Commission, 2016）或美国的HIPAA（Health et al., 2024）。
- **用途：**测试数据对确定影响系统状态及其配置（如系统时间和日期）的前置条件和期望结果至关重要，还包括指定用户权限及建立产品、部门和类别之间的关系。
- **覆盖准则：**测试数据必须与所选测试技术的覆盖准则一致。除有效测试数据外，还可能需要无效测试数据，例如用于逆向测试。
- **数据格式：**系统化的数据管理（如在API测试中）可能需要结构化数据（如CSV、JSON、XML或数据库）。
- **可追溯性：**可追溯性确保测试数据在测试用例变更时的维护性。
- **维护性：**应避免在详细测试用例中硬编码测试数据，以便于缺陷检测和维护。测试用例应将测试逻辑与测试数据分离（参见第1.3.2节）。
- **依赖性：**依赖数据需要一系列步骤来创建。
- **可用性：**服务虚拟化可通过模拟缺失或不可访问的服务来解决数据缺失问题，实现与外部系统或服务的交互。
- **时间敏感性和数据老化：**涉及过时或时间敏感数据的测试用例可能以意外或不准确的方

式影响系统行为。

1.3.6 使用关键词驱动测试开发测试脚本

如果使用关键词驱动测试，测试分析师使用关键词创建测试脚本。测试脚本的实施由技术测试分析师（TTA）、测试自动化工程师（TAE）或开发者完成。

测试分析师通过分析测试依据或与利益相关方协作来识别和指定关键词。关键词可分为两类：动作关键词和验证关键词。动作关键词必须与测试对象（例如执行功能、提交数据、在测试对象内导航）、测试环境（例如设置配置、激活模拟器）或其他组件或系统（例如触发测试对象的接口）交互。验证关键词表示断言，用于评估测试对象产生的实测结果是否与期望结果匹配。

关键词至少存在于两个抽象层：领域层和测试接口层。领域层关键词对应于业务相关动作，反映应用领域的术语，从测试对象接口的技术细节中提取。测试接口层关键词通过测试接口与测试对象、测试环境项或其他组件或系统通信，位于最低抽象层。额外的中间层可支持关键词的可维护性。

关键词可以是原子化的或由其他关键词组成。关键词的结构和抽象层是独立属性。然而，复合关键词通常位于较高抽象层，而原子关键词通常位于测试接口层。

测试分析师在关键词驱动测试中的任务包括：

- 指定关键词及其参数。
- 指定关键词测试用例，即使用关键词的测试脚本。
- 指定使用关键词测试脚本的额外步骤，如前置条件、验证动作和测试后清理测试环境。
- 维护关键词测试用例以反映测试对象的变更。
- 执行关键词测试脚本，可为自动化或手动执行。
- 分析失效的关键词测试用例以确定失效原因。

在分析测试依据时，测试分析师寻找测试对象与其环境（如用户、其他系统和设备）的交互。例如，用户故事是“作为会员，我想认证自己，以便访问设施”，其验收准则为“有效会员卡可用于认证”。测试分析师可能指定一个（动作）关键词“Authenticate Member”包含参数“member card”和一个验证关键词“Verify Access”。测试分析师检查识别的关键词是否位于适当的抽象层。

指定关键词时，测试分析师需牢记，关键词必须：

- 包含动词（+名词）。
- 使用动词的命令式形式（+名词）。
- 含义唯一
- 充分文档化。
- 反映应用领域的词汇。
- 可复用。

在编写关键词测试用例时，测试分析师可能识别缺失的关键词，并立即对这些关键词予以指定。关键词测试用例可以列表或表格等格式编写。

关键词在项目生存周期中可能发生变化，易于被冗余指定（Rwemalika et al.,2019）。本节提到的建议旨在避免冗余并减少维护工作。

尽管关键词驱动测试是一种测试自动化方法，手动测试也可从中受益。如果用于手动测试，它有效支持从手动测试到自动化测试的后续转换。

有关测试执行自动化和关键词驱动测试的更多细节，可参见（ISTQB-TTA,v4.0）、（ISTQB-TAE,v2.0）和（ISO/IEC/IEEE 29119-5,2016）。

1.3.7 管理测试件的工具

使用工具进行适当的测试件管理可帮助测试分析师支持整体测试过程。工具可提供工作产品的状态，支持测试监督和测试控制。在被测系统失效时，工具允许测试分析师检查先前测试运行的测试结果并分析缺陷发生的时间点。

管理测试件的关键工具类型包括：

- **测试管理工具：**用于提供所有相关测试件的存储库（如测试条件、测试用例、测试脚本、测试套件和测试运行）。工具支持可追溯性（例如通过可追溯性矩阵）、测试用例检索、测试运行调度、测试结果记录以及测试进度和质量的整体报告。
- **缺陷管理工具：**用于记录缺陷、确定缺陷优先级和监控缺陷解决过程。
- **测试数据管理工具：**用于创建和维护测试数据，包括保护敏感数据（参见第1.3.5节）。
- **配置管理工具：**用于支持开发、发布和操作过程中的测试相关活动，包括管理测试环境

的配置和可用性。

- **需求管理工具：**用于包含和跟踪概要需求，确保这些需求在整个SDLC中明确定义、版本控制和可追溯。

测试分析师通过以下方式支持测试件管理：

- 分析项目和版本发布以选择正确的测试件子集（例如，选择与被测系统版本匹配的标注版本的规格说明）。
- 为测试管理工具中的测试用例定义适当的功能结构（例如按功能或模块）或技术结构（例如按测试类型或环境）。
- 为测试用例添加元数据（例如与测试执行或特定测试环境相关的工作量信息）。
- 确保需求、测试条件、测试、测试运行和缺陷之间的可追溯性。
- 为回归测试选择正确的测试套件（用于手动/自动化测试执行）。
- 测试用例的配置管理，包括识别过时的测试用例。

工具帮助组织、跟踪和确保测试过程的质量。测试分析师通过选择、结构化和维护测试用例、确保可追溯性以及管理测试用例版本，支持高效测试和测试控制。

2. 测试分析师在基于风险的测试中的任务 - 90分钟

关键词

影响分析（impact analysis）、产品风险（product risk）、回归测试（regression testing）、风险分析（risk analysis）、风险评估（risk assessment）、风险控制（risk control）、风险识别（risk identification）、风险缓解（risk mitigation）、风险监测（risk monitoring）、基于风险的测试（risk-based testing）

第2章的学习目标：

2.1 风险分析

TA-2.1.1 (K2) 总结测试分析师对产品风险分析的贡献。

2.2 风险控制

TA-2.2.1 (K4) 分析变更的影响，以确定回归测试的范围。

测试分析师在基于风险的测试中的任务简介

基于风险的测试是一种根据测试项风险级别确定测试工作优先级的测试方法，由测试经理确定该方法。测试分析师在实施该方法中扮演积极角色。基于风险的测试在（ISTQB-TM, v3.0）中有更详细的描述。

2.1 风险分析

根据（ISTQB-CTFL, v4.0.1，第5.2节），风险分析包括风险识别和风险评估。本教学大纲聚焦于测试分析师应如何参与风险分析活动，以确保正确实施基于风险的测试。

2.1.1 测试分析师对产品风险分析的贡献

测试分析师通常具备系统的独特知识，以及对常见问题、其影响以及测试如何缓解风险的经验和直觉知识。这使得测试分析师成为产品风险分析的重要利益相关方。

在风险识别中，测试分析师利用自身经验和知识参与回顾会、风险研讨会、头脑风暴和创建检查表。测试分析师还可与利益相关方进行访谈，以更好地理解利益相关方认为最重要的风险。

在风险评估中，测试分析师与其他利益相关方一起，通过估算以下因素确定风险级别：

- 受影响功能的使用频率和关键性
- 受影响业务目标的关键性
- 财务、环境和声誉损害
- 测试依据的质量
- 法律或安全需求

测试分析师还根据受影响的质量特性帮助对产品风险进行分类，例如使用ISO/IEC 25010（2023）的产品质量模型。风险级别通常在测试对象中分布不均。在这种情况下，测试分析师应将测试对象分解为测试项（例如组件、接口和功能），并分别评估每个测试项所对应的风险。

最后，作为产品风险评估的一部分，测试分析师提出适合的测试活动以缓解每个已识别的产品风险。这些活动可能包括静态测试和动态测试。根据风险级别、相关测试项类型和受影响的质量特性等因素，测试分析师指明所需的测试级别、测试类型、测试技术、测试独立性水平和测试彻底性。本着“左移”精神，测试分析师指明哪些测试活动可尽早缓解风险，以最大限度减少测试工作量。

2.2 风险控制

根据（ISTQB-CTFL，v4.0.1，第5.2.4节），风险控制包括风险缓解和风险监测。测试分析师（TA）了解系统的功能及其相关的潜在风险。因此，测试分析师在（ISTQB-CTFL，v4.0.1，第5.2.4节）提到的多项风险缓解活动中扮演关键角色：

- 执行评审，详见本大纲第5.2.2节。
- 应用适当的测试技术和覆盖水平，详见第3.5节
- 应用适当的测试类型，详见第4章。
- 执行回归测试，见下文所述。

此外，由于风险和风险级别并非静态而是随时间变化，基于风险的测试涉及定期的风险监测。在迭代生存周期中，监测频率由团队决定（通常每次迭代监测一次）。在其他生存周期中，监测频率由负责产品风险管理的人员（通常是测试经理）设定。测试分析师通过根据变更更新风险登记册，并调整上述风险缓解活动来做出贡献。

2.2.1 确定回归测试的范围

回归测试的主要目标是确保变更发生后，对测试对象的质量仍保持信心。然而，由于约束（例如时间、预算、测试环境或测试数据的限制），在回归测试期间可能无法执行所有回归测试。这一问题主要与手动执行的测试相关，但也适用于自动化测试——例如当测试周期短且存在许多长时间运行的自动化测试时。这使得需要根据特定标准选择适当的回归测试。每次测试周期都需要评审回归测试的范围。评审可能得出结论，现有回归测试套件需要调整。

选择自动化测试最可靠的技术是影响分析，可由工具支持。此类工具基于自动化配置管理系统，记录每个测试用例执行期间激活的配置项。当发生变更时，工具跟踪哪些配置项被修改，并选择与变更项交互的回归测试。通过这样，工具确保在配置项变更后，测试重点关注最可能发现失效的区域（Juergens et al.，2018）。

对于手动测试执行，尚无确凿证据表明某种回归测试选择技术明显优于其他技术，因为结果取决于多种因素（Engström et al.，2010）。因此，测试分析师必须根据具体情况决定使用哪种技术。常用技术包括：

- **基于风险的测试选择：**测试分析师维护回归测试套件与风险登记册的可追溯性。当发生

变更并相应更新风险登记册时，测试分析师调整回归测试套件以覆盖最高风险级别。

- **基于历史的测试：**测试分析师评估过去的测试执行，确定哪些测试暴露了缺陷或对自上次测试执行以来发生的类似变更敏感。作为回归测试的一部分，再次执行相应测试，增加了暴露类似缺陷的可能性。测试分析师还可包括一些长时间未执行的测试，以确保其仍通过。
- **基于覆盖的测试：**测试分析师根据所选的测试技术，选择少量能实现尽可能多覆盖的测试。测试数量必须与每个测试的覆盖增加量仔细平衡。
- **需求可追溯性矩阵：**用于评估需求变更对相关测试的影响。当新需求或变更需求间接影响现有功能时，此方法特别有价值。测试分析师选择直接受影响功能及相关功能的回归测试，以覆盖可能的意外副作用。在敏捷软件开发中，可通过选择覆盖受新增或变更用户故事影响的验收准则的测试来完成类似操作。
- **基于运行配置的测试：**测试分析师根据测试对象的使用模式选择回归测试用例。例如，在测试在线商店时，一个测试可包括用户登录、搜索产品、添加到购物车和下订单。当应用程序发生重大变更时，此技术提供对整体系统功能的快速概览。如果测试过多，测试分析师优先选择经常发生且覆盖关键功能和业务流程的关键使用模式。
- **影响分析：**如果测试分析师知道哪些测试用例与变更的配置项交互，也可采用影响分析来选择用于手动执行的回归测试用例。

通常需要结合多种选择技术，以构建更全面和有效的回归测试套件。然而，测试分析师必须仔细平衡覆盖需求与测试套件的可管理规模。在每个测试周期后，测试分析师分析测试结果以确定所应用技术的有效性（参见第5.3.1节）。下次，测试分析师保留有效技术并替换无效技术。这会随时间持续改进回归测试选择。在变更频繁的迭代和增量开发模型中，这一点尤为重要。

3. 测试分析与测试设计 - 615 分钟

关键词

基于检查表的测试（checklist-based testing）、基于行为的测试技术（behavior-based test technique）、组合测试（combinatorial testing）、众测（crowd testing）、CRUD测试（CRUD testing）、基于数据的测试技术（data-based test technique）、判定表测试（decision table testing）、域测试（domain testing）、等价类（equivalence partition）、基于经验的测试（experience-based testing）、蜕变关系（metamorphic relation）、蜕变测试（metamorphic testing）、随机测试（random testing）、基于规则的测试技术（rule-based test technique）、基于场景的测试（scenario- based testing）、基于会话的测试（session-based testing）、状态转移测试（state transition testing）、测试章程（test charter）

第3章的学习目标

3.1 基于数据的测试技术

TA-3.1.1 (K3) 应用域测试。

TA-3.1.2 (K3) 应用组合测试。

TA-3.1.3 (K2) 总结随机测试的收益和局限性。

3.2 基于行为的测试技术

TA-3.2.1 (K2) 解释 CRUD 测试。

TA-3.2.2 (K3) 应用状态转移测试。

TA-3.2.3 (K3) 应用基于场景的测试。

3.3 基于规则的测试技术

TA-3.3.1 (K3) 应用判定表测试。

TA-3.3.2 (K3) 应用蜕变测试。

3.4 基于经验的测试

TA-3.4.1 (K3) 为基于会话的测试准备测试章程。

TA-3.4.2 (K3) 准备支持基于经验的测试的检查表。

TA-3.4.3 (K2) 举例说明众测的收益和局限性。

3.5 应用最合适的测试技术

TA-3.5.1 (K4) 选择适当的测试技术来缓解给定场景的产品风险。

TA-3.5.2 (K2) 解释自动化测试设计的收益和风险。

测试分析与测试设计简介

测试技术主要用于测试分析和测试设计。本章讨论的测试技术涵盖黑盒测试技术和基于经验的测试技术。白盒测试技术在（ISTQB-TTA, v4.0）中讨论。

本大纲将黑盒测试技术根据其建模的底层测试条件分为三类：

- 数据元素（基于数据的）。
- 动态行为元素（基于行为的）。
- 静态行为规则元素（基于规则的） 。

3.1 基于数据的测试技术

在本节中，术语“域”用于表示测试项的输入数据集合。来自域的不同区域的输入数据会导致测试项的不同行为。基于数据的测试技术旨在验证实现是否正确处理特定的域区间。基础级课程大纲（ISTQB-CTFL, v4.0.1，第4.2节）涵盖了可归类为基于数据的两种测试技术：等价类划分（EP）和边界值分析（BVA）。本大纲介绍了三种进一步的基于数据的测试技术：

- 域测试：将EP和BVA扩展到具有多个参数和复杂划分的域。
- 组合测试：关注多维域中多个参数的交互。
- 随机测试：根据指定的概率分布从域中随机选择输入。

注意，随机测试通常可指随机输入数据和随机事件。本大纲仅讨论随机输入数据的测试。

3.1.1 域测试

域测试验证测试项在域的等价类及其边界上的行为是否符合规范。在这种情况下，等价类使用通过布尔运算符（例如AND、OR和NOT）组合的原子条件来定义，涉及一个或多个交互变量。每个原子条件定义等价类的边界。闭合边界由关系表达式中的运算符≤、≥或=形成，开放边界由关系表达式中的运算符<、>或≠形成。

例如，表达式“身高 > 1.29米且体重 / 身高² ≥ 30”定义了由两个变量组成的二维域的等价类，具有一个开放边界和一个闭合边界。

域测试通过选择能揭示等价类实现中缺陷（例如错误运算符或常数）的适当覆盖项来识别缺陷。域

测试的覆盖准则涉及ON、OFF、IN和OUT点：

- 对于闭合边界，ON点位于边界上。对于开放边界，ON点位于等价类内部，且根据给定精度最接近边界。
- 对于闭合边界，OFF点位于等价类外部，且根据给定精度最接近边界。对于开放边界，OFF点位于边界上。
- IN点属于等价类且不是ON点。
- OUT点位于等价类外部且不是OFF点。

这些点类型与等价类的边界相关。同一点对不同边界可能具有不同类型。

覆盖准则包括：

简化域覆盖（Jeng et al., 1994），要求以下覆盖项：

- 对于由<、≤、>或≥运算符定义的每个边界，一个ON点和一个OFF点。
- 对于=运算符，一个ON点和位于边界两侧的两个OFF点。
- 对于≠运算符，一个OFF点和位于边界两侧的两个ON点。

每个OFF点应根据给定精度尽可能接近对应的ON点。

可靠域覆盖（Forgács et al., 2024），要求以下覆盖项：

- 对于由<、≤、>或≥运算符定义的每个边界，一个ON点、一个OFF点、一个IN点和一个OUT点。
- 对于=运算符，一个ON点和位于边界两侧的两个OFF点。
- 对于≠运算符，一个OFF点和位于边界两侧的两个ON点。

对于这两种覆盖准则，覆盖项数量与边界数量呈线性关系。可通过对不同边界使用相同覆盖项进行优化。例如，所有等价类的边界可使用公共IN点，或一个边界的ON和OFF点对可作为相邻等价类边界的OFF和ON点。对于具有多个边界的域，可使用高级优化算法（Forgács et al., 2024）。

可靠域覆盖比简化域覆盖产生略多的覆盖项，但可检测更多的域缺陷（Site of Software Test Design, 2020）。

域测试可应用于任何测试级别。它将BVA和EP（参见ISTQB-CTFL, v4.0.1，第4.2节）推广到更复杂的域。域测试的各种方法可在教科书中找到，如（Beizer, 1990，第6章；Binder, 2000，第10.2.4节；

Kaner; Padmanabhan, et al., 2013；Jorgensen, 2014，第5章）。

3.1.2 组合测试

某些软件失效源于参数值的特定组合，称为交互失效。组合测试旨在通过探索这些参数值组合来揭示此类失效。

在组合测试中，测试条件通常结合配置参数或输入数据值。因此，组合测试有两种主要方法。第一种方法使用配置参数的组合，可使用相同的测试用例对每个组合进行测试。第二种方法使用输入数据值的组合，这些组合成为完整测试用例的一部分，为被测系统创建测试套件（D. R. Kuhn et al., 2013）。

特定参数-值对由参数及其值组成（例如“(颜色, 红色)”）。

组合覆盖准则包括以下内容（Ammann et al., 2008；Forgács et al., 2019）：

- **基本选择覆盖：**假设某些参数-值对比其他参数-值对更重要。为每个参数选择一个基础参数-值对，基础覆盖项是基础参数-值对的组合。后续覆盖项通过将每个参数的基础参数-值对替换为每个非基础值来创建。
- **成对覆盖：**覆盖项是任意两个参数的参数-值对。工具可用于生成覆盖项。然而，找到实现成对覆盖的最小测试用例集通常很困难。

对于具有多个值的参数，可先应用等价类（EP）以减少值数量和结果组合集。使用分类树或特征模型（参见IREB-Glossary, 2024）捕获参数及其值支持此活动。最终测试用例数量可能受参数-值对之间的约束、手动添加的已知问题组合或无效/不可行组合的影响。

组合测试的关键见解是并非每个参数都导致失效，大多数失效由单一参数值或较少参数之间的交互触发（Cohen et al., 1994）。这与耦合效应假设一致，表明发现程序中的简单缺陷通常也能发现复杂缺陷（Offutt, 1992）。在一项有限研究中（D. Kuhn et al., 2004），结果显示约97%的失效由一到两个交互条件引起，表明成对测试是一种有效的测试技术。

有关组合测试的更多信息，包括其他覆盖准则如diff-pair-t（只覆盖具有“差异”的参数值对组合）或n-wise（任意n个参数的组合都要覆盖）测试，可参见（Ammann et al.,2008；Forgács et al., 2019）。支持组合测试的工具也可在（Czerwonka, 2004）中找到。

3.1.3 随机测试

随机测试涉及根据指定的概率分布从测试项的输入域中随机选择测试数据。为验证目的，建议基于运行配置的分布。为验证目的，分布应与使用无关以避免偏见。期望结果可能通过测试结果参照物添加到测试用例中，通常需要自动化测试结果参照物。

随机测试可分为引导式和非引导式。在非引导式随机测试中，概率分布在整个过程中保持固定。引导式随机测试，如自适应随机测试（Huang et al., 2019）这类技术，根据先前选择的值调整分布，并随时间演变。引导式随机测试旨在有效覆盖输入域，鉴于缺陷通常聚集在特定域区间。

随机测试缺乏公认的覆盖准则。因此，出口准则只能依赖执行的测试数量、测试时间或类似的完成度量。

随机测试在域知识有限或需要大量测试数据时尤为有价值。它成本效益高，并以概率方式提供对测试对象可靠性的见解。随机测试有助于避免偏见，如因对某些代码或功能的不当信任而在手动测试中忽略缺陷。然而，随机测试也存在若干挑战和局限性，包括忽略数据语义、可能错过与数据含义相关的缺陷、忽略某些缺陷、生成冗余测试、依赖自动化测试结果参照物以及随机输出导致测试结果不一致。在每种测试场景中平衡随机测试的收益和局限性至关重要。

传统上，随机测试被认为比其他测试技术效果较差。近年来，许多实证研究对此假设进行了探讨。研究发现，在上述情况下，随机测试可能比其他基于数据的测试技术更有效和高效（Arcuri et al., 2012；Wu et al., 2020）。随机测试也应用于模糊测试和混沌工程。

3.2 基于行为的测试技术

基于行为的测试技术从测试项的动态（即状态依赖）行为规格说明中推导测试用例。本大纲讨论三种基于行为的测试技术：

- CRUD测试。
- 状态转移测试。
- 基于场景的测试。

CRUD测试和基于场景的测试扩展了（ISTQB-CTFL, v4.0.1，第4.2节）中已知的黑盒测试技术范围。本大纲为状态转移测试补充了额外的覆盖准则。

3.2.1 CRUD测试

CRUD测试验证测试项处理的数据实体的生存周期。CRUD代表创建（Create）、读取（Read）、更新（Update）和删除（Delete），这是功能可对实体执行的四种基本操作。

CRUD矩阵提供数据实体生存周期的概览。其列表示实体，行表示功能。假设一个功能对给定实体执行一个或多个特定的创建、读取、更新或删除操作，这在矩阵中使用操作的首字母C、R、U或D表示。为创建CRUD矩阵，测试分析师确定每个功能对哪些实体执行哪一种操作（共四种操作）。需特别注意读取操作，通常与C-U-D操作存在隐式关联。

CRUD测试包括两部分（Koomen et al.,2006）：

- **完整性测试：**静态测试，验证是否对每个实体实现了所有可能的操作（即C、R、U和D）（也就是验证是否为每个实体实现了完整的生存周期）。缺少某个操作是需要调查的异常。
- **一致性测试：**动态测试，旨在整合各种功能并检查实体是否被一致使用。验证功能在处理实体时是否正确交互。测试用例应覆盖CRUD矩阵中的所有操作。此外，还应包括逆向测试（例如，读取尚未创建的实体）。测试用例按实体设计，通过组合功能覆盖其整个生存周期。

CRUD覆盖通过测试套件执行的操作数除以CRUD矩阵中的总操作数来衡量。更严格的CRUD覆盖可考虑将特定操作组合作为覆盖项（例如，每次U后，应覆盖所有可能的R）

CRUD测试主要用于系统级别，重点关注测试项在处理实体时的行为缺陷，如数据完整性违规、访问控制缺陷或数据不一致。

3.2.2 状态转移测试

许多复杂系统是有状态的（即系统对事件的反应取决于系统的当前状态）。有状态系统的示例包括嵌入式系统、基于对话的系统、控制系统或处理实体及其生存周期的系统。

状态简化了复杂的内部细节，便于利益相关方理解预期行为。在测试分析期间，测试分析师必须确保基于状态的模型以测试所需的详细程度表示测试项的预期行为。如果测试依据已包含此类模型，测试分析师需检查其是否包含测试条件，并在需要时调整或设计新模型。

在基于状态的模型中，节点表示状态，边表示状态转移。状态转移由事件触发，可能包括守卫条件

和动作（参见ISTQB-CTFL,v4.0.1，第4.2.4节）。此类模型有多种变体，如扩展有限状态机（Bochmann et al., 1994）、Harel状态图（Harel, 1987）或UML状态机（OMG® UML, 2017）。

除（ISTQB-CTFL, v4.0.1）讨论的状态转移覆盖准则外，下文将讨论另外两种覆盖准则，经实证证明，它们具有较高的缺陷检测有效性：

- **N-切换覆盖：**适用于N+1个连续转换的有效序列，也称为N-切换（Chow, 1978）。0-切换覆盖等同于有效转换覆盖（参见ISTQB-CTFL, v4.0.1，第4.2.4节）。1-切换是一个状态的输入和输出转换对。通过在N-切换末端扩展有效的后续状态转换，形成N+1切换。在实践中常使用0-切换和1-切换覆盖。由于意外事件序列导致失效存在高风险时，才需要采用100% 的2-切换或更高覆盖，因为N-切换数量随N呈指数增长。
- **环路覆盖**（Ammann et al., 2008）：适用于形成循环的基于状态模型中的路径。环路是一个起点和终点状态相同的循环，且循环中无其他状态重复出现。覆盖项为环路。（Antoniol et al., 2002）发现此准则在发现缺陷方面高度有效。

状态转移测试非常适合使用基于模型的测试工具进行自动化。与大多数黑盒测试技术一样，状态转移测试通过在建模期间检测规格说明中的缺陷帮助实现缺陷预防。状态转移测试可应用于任何测试级别。

进一步的状态转移覆盖准则在（Rechtberger et al., 2022）中讨论。近期基于状态的模型为（Forgács et al., 2024）讨论的动作-状态模型。

3.2.3 基于场景的测试

基于场景的测试评估测试项在现实场景中的行为。用户研究、用户故事、角色和用户旅程图（参见第11节）等技术可帮助识别有价值的场景。在基于场景的测试中，测试分析师基于构成测试项工作流程的动作序列创建场景模型（参见ISO/IEC/IEEE 29119-4, 2021）。本大纲讨论以下两种模型：活动图和用例。其他模型包括流程图、业务流程建模和符号图（OMG® BPMN, 2013）、序列图或协作图（OMG® UML, 2017）。这些模型不在本大纲讨论范围，详情参见（ISTQB-AcT, v1.0）。

活动图是系统内工作流程的图形表示。活动图特别适合建模业务流程，但也可建模控制流。活动图扩展了流程图的符号，允许建模并发性。活动图的主要元素包括开始和结束节点、动作、转换、决策节点、合并节点、分叉节点、连接节点和泳道。

用例是表示用户与系统或系统之间交互的文本或图形描述。模型中区分三种场景：

- **主场景**（即“快乐路径”）：从用户角度实现特定目标的典型、预期动作序列。一个用

例只有一个主场景。

- **扩展**（或替代场景）：除主场景外的动作序列，最终实现主场景目标。
- **异常**：由于意外动作（如异常使用或无效输入）无法实现主场景目标的动作序列。

在基于场景的测试中，测试分析师设计测试用例以覆盖场景（即覆盖项），通常遵循基于风险的优先级排序。当场景模型不含循环时，每个场景可用单独的测试用例进行测试（即模型中的所有可能场景可通过一个测试套件来执行）。当存在循环时，潜在场景（路径）数量可能是无穷的。在这种情况下，可对场景模型应用简单循环覆盖。简单循环覆盖要求测试每个循环执行：零次（即跳过）、恰好一次、多次（即典型迭代次数）和最大迭代次数（如果可能）。

基于场景的覆盖通过已执行的场景数除以所有已识别场景数来衡量。可通过对场景模型应用各种覆盖准则来识别场景（参见Koomen et al., 2006）。覆盖准则可能要求多次测试每个场景。例如，场景可能需要对场景中出现的变量进行额外的等价类（EP）或边界值分析（BVA）覆盖。在这种情况下，一个场景可能需要多个测试用例进行测试。

基于场景的测试通常在系统测试或验收测试中执行，表现为从用户角度关注系统功能适用性的端到端测试（参见第4.1节）。然而，基于场景的测试也可在其他测试级别使用（例如，基于交互协议的场景组件集成测试，或基于调用各种方法的有状态面向对象类的组件测试）以及非功能测试（例如，场景可构成用于可靠性、灵活性或兼容性测试的素运行配置元素）。

3.3 基于规则的测试技术

基于规则的测试技术验证测试项的无状态行为的实现，这些行为由不依赖其状态的规则（如业务规则）指定。本大纲讨论两种基于规则的测试技术：

- 判定表测试
- 蜕变测试

基础级课程大纲（ISTQB-CTFL, v4.0.1）涵盖了判定表测试的基础知识。本大纲讨论更高级的主题。使用的术语和符号遵循标准（OMG® DMN, 2024）。

3.3.1 判定表测试

在判定表测试中，测试分析师通常先创建完整判定表或分析从测试依据中获得的现有判定表。完整

判定表的规则数量是其条件值数量的乘积。规则数量随条件及其值数量的增加呈指数增长，这促使对判定表进行最小化。

判定表可通过使用“无关”运算符“-”合并规则来最小化。建议在合并规则时忽略不可行规则（即条件值组合永远不会发生的规则）。通常，在合并前从判定表中移除不可行规则。一种可行的系统化最小化算法是寻找仅在一个条件上不同且覆盖该条件所有可能值的动作等价规则。这些规则会被合并，不同的条件值被替换为“-”。系统化最小化的结果可能取决于列的最小化顺序，不一定总能得到最优解。测试分析师需检查是否可能进一步最小化。

测试分析师的任务是，可能需与利益相关方一起依据以下准则评审判定表：

- 一致性：如果两个不同规则适用于相同的条件值组合，则它们是动作等价的。
- 可行性：不包含不可行规则。
- 完整性：没有遗漏可行的条件值组合。
- 正确性：规则建模系统的预期行为。

此外，建议规则不重叠（即，对于任何条件值组合，最多只有一个规则适用）。当原始判定表已最小化或规则合并不正确时，可能会出现重叠规则。

校验和程序使用最小化判定表中的规则数量指示重叠和缺口。对于最小化表中的每个规则，计算其在原始判定表中代表的规则数量。条件没有“-”的规则（即所有条件具有单个值）得分为1。每个“-”值将规则得分乘以相应条件的单个值数量。规则得分的总和是最小化判定表的校验和。如果校验和小于原始判定表的校验和，表明最小化判定表不完整。如果校验和高于原始判定表的校验和，表明存在规则重叠或额外规则（例如不可行的条件组合）。如果最小化正确执行，校验和相等。仅相等校验和不能保证最小化判定表与原始判定表等价。

判定表覆盖率通过将已执行的列数除以判定表中所有可行列的总数来衡量。在实现由判定表规则生成的测试用例时，测试分析师必须实现条件和动作。测试分析师需决定如何实现给定规则的“-”条件值，因为“-”表示至少两个值。然而，如果判定表关联的风险级别高，测试分析师应避免最小化，并测量完整判定表中可行列的判定表覆盖率。

3.3.2 蜕变测试

蜕变测试（Metamorphic Testing, MT）是一种基于现有源测试用例生成测试用例的技术。通过基于

蜕变关系（Metamorphic Relation, MR）改变（蜕变）源测试用例，生成一个或多个后续测试用例。蜕变关系定义测试项的属性，描述测试用例输入的变更如何反映在其期望结果中。

测试分析师将蜕变关系的源测试用例和后续测试用例组合成测试规程，进行联合结果评估。如果满足蜕变关系，则测试通过，否则不通过。在失效情况下，需后续调试以确定涉及的哪个单独测试用例不通过。

例如，考虑一个计算一组数字平均值的功能。创建一个源测试用例，包含一组数字和预期平均值，并运行测试用例以确认通过。一个蜕变关系可能表明，该组数字的任何排列都产生相同的平均值。使用此蜕变关系，测试分析师可创建多个后续测试用例，输入相同的数字集但顺序不同，期望结果保持不变。

对于同一平均值功能，测试分析师还可使用另一个蜕变关系，表明如果该组数字中的每个数字乘以相同数字x，则期望结果也将乘以x。使用此蜕变关系，测试分析师可通过选择不同的x值从源测试用例创建任意数量的后续测试用例。这在测试设计和执行自动化时尤为有用。测试分析师还可组合两个或多个蜕变关系来创建后续测试用例（例如，排列并乘以2）。

蜕变测试也适用于有测试结果参照物问题的情况（参见第1.3.4节）。在这种情况下，源测试用例和后续测试用例的期望结果不可用，因此无法单独评估其测试结果。例如，一个基于AI的精算程序基于大型数据集预测死亡年龄。蜕变关系可能表明，如果吸烟数量增加，预测的死亡年龄应降低。

目前，蜕变测试没有公认的覆盖度量来提供有用的出口准则。仅覆盖每个蜕变关系一次是不充分的，因为只能获得对期望结果的部分验证。测试分析师可结合蜕变测试与随机测试，生成多个详细源测试用例和同一蜕变关系的后续测试用例。例如，在上述平均值计算中，可使用随机数生成器为源测试用例生成各种输入，以及随机的排列和乘数用于后续测试用例。

蜕变测试可用于大多数测试项，且适用于功能和非功能测试（例如，负载测试——利用蜕变规则设计后续测试用例生成负载；或易安装性测试——使用多种可按不同顺序选择的安装参数开展测试）。它是基于AI系统的首选测试技术（ISTQB-AI, v1.0）。

更多细节参见标准（ISO/IEC/IEEE 29119-4, 2021）或综述文章（Segura; Towey, et al., 2020）和（Segura; Fraser, et al., 2016）。

3.4 基于经验的测试

测试分析师采用基于经验的测试方法和技术，利用其专业知识和过往经验指导测试。本章详细描述测试分析师在基于会话的测试和基于检查表的测试中使用文档的情况（参见ISTQB-CTFL, v4.0.1，第

4.4.2节和第4.4.3节）。此外，还讨论了众测。所有基于经验的测试技术均可用于基于协作的测试，如验收测试驱动开发。更多细节参见（ISTQB-CTFL，v4.0.1，第4.5节）。

3.4.1 支持基于会话测试的测试章程

在探索性测试中，测试章程为测试会话提供任务，概述其范围、目标以及限制、时间线和风险等信息。它作为测试的路线图，为测试会话提供结构。测试章程帮助测试分析师专注于特定测试区域或功能，同时允许测试分析师在需要时灵活地探索系统。测试章程不指定每个测试会话中执行的测试套件。

在为基于会话的测试准备测试章程时，测试分析师需考虑影响测试章程设计的特定因素，特别是：

- 客户和需求因素（例如，从客户引出的需求、系统的业务用例、质量要求），以及用户旅程图（即用户随时间与系统的交互）。
- 产品因素（例如，功能流程、产品主要目标、产品功能、软件设计和接口）。
- 项目管理因素（例如，时间约束、项目目的、估算工作量和业务价值）。

测试章程包括描述测试目的的任务以及各种附加信息。

一个流行的轻量级任务格式为“探索[目标]使用[资源]以发现[信息]”（Hendrickson，2013），其中：[目标]：描述要探索的内容（例如，区域、特征、风险、组件和需求）。[资源]：描述测试分析师将使用的资源（例如，测试数据、配置、工具、限制、启发式方法和依赖关系）。[信息]：说明测试分析师旨在发现的信息类型（例如，质量特性评估、预期缺陷类型和违反标准的情况）。

测试章程可能包含但不限于以下附加信息（Ghazi et al.,2017）：

- 组织信息：例如，测试会话的持续时间、开始日期和时间、测试人员姓名。
- 测试目的：例如，测试的动机和测试章程的任务。
- 测试范围：例如，被测系统内的特定关注区域、测试级别、使用的测试技术、测试思路、出口准则、优先级、测试章程应覆盖的内容以及对不测试内容的描述。
- 入口准则：即开始测试会话必须满足的前置条件。
- 产品相关信息：例如，定义、数据、组件间工作流程和系统架构。
- 限制：即产品绝对不可做的内容。
- 测试环境描述。

- 有助于测试的现有数据源、产品信息和测试工具。
- 历史信息：例如，以前发现的缺陷（如兼容性和互操作性缺陷）、当前未解决的异常问题以及过去的测试相关失效模式。
- 约束和风险：例如，使用的法规、规则和标准。

在测试会话期间，测试分析师记录测试日志，包括问题、观察结果或未来测试的思路以及测试结果。这些数据记录在会话表中。

特定测试章程中包含的信息范围和细节可能不同，影响测试分析师的灵活性。例如，仅定义一般测试目的为探索提供充足空间，而添加要用的测试技术信息可能限制测试分析师。另一方面，添加信息（如系统绝对不可做的事情）可有助于降低报告假阳性结果的可能性。

3.4.2 支持基于经验的测试技术检查表

基于检查表的测试因其适应性、简单性和确保软件质量的有效性，是一种被广泛使用的测试技术。通过使用检查表，测试分析师确保覆盖测试项所有已知的重要方面，避免遗漏关键区域。检查表还能在不同测试周期和不同测试分析师之间引入一致性。使用检查表时，测试分析师专注于重要方面。检查表记录了测试分析师过去的失效和缺陷经验，作为提醒或灵感来源（例如，在探索性测试中思路耗尽时）。通过重用标准检查表或同行创建的检查表，测试分析师可节省时间，但前提是这些检查表与测试项相关。检查表有助于减少文档化测试用例的工作量，这在需求和软件不断变化时是一大优势。

准备检查表通常是测试分析师的责任。检查表支持基于经验的测试，有助于组织、结构化和指导测试。创建可重用、可维护、清晰且高效的检查表需要努力。

以下步骤可作为创建基于经验测试的适当检查表的指南：

首先，测试分析师确定检查表的范围、目标和格式，因为这些影响测试深度和所需的详细程度。读-做检查表包含特定流程需考虑的主要元素（例如，检查表项包含需检查的输入字段的具体无效输入）。做-确认检查表作为指导思考过程的辅助工具，提供基于经验的测试思路以进一步探索应用（例如，检查表项可能要求检查搜索结果是否相关）。

接下来，测试分析师收集定义检查表项所需的信息。这包括从经验丰富的专业人员获取见解、浏览缺陷库和缺陷分类（Beizer, 1990; Kaner; Falk, et al., 1999）、评审相关文档以及分析风险、测试用例和潜在场景。检查表项应清晰、具体、无歧义、一致、相关、可维护、可操作和可衡量。应以可回答“是”、“否”或“不适用”的问题形式表述。需根据重要性、潜在影响和风险级别分配优先级。检

查表不是全面的操作指南，而是供专家使用的快速、简单工具。

最后，测试分析师基于功能区域、用户角色、测试级别或其他相关准则，将检查表项归类为逻辑组，从而对检查表进行结构化整理和组织。对于长检查表，创建单独类别尤为有用。

尽可能使用模板和标准。测试分析师可通过利用符合行业标准和最佳实践的现有模板或预定义检查表节省工作量。

检查表永远不会最终定稿。测试分析师持续评审和优化检查表，根据新发现、优先级的变化、其他测试人员的反馈或从先前测试周期中获得的经验教训进行调整。通过与其他测试人员共享检查表，测试分析师促进一致性和协作，帮助他们更好地理解测试项和测试期间需关注的重点领域。

3.4.3 众测

众测将测试分发给来自不同背景和地点的内部或外部测试人员群体。它是确认易用性的成本效益高的方法，覆盖功能和非功能质量特性（Alyahya, 2020; Leicht et al., 2017）。

众测的收益包括：

- **多样化的测试环境：**测试人员可位于不同地理位置，使用各种设备、浏览器和网络条件的多种环境配置。
- **更高灵活性：**易于扩展，以在短时间内处理大量测试。
- **成本效益：**通常比维护大型、多样化的内部测试团队或补充外部测试服务成本更低。
- **快速反馈：**测试人员可提供快速反馈，帮助早期识别和修复失效。
- **真实用户视角：**测试人员可能是应用程序的实际用户，能更好地提供用户体验和易用性的见解。这在用户验收测试中尤为有价值。
- **多样性：**每次测试由不同测试人员执行，测试不可重复。虽然这可能是一个局限性，但也带来更广的覆盖范围，增加发现缺陷的机会。

众测的局限性包括：

- **测试质量不可靠：**测试质量因个体测试人员的技能而异，尽管在目标是用户体验反馈时可能无关紧要。
- **沟通挑战：**协调来自不同地点、时区、文化差异和语言障碍的众多测试人员可能具有挑

战性。

- **安全性：**与外部测试人员共享软件存在数据安全和保密风险。采取适当措施可缓解这些风险，从而既能够负责任地开展众测，又不会泄露敏感细节或助长抄袭行为。
- **文档和报告：**在与大量且背景多样化的测试人员合作时，确保全面的测试文档和管理大量发现（包括重复和误报）可能具有挑战性。

众测是一种不会替代测试分析师应用测试技术的方法，但可增加多样化测试环境的覆盖范围。

3.5 应用最合适的测试技术

测试应在给定场景下尽可能有效和高效。为此，测试分析师支持测试经理选择最合适的测试技术。此外，测试分析师可使用自动化支持测试活动，包括自动化测试设计（本节描述）和支持测试执行自动化（参见第*1.3.6*节）。

3.5.1 选择测试技术以缓解产品风险

选择最合适的测试技术对有效和高效缓解产品风险至关重要，受多种因素影响，包括以下内容：

测试目的（参见ISTQB-CTFL，v4.0.1，第1.1.1节）：指定评估测试对象的哪些方面，指导测试技术的选择，取决于系统类型（例如，工程中数值计算的域测试与信用风险管理中的判定表测试）。

产品风险与潜在缺陷相关。特定测试技术最适合检测这些缺陷，因为大多数测试技术专注于检测特定类型的缺陷（参见本大纲第3.1节、第3.2节、第3.3节和第3.4节）。例如：

- 基于数据的测试技术可检测数据处理、域实现、用户界面、计算和参数组合中的缺陷。
- 基于行为的测试技术可检测用户需求缺陷，如缺失功能、通信缺陷和处理缺陷。
- 基于规则的测试技术可检测逻辑和控制流中的缺陷。

风险分析还有助于确定适当的测试方法，例如：

- 基于覆盖的出口准则：风险级别越高，可能需要更严格的覆盖（例如，组合测试中的全组合覆盖而非成对覆盖）。但测试分析师必须始终考虑覆盖强度与所需测试工作量之间的权衡。
- 基于经验的测试可在难以定义覆盖、风险级别低或项目进度紧张时使用。

测试依据。如果测试对象的规格说明使用模型，测试分析师可使用基于这些模型的测试技术。如果从测试依据无法或难以推导测试结果参照物，可应用蜕变测试或基于经验的测试之类的测试技术。

反复出现的缺陷类型的知识：可能提示选择专注于检测此类缺陷的测试技术（例如，基于检查表的测试）。如果预期发现与之前迭代或项目中类似的缺陷，使用先前成功的测试技术可能是合理的。

测试分析师的知识和经验。如果测试分析师不熟悉某一特定测试技术，不建议在关键测试任务中使用该技术。领域知识也可能影响测试技术的选择。例如，缺乏领域知识或完全没有，表明探索性测试之类的测试技术可能效果不佳。

所使用的软件开发生存周期。顺序开发模型适合使用更正式的技术。相反，迭代开发模型可能更适合采用轻量级测试技术（例如，基于经验的测试技术）或在自动化测试设计的情况下使用。

客户和合同要求。合同可能明确要求执行特定测试（例如，特定测试级别或测试类型），这会影响测试技术的选择（例如，客户提供的带场景集的验收准则建议使用基于场景的测试技术）。

法规要求。当项目遵循标准时，可能要求使用特定测试技术。例如，标准（ISO 26262, 2018）根据分配给测试对象的汽车安全完整性级别（ASIL），要求使用等价类划分、边界值分析或错误猜测等测试技术。

项目约束，如时间和预算，可能影响耗时技术或需要昂贵资源技术的使用。

测试技术通常组合使用以提高缺陷检测的效率和有效性。例如：

- 边界值分析（BVA）可用于状态转移测试中的守卫条件。
- 域测试可用于基于场景的测试，以确定判定表中某个条件的值或被测系统中某个变量的值。
- 基于场景的测试可与判定覆盖（ISTQB-TTA, v4.0中讨论的一种白盒测试技术）结合，以严格覆盖业务流程中的判定。例如，参见（Koomen et al.,2006）中的流程循环测试。
- 基于场景的测试可与环路覆盖结合，解决循环业务流程活动的特定风险。

3.5.2 自动化测试设计的收益和风险

测试分析师可使用工具应用测试技术，特别是黑盒测试技术。在自动化测试设计时，测试分析师创建测试模型并从中自动生成测试件。例如，测试分析师设计状态转移模型，并让基于模型的测试工具为环路覆盖生成测试用例。

自动化测试设计通常提高测试的效率和有效性。其收益包括：

- **缺陷预防。**为测试建模是评估测试依据质量的有效方法（参见第5.2.1节）。
- **扩展能力。**自动化支持应用更复杂的测试技术和覆盖准则，如组合测试、随机测试或N-切换覆盖，降低未测试代码的风险。
- **提高可理解性。**工具中指定的测试选择准则更清晰直观地指向测试条件，并更易于理解地证明所生成覆盖的合理性。
- **减少重复工作。**测试件可从测试模型生成，减少手动指定测试等重复工作。
- **减少维护工作。**测试模型是导出测试件的单一事实来源，因此只需维护测试模型。
- **动工作易出错。**工具生成的测试件具有更高的质量和一致性。
- **增强团队协作。**利益相关方可评审测试模型以发现缺陷或更好地理解测试条件。
- **增强可追溯性。**将测试模型的元素链接到测试条件比链接测试用例本身更容易。如果工具支持，生成的测试用例将继承这些链接，提高测试整体可追溯性。
- **多样化输出格式。**测试件可根据其他工具和后续活动的要求生成不同输出格式。

测试分析师还必须考虑自动化测试设计的风险，包括：忽略模型中未显示的测试条件,低估测试模型的维护工作量，利益相关方难以理解模型以及测试自动化的通用风险（参见ISTQB-CTFL，v4.0.1，第6.2节）。

4. 测试质量特性 - 60 分钟

关键词

适应性（adaptability）、兼容性（compatibility）、灵活性（flexibility）、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）、功能测试（functional testing）、易安装性（installability）、交互能力（interaction capability）、互操作性（interoperability）、易用性（usability）、用户体验（user experience）

第4章的学习目标

4.1 功能测试

TA-4.1.1 (K2) 区分功能正确性、功能适合性和功能完备性测试。

4.2 易用性测试

TA-4.2.1 (K2) 解释测试分析师如何为易用性测试做出贡献。

4.3 灵活性测试

TA-4.3.1 (K2) 解释测试分析师如何为适应性和易安装性测试做出贡献。

4.4 兼容性测试

TA-4.4.1 (K2) 解释测试分析师如何为互操作性测试做出贡献。

质量特性测试简介

本教学大纲以 ISO 25010（ISO/IEC 25010，2023）中提供的软件质量模型为指导，并论述了测试分析师应关注的质量特性。然而，所使用的易用性术语与该标准存在差异，是为了与《易用性测试大纲》（ISTQB-UT，v 1.0 ）保持一致。

附录 F 对现行 ISO 25010 标准的质量模型、与上一版本相比的差异，以及侧重于测试特定质量特性的相关ISTQB® 教学大纲进行了概述。

4.1 功能测试

功能测试是测试分析师的核心任务之一。（ISTQB-CTFL，v 4.0.1 ）仅将功能测试简要描述为一种测试类型，而本教学大纲进行了更详细的阐述，论述了功能性的子特性。

4.1.1 功能性的子特性

ISO 25010（ISO/IEC 25010，2023）中的产品质量模型将功能性分为三个不同的子特性，尽管这些子特性都可以通过功能测试来进行评估，但并非每一项功能测试活动都能同样有效地涵盖这些方面。测试分析师应当能够选择合适的测试级别和测试技术，以应对功能性具体子特性相关的产品风险。

功能完备性测试应涵盖软件的所有的指定任务以及预期用户目标，核心在于是否所有要求的内容都得到了实现。

功能完备性方面应尽早予以关注，在顺序开发模型中，可通过评审需求规格说明来实现这一目标，在敏捷软件开发中，通过在协作编写用户故事的过程中讨论用户故事（包括验收准则）也能达到这一目标。在系统测试、系统集成测试以及验收测试中，功能完备性可以通过动态方式进行测试。

基于行为的测试技术（如基于场景的测试）非常适用，不过其他黑盒测试技术也同样合适。在确定功能完备性的达成程度时，测试依据、测试条件与测试用例之间的可追溯性至关重要。

功能正确性测试 旨在回答这样一个问题：对于有效输入和无效输入，实际结果是否正确（例如，是否准确、精确且一致），找到一个能详细提供期望结果的测试结果参照物至关重要。

功能正确性可以在任何测试级别进行测试，就左移而言，大多数功能正确性测试应在组件测试和组件集成测试中进行。即使测试分析师并不负责这些测试级别，仍应积极参与其中，以最大程度地实现测试目的。

所有黑盒测试技术、基于经验的测试技术以及基于协作的测试均适用。

功能适合性测试验证功能是否有助于完成指定的任务和目标。其重点在于检查所实现的所有内容是否都能满足用户的需求。

功能适合性测试可能包括用户界面设计评审，特别是交互式应用程序。在顺序开发模型中，动态测试始于系统测试和验收测试；在敏捷软件开发中，动态测试始于演示环节。

探索性测试和基于协作的测试是最合适的，此外，基于行为的黑盒测试技术也同样合适。

4.2 易用性测试

易用性是指与用户相关的各种质量特性的宽泛概念，涵盖产品质量模型（*ISO/IEC 25010，2023*）中的交互能力以及 ISO 25019 使用质量模型（*ISO/IEC 25019，2023*）中的有益性，更多关于易用性测试的内容可参见（ISTQB-UT，v1.0),、(*ISO 9241-210,*2019)和（UXQB-FL，v4.01)。

4.2.1 测试分析师对易用性测试的贡献

易用性测试通常会侧重于评估以下方面：

- **交互能力** - 使用户能够在特定的使用周境中有效地、高效地且令人满意地完成任务（ISO/IEC 25010，2023）
- **用户体验** - 在用户与测试对象进行交互之前、交互之中以及之后，了解用户的各种看法和感受
- **易访问性** - 确保残障用户、具有不同文化背景的用户或者存在语言障碍的用户群体能够以有效且高效的方式使用系统

测试分析师可以在早期阶段参与易用性测试的工作，借助其对目标用户群体、用户目标、使用周境、在使用系统时可能遇到的困难以及负面用户体验的了解来开展协作。

测试分析师能够为主要的易用性评估技术做出如下贡献：

- **易用性评审**由易用性专家进行，目的是识别潜在的易用性问题以及与既定准则偏差的情况。易用性的评审形式可以从非正式评审到审查。测试分析师可以根据用户群体的具体需求、特定的业务目标、优先级以及使用周境（例如，定制通用的易用性检查表）来调整评审准则。

- **易用性测试**会话涉及未来目标用户或其代表尝试完成预先设定的任务，以评估这些任务是否能以有效、高效且令人满意的方式完成。测试分析师可以根据人物角色、用户群体或运行配置来设计易用性测试会话的场景。
- **用户问卷或用户调查**（包括评分和反馈）用于衡量用户满意度。用户问卷的例子有 SUMI（软件易用性度量调查表，1991 ）和 WAMMI（网站分析和度量调查表，1999 年）。测试分析师可以协助设计问卷并评估回答情况，以实现目标用户在使用周境中的特定目标。

在易访问性测试中，常见的测试目的之一是验证是否符合标准。国际标准的《网络内容无障碍性指南》（WCAG，2023 ）将符合程度分为三个级别（A 级、AA 级和 AAA 级），这些级别代表了网络内容易访问性程度的逐步提高。国家标准包括英国的《平等法》（英国政府，2010 ）以及美国的《美国残疾人法案》（美国司法部，2010 ）。测试分析师可以通过对使用周境的分析，确定所需的合规级别以及预期目标群体的具体需求

4.3 灵活性测试

灵活性测试（也称为可移植性测试）验证测试对象是否能够适应其使用周境或系统环境的变化。ISO 25010 产品质量模型（ISO/IEC 25010，2023）对灵活性的以下子特性进行了区分：

- 适应性
- 可扩展性
- 易安装性
- 易替换性

灵活性既包含技术层面的内容，也包含非技术层面的内容。本章节将重点介绍测试分析师如何为适应性和易安装性测试做出贡献。可扩展性和易替换性与技术层面有关，相关内容分别在（ISTQB-PT，v1.0）和（ISTQB-TTA，v4.0),）中有详细阐述。

4.3.1 测试分析师对适应性测试和易安装性测试的贡献

适应性测试验证测试对象是否能够适配于或迁移至预期的目标硬件、软件或其他运行或使用环境。

测试分析师通过识别预期的目标环境（例如，所支持的移动端操作系统版本以及可能使用的浏览器版本）并设计涵盖这些环境组合的测试，来支持适应性测试。由于这需要能够代表各种环境参数配置的测试

数据进行，因此通常会采用组合测试等测试技术（详见 3.1.2）。另一个例子是将各种平台组件整合到客户项目中，以确保在跨目标环境的兼容性。根据产品风险情况，测试分析师会设计和执行冒烟测试或更全面的测试套件，以验证测试对象是否已正确适配了目标环境。

通过遵循适应性测试的良好实践（例如，尽早定义目标环境、采用组合测试、在新环境中进行冒烟测试，以及监控环境特定的缺陷），测试分析师能够发现那些可能限制软件生存周期和易用性（例如在不同屏幕尺寸上的易用性）的缺陷。测试分析师在适应性测试方面的工作还应得到由测试自动化工程师实施的跨平台自动化测试支持。严格的适应性测试可以及早发现环境特定的问题，从而有助于避免在部署后频繁进行重大更新或重新设计，并降低维护成本。

易安装性测试验证测试对象能够在指定环境中正确地进行安装、卸载、更新和重新配置。易安装性测试不局限于检查安装过程是否能够顺利完成。

测试分析师所关注的典型易安装性测试目的为：

- 验证在各种环境参数配置下安装程序的执行是否正确，这使得诸如组合测试之类的测试技术变得很有用，与适应性测试类似。
- 设计并执行测试，以确定测试对象在安装或更新后是否能正常运行
- 检查用户安装、卸载或更新软件的难易程度，这包括评审安装说明文档。
- 测试权限相关的行为，尤其是针对移动应用（详见ISTQB-MAT，v1.0）。

4.4 兼容性测试

兼容性测试用于验证测试对象在使用时与其他组件或系统是否兼容。ISO 25010 产品质量模型（ISO/IEC 25010, 2023）将兼容性的两个子特性进行了区分：互操作性和共存性。因此，兼容性测试可以细分为以下几种测试类型：

- 互操作性测试，用于验证测试对象与预期将发生交互的组件或系统之间的兼容性。这类测试通常属于黑盒功能测试范畴，因此测试分析师通常会负责这些测试工作。
- 共存性测试，用于验证测试对象能否与其他组件或系统共享其目标环境而不会产生干扰。这种技术测试类型已在《技术测试分析师大纲》（ISTQB-TTA，第 4 版）中有所论述。

4.4.1 测试分析师对互操作性测试的贡献

互操作性测试的目标是验证两个或多个组件或系统能够交换信息，并能够相互使用已交换的信息。如果信息交换涉及数据转换，那么互操作性测试就必须包括对这种转换的验证。

当多个系统需要协同工作、共享数据或共同执行任务时，互操作性测试就显得尤为重要。这种情况在现代软件架构中尤为常见，比如云解决方案、Web 服务、微服务、容器化以及物联网等。

互操作性通常发生在不同的架构层面。测试分析师必须了解可能存在的交互方式，以便定义适当的测试条件来涵盖这些交互。并非所有的交互都可能被记录下来。测试分析师可能从架构和设计文档中间接获取有关交互的信息。因此，理解这些文档至关重要，以确保将所有重要的交互方面都进行测试。

互操作性测试能够发现以下方面的缺陷：

- 对于数据交换过程中的数据转换。
- 已交换数据的解析或使用。
- 通信流和协议。
- 对标准的符合性。
- 端到端功能。
- 设计文档。

互操作性测试通常在集成测试阶段进行。黑盒测试技术（例如侧重于已交换数据的基于数据的测试技术、用于解析或使用数据的基于行为的测试技术、以及用于数据转换的端到端功能或基于规则的测试技术），都非常适合用于互操作性测试。基于经验的测试技术可以有效地补充黑盒测试。来自功能测试或适应性测试的测试用例可以复用于互操作性测试。

互操作性的通用标准示例包括（*ISO 15745*，2003）和（*ISO 16100*，2009）。ETSI（*ETSI EG 202 237 v1.2.1*,2010）在电信领域提供了一个具体的互操作性测试方法的示例。

5. 软件缺陷预防 - 225 分钟

关键词

临时评审（ad hoc reviewing），基于检查表的评审（checklist-based reviewing），缺陷预防（defect prevention），基于模型的测试（model-based testing），基于阅读视角的文档评审（perspective-based reading），评审技术（ review technique），基于角色的评审（role-based reviewing），根本原因分析（root cause analysis），基于场景的评审（scenario-based reviewing），测试结果（test result）

第五章的学习目标:

5.1 缺陷预防实践

TA-5.1.1 (K2) 解释测试分析师如何为缺陷预防做出贡献。

5.2 支持阶段遏制

TA-5.2.1 (K3) 使用测试对象的模型来检测规格说明中的缺陷。

TA-5.2.2 (K3) 应用评审技术从测试依据中发现缺陷。

5.3 减少缺陷再次发生

TA-5.3.1 (K4) 分析测试结果以识别缺陷检测方面的潜在改进措施。

TA-5.3.2 (K2) 解释缺陷分类如何支持根本原因分析。

软件缺陷预防简介

缺陷预防的目标在于采取措施降低工作产品中发生（或再次发生）缺陷的可能性，并减少缺陷向软件开发生存周期后续阶段的传播。这些方面的努力带来了重要的实际价值，包括成本和人力的降低、生产效率的提高以及产品质量的提升。缺陷预防工作是整个团队的责任。测试分析师可以利用其特定的知识和经验来为此做出贡献。

缺陷预防实践包括：

- 防止缺陷引入，这是质量保证活动的一部分
- 防止缺陷在软件开发生存周期的后续阶段中传播（详见 *5.2*）
- 防止缺陷再次发生（详见*5.3*）

5.1 缺陷预防实践

5.1.1 测试分析师对缺陷预防的贡献

测试分析师可以通过多种方式为缺陷预防工作做出贡献，他们可以凭借自身的领域知识、测试技能以及分析能力来实现这一目标。示例包括：

- 参与风险分析 - 确保已识别的风险得到妥善缓解（例如，选择最合适的测试技术）。
- 评审需求、模型和规格说明 - 能够尽早发现测试依据中的缺陷，防止这些缺陷被编入代码，从而显著降低修复它们所需的成本。
- 参与回顾会议 - 从而能够识别出在测试分析、测试设计、测试实施以及测试执行方面可能存在的改进之处（例如，采用更有效的测试技术、针对特定风险领域进行测试、或者改进测试数据和测试环境以减少假阳性结果和假阴性结果），从而更好地防止遗漏缺陷。
- 缺陷数据收集与评估 - 通过收集有关缺陷的详细数据，实现缺陷的分类和统计分析，从而便于根本原因分析，进而支持根本原因分析和过程改进。
- 参与根本原因分析——通过提出纠正措施来解决已识别的根本原因，从而防止缺陷再次发生。

除了参与缺陷预防工作之外，测试分析师还会（通常会与测试经理共同商讨）评估所提出的措施是否

达到了预期效果。有助于评估这些措施效果的度量示例包括：

- 缺陷清除率(DRE) - 该指标衡量的是在产品发布前被消除的缺陷数量与总缺陷数量的比率。该比率的分母（未知值）可以通过估算或用截至某一特定时间点（例如，版本发布后6个月）所发现的总缺陷数量来替代。高DRE意味着有较少的缺陷逃入了生产环节，这可能意味着有较好的缺陷预防实践。然而，DRE并未区分是通过预防措施还是通过检测发现的缺陷。
- 阶段遏制有效性（PCE）——衡量同一阶段引入和消除的缺陷数量与该阶段内引入的总缺陷数量之间的比例关系。高PCE表明有较少的缺陷会进入后续阶段。
- 质量成本——阐述了缺陷预防、缺陷检测以及缺陷清除成本之间的关系（参见ISTQB-TM，v3，第3.2.1节）。

5.2 支持阶段遏制

阶段遏制的目标是在缺陷引入的软件开发生存周期的同一阶段就将其检测并清除掉。在敏捷软件开发中，可以采用类似的“左移”方法。

该策略降低了质量成本，由于测试依据是进行测试分析和测试设计的重要输入，测试分析师通过评估测试依据的质量，能够最有效地为阶段遏制做出贡献。

尽早关注测试依据的质量，能够最大程度地减少后续工作量，并防止缺陷向后续阶段传播。本大纲探讨了测试分析师在发现测试依据中的缺陷时常见的两种方法：为测试目的进行建模以及运用各种评审技术（*ISO/IEC 20246*，2017）对测试依据进行评审。

5.2.1 使用模型检测缺陷

建模以一定的精度和细节程度对系统提供抽象的展示，它有助于利益相关方更好地理解正在开发的系统。如下文所述，建模至少可以通过三种方式支持阶段遏制：

- 检测规格说明中的缺陷
- 检测模型中的缺陷
- 通过基于模型的测试来检测测试对象中的缺陷

检测规格说明中的缺陷，规格说明通常是以非正式的书面形式给出的。测试分析师可以使用模型来

正式表述这些规格说明。在创建模型时，测试条件（例如需求）会被映射到模型元素，并与这些元素建立关联以实现可追溯性。在模型中对测试条件进行形式化和可视化能够有效地揭示诸如不完整、不一致或有歧义等问题。建模的优势在于，对规格说明进行转换的同时，评审仍然能够对其原始形式进行检查。因此，测试分析师还能协助找到针对所发现缺陷的恰当解决方案。

基于数据的模型包括域模型和组合测试模型，这些模型使测试分析师能够检测出域缺陷，例如重叠分区、域覆盖范围的缺失、空分区，以及参数组合缺失或不正确等问题。

基于行为的模型包括 CRUD 矩阵，基于状态的模型或场景模型。它们使测试分析师能够检测到不完整或不一致的实体生存周期、缺失或错误的状态转移、死锁、无限循环、模糊不清或不一致的系统行为，以及缺失的异常处理。

基于规则的模型，如判定表或者蜕变关系，使测试分析师能够检测业务规则中的缺陷，如遗漏、不一致、歧义、冗余、易出错的场景或复杂的业务逻辑。

建模还能检测出诸如命名不一致、数据值（例如边界值）以及输入或输出方面的缺陷。它还能识别出缺失、不完整、有歧义或不必要的信息。

检测模型中的缺陷。如果测试依据中包含模型，测试分析师可以对这些模型进行分析以检测缺陷。本大纲讨论了软件测试中三种典型模型的缺陷检测相关内容：状态转移图（详见第 3.2.2节）、活动图（详见第 3.2.3节 ）和判定表（详见第3.3.1节 ）。上述模型中缺陷的示例包括：

状态转移图 - 缺失/错误的状态、不恰当的转移、错误的守卫条件或动作、冗余或不可到达的状态，以及非确定性行为。

活动图 - 缺失的、无法执行或无效的动作、动作顺序错误、决策节点中的错误、非互斥或不完整的守卫条件、缺少同步点，或者未同步的并行流程，这些都可能导致意外行为。

判定表 - 规则重叠、不一致或不可行，以及不完整（例如，缺少条件的组合，或者对于给定的条件组合缺少相应的动作）。

所有模型也可能存在缺陷，例如语法错误、拼写错误、重复项以及模型元素命名不一致等问题。

通过使用基于模型的测试（MBT）来检测缺陷。在这种测试方法中，MBT工具会根据测试分析师定义的 MBT 模型和测试选择准则，来设计并生成测试用例以及其他测试设计测试件。MBT在发现规格说明中的异常方面非常有效，因为它能够根据MBT模型对测试对象的预期行为进行全面覆盖和系统的探索。MBT 模型，尤其是图形化的模型，有助于与利益相关方进行沟通，建立对测试依据的共同认知和理解。有关 MBT 的更多详情，请参阅《基于模型的测试工程师大纲》（ISTQB-MBT，v1.1）。

5.2.2 应用评审技术

定义完善的测试依据是确保工作产品质量以及项目成功的关键基础。对测试依据的评审有助于尽早发现并解决缺陷，从而防止缺陷遗漏到后续阶段。

在个人评审时，测试分析师可以采用多种评审技术来识别测试依据中的缺陷。选择最合适的评审技术能够提高评审的效率和有效性。这一选择应考虑诸如评审目标、项目目标、可用资源、测试依据类型、相关风险、业务领域以及公司文化等因素。

下面，本教学大纲将讨论测试分析师常用的五种评审技术。

临时评审是由评审员以非正式的方式进行的，不基于结构化的过程。评审员几乎得不到关于如何完成这项任务的指导。临时评审所需的准备工作很少，而且很大程度上取决于评审员的技能水平。在评审过程中，评审员会阅读测试依据，并记录遇到的异常情况。如果这种技术未得到妥善管理，可能会导致多个评审员提交大量重复的异常报告。

基于检查表的评审是指将测试依据与预先定义的检查表进行对照评估。检查表能够提醒评审员检查特定的要点，并且能使评审去个人化。检查表可以是通用的，也可以针对质量特性、测试目的或测试依据类型。测试分析师会根据测试依据类型、风险级别或测试条件来定制检查表。这样可以确保评审聚焦于测试依据的最相关方面。检查表应定期更新，以纳入之前遗漏的缺陷。保持检查表的更新可以防止遗漏新发现的异常情况。检查表并非包含全部内容。因此，测试分析师并不局限于检查列出的项目。这种做法最大限度地提高了缺陷检测能力，使得测试分析师能够捕捉到检查表可能未明确涵盖的异常情况。

基于场景的评审涉及模拟某个过程或活动，以发现异常情况并细化测试依据。当测试依据采用基于场景的格式，例如用例或活动图，这种评审技术最为有效。在这种情况下，评审员可以根据工作产品的预期使用情况进行“演练”。场景提供了有价值的指导原则，但评审员不应仅限于依据已记录的场景进行评估，还应当跳出这些场景的限制，去发现更多的异常情况。

基于角色的评审包括为评审员分配特定的角色或职责。常见的角色依据具体的终端用户类型（例如，经验丰富的、缺乏经验的、年长的或儿童用户）或组织内的特定角色（例如，管理员或普通用户）来设定。每个角色都可以通过一个角色设定来加以描述（即，为代表特定用户群体的特征、需求、目标和偏好而设计的具体但虚构的人物）。根据评审员的角色来分配评审任务，基于角色的评审使得个人能够专注于测试依据的特定方面，从而确保全面的覆盖，同时避免异常情况的重复出现。

基于阅读视角的文档评审涉及从多个角度或视角（例如设计人员、测试人员、营销人员、管理员和最终用户）来评审测试依据。这使得个人评审更加深入，同时减少了不同评审员之间对相同问题的重复评

审。此外，基于阅读视角的文档评审要求评审员尝试利用正在评审的测试依据，来生成他们从中可以推导出的工作产品。例如，测试人员会尝试根据需求规格说明书生成验收测试草案，以查看是否包含了所有必要的信息。

5.3 减少缺陷再次发生

测试分析师能够主动采取措施帮助减少软件中缺陷的再次发生。本大纲探讨了两种与减少缺陷再次发生相关的方法：通过分析测试结果来改进测试分析和测试设计，以及利用缺陷分类来支持根本原因分析。

5.3.1 分析测试结果以提高缺陷检测能力

测试结果能够帮助测试分析师识别失效，同时也能为测试分析师提供反馈，以帮助提高缺陷检测的有效性。下面将介绍一些用于分析测试结果的常用技术。

预测与实际缺陷集群分析。通常，少数组件会包含大部分缺陷（详见 ISTQB-CTFL，v 4.0.1 ，第 1.3 节）。在测试完成后，测试分析师可以预测缺陷高发区域，并将预测的缺陷集群与实际的缺陷集群进行比较。如果存在差异，可以对发现缺陷多于预期的区域应用更严格的测试。在确定集群时，应采用可衡量的准则以确保清晰易懂和一致性（例如，缺陷密度和缺陷严重程度）。在这些准则中，缺陷的严重程度应起到关键作用。小部分关键或重大的缺陷，通常比大部分轻微或不重要的缺陷更为重要（即需要更严格的测试）。

缺陷检测率（DDP）分析是衡量测试级别有效性的重要指标之一。 计算DDP 时，应将遗漏缺陷的数量限制在该测试级别本可检测到的范围内。应为缺陷计数设定明确的界限，像时间限制（例如，在发布后规定的时间范围内发现的缺陷）和排除标准（例如，第三方组件中的缺陷或特定客户环境中的缺陷），以确保一致性。对于给定的测试级别，如果 DDP 较低，则表明遗漏的缺陷比例较高，这意味着缺陷检测效果不佳。在这种情况下，测试分析师应分析原因并提出措施加以改进，使测试级别更加聚焦和严谨。DDP 最好按严重程度级别进行划分，因为减少遗漏的缺陷的优先级通常取决于其严重程度。

结构覆盖分析是指评估测试对测试对象特定区域的覆盖程度。找出覆盖程度低的区域有助于测试分析师将测试工作量放在这些区域上。扩大覆盖范围有助于发现新的缺陷以及此前遗漏的缺陷。诸如语句覆盖、分支覆盖或神经元覆盖率等结构覆盖，通常通过测试工具进行度量。在选取需要覆盖的额外区域时，应考虑它们的风险级别。

测试差距分析是指评估测试在多大程度上覆盖了近期的代码变更。这使得测试分析师能够将更多的

测试工作量集中在那些特别容易出错的区域（即那些从未进行过测试的新变更），而非集中于所有覆盖率较低的区域（例如那些长时间未更改且在之前版本中已进行过测试的代码）。

缺陷到达模式分析。在一个项目（例如迭代项目）的不同连续阶段中发现的缺陷数量或密度，可以将其与描述这些值随时间分布的理论模式进行比较。缺陷到达模式的一个典型例子是瑞利模型Rayleigh model (Elsayed, 2021)。该模型有一个单峰且向右偏斜，表明预期发现的缺陷数量首先随着时间的推移而增加，并在达到最大值后缓慢下降至零。基于对这种模式的分析，可以推断出现有测试用例的强度以及它们改进的可能性。例如，假设缺陷检测保持在恒定的低水平，而该模式表明本应该出现增长，这可能意味着现有的测试力度不够，无法发现更多的缺陷。

上述的分析方法使用了各种与缺陷相关的指标，例如缺陷数量或缺陷检测率（DDP）。这些指标是根据测试结果（例如测试通过/不通过的数量）、缺陷报告以及结构指标（例如代码覆盖率或缺陷密度）计算得出的。然而，需要注意的是，这些指标并非总是像看起来那样容易从测试结果中读取出来。例如，不通过的测试用例数量并不一定与这些测试所检测到的缺陷数量相同。要计算实际检测到的缺陷数量，必须仔细分析调试过程的结果，因为测试结果与缺陷之间可能存在多对多的关系。多项测试可能检测到同一个缺陷，或者一项测试可能检测到多个缺陷。此外，缺陷的严重程度可能与测试的关键程度不同，因为一个关键的测试用例可能会由于一个不重要的陷而不通过。

5.3.2 通过缺陷分类来支持根本原因分析

根本原因分析（RCA）是一种用于识别找出并解决缺陷的深层原因或根本原因，而非仅仅关注其表象的技术。RCA 为质量改进提供了结构化的方法，并且其主要目标是防止缺陷的再次发生。测试分析师会使用多种技术来识别缺陷和失效的根本原因（例如，缺陷分类法、五问法、因果图和帕累托分析）。

传统的根本原因分析（RCA）方法是让领域专家在解决缺陷后，对其进行详细研究。然而，通常会有许多缺陷需要分析。因此，为每个缺陷制定预防措施计划会非常低效且耗时。解决这个问题的一种方法是先对缺陷进行分类，然后针对出现的缺陷类型进行根本原因分析。

缺陷分类基于如下认知：单个缺陷能捕捉到大量关于开发过程以及被测试系统的信息。缺陷分类使测试分析师能够从缺陷中提取有关开发过程的各类信息，并将其转化为过程指标。这进而有助于深入了解开发过程中所犯错误的类型，有助于过程改进。缺陷分类能够将定量的缺陷统计与定性的根本原因分析有效衔接起来。为了有效支持根本原因分析（RCA），缺陷应在整个软件开发生存周期（SDLC）内，从早期测试到生产的各个阶段得到统一的分类。

测试分析师应当协助其所在的组织对软件缺陷分类进行标准化。这样做能够促进开发者之间以及与组

织之间关于缺陷的沟通和信息交流，从而有助于进行根本原因分析。

缺陷分类方法的示例有：

- 正交缺陷分类法（ODC）(Chillarege, 1992)，该方法将每个缺陷按照正交（即相互排斥）的属性进行分类，这种分类在缺陷报告时以及缺陷修复时都会进行收集。
- IEEE 1044（IEEE 1044，2009），对软件异常现象进行了标准分类，为失效和缺陷的分类提供了核心属性集。
- 基于严重程度分类，即根据缺陷的严重程度对其进行分类（例如，关键的、严重的、一般的、轻微的）。
- 缺陷分类模型，例如(Beizer, 1990)或(Catolino et al., 2019)。

还可以通过诸如（ISO/IEC 25010，2023）这样的软件质量模型或 FURPS 模型 (Grady et al., 1987)将缺陷映射到质量属性。

有关 RCA 的更多信息详见（ISTQB-ITP，v1.0 ）。

6. 参考文献

标准

ETSI EG 202 237 v1.2.1: Internet Protocol Testing (IPT), 2010. Standard. European Telecommunications Standards Institute.

IEEE 1044: Standard Classification for Software Anomalies, 2009. Standard. Institute of Electrical and Electronics Engineers.

ISO 15745: Industrial automation systems and integration - Open systems application integration framework, 2003. Standard. International Organization for Standardization.

ISO 16100: Industrial automation systems and integration - Manufacturing software capability profiling for interoperability, 2009. Standard. International Organization for Standardization.

ISO 26262: Road vehicles - functional safety - Part 6: Product development at the software level, 2018. Standard. International Organization for Standardization.

ISO 9241-210: Ergonomics of human-system interaction, part 210: Human-centred design for interactive systems, 2019. Standard. International Organization for Standardization.

ISO/IEC 20246: Software and systems engineering - Work product reviews, 2017. Standard. International Organization for Standardization.

ISO/IEC 25010: Systems and software Quality Requirements and Evaluation (SQuaRE) - Product quality model, 2023. Standard. International Organization for Standardization.

ISO/IEC 25019: Systems and software Quality Requirements and Evaluation (SQuaRE) - Quality-in-use model, 2023. Standard. International Organization for Standardization.

ISO/IEC/IEEE 29119-3: Software testing, Part 3: Test documentation, 2021. Standard. International Organization for Standardization.

ISO/IEC/IEEE 29119-4: Software testing, Part 4: Test techniques, 2021. Standard. International Organization for Standardization.

ISO/IEC/IEEE 29119-5: Software testing, Part 5: Keyword-Driven Testing, 2016. Standard. International Organization for Standardization.

OMG® BPMN: Business Process Model and Notation, version 2.0, 2013. Standard. Object Management Group, OMG®.

OMG® DMN: Decision Model and Notation, version 1.5, 2024. 2024-01. Standard. Object Management Group.

OMG® UML: Unified Modeling Language, version 2.5, 2017. Standard. Object Management Group.

RTCA DO-178C: Software Considerations in Airborne Systems and Equipment Certification, 2011. Standard. Radio Technical Commission for Aeronautics, RTCA Inc.

ISTQB®文档

ISTQB-ACT，认证测试工程师-验收测试大纲v2019_CN1.0（中文版），2023年。

ISTQB-AI，认证测试工程师-人工智能测试大纲 v1.0_CN1.1（中文版），2023年。

ISTQB-CTFL，认证测试工程师-基础级大纲v4.0_CN1.3（中文版），2024 年。

ISTQB-ITP，认证测试工程师专家级大纲-改进测试过程 v1.0.2（中文版），2011 年。

ISTQB-MAT，认证测试工程师-移动应用测试大纲v2019（中文版），2019 年。

ISTQB-MBT，认证测试工程师-基于模型的测试大纲 v2015（中文版），2017年。

ISTQB-PT，认证测试工程师-性能测试大纲v2018_CN1.0（中文版），2020年。

ISTQB-TAE，认证测试工程师-测试自动化工程师大纲v2.0_CN1.0（中文版），2025年。

ISTQB-TAE，认证测试工程师-高级测试管理大纲v3.0_CN1.1（中文版），2025年。

ISTQB-TTA，认证测试工程-高级技术测试分析师大纲v4.0_CN2.1（中文版），2025年。

ISTQB-UT，认证测试工程师-易用性测试大纲v2018_CN3.1（中文版），2023年。

文献

AMMANN, Paul; OFFUTT, Jeff, 2008. Introduction to Software Testing. Cambridge University Press. BEIZER, Boris, 1990. Software Testing Techniques (2nd Ed.) International Thomson Computer Press. BINDER, Robert V., 2000. Testing Object-Oriented Systems. Addison-Wesley.

ELSAYED, Elsayed A., 2021. Reliability Engineering. Wiley.

FORGÁCS, Istvan; KOVÁCS, Attila, 2019. Practical Test Design: Selection of traditional and automated test design techniques. BCS, The Chartered Institute for IT.

FORGÁCS, Istvan; KOVÁCS, Attila, 2024. Modern Software Testing Techniques. Apress.

GRADY, Robert; CASWELL, Deborah, 1987. Software Metrics: Establishing a Company-wide Program. Prentice Hall.

HENDRICKSON, Elisabeth, 2013. Explore It! Pragmatic Bookshelf.

JORGENSEN, Paul, 2014. Software Testing. A Craftsman's Approach. CRC Press.

KANER, Cem; FALK, Jack; NGUYEN, Hung Q., 1999. Testing Computer Software. Wiley.

KANER, Cem; PADMANABHAN, Sowmya; HOFFMAN, Douglas, 2013. The Domain Testing Workbook. Context Driven Press.

KOOMEN, Tim; AALST, Leo van der; BROEKMAN, Bart; VROON, Michiel, 2006. TMap Next for result- driven testing. UTN Publishers.

KUHN, D. Richard; YU, Lei; KACKER, Raghu N., 2013. Introduction to Combinatorial Testing. CRC Press.

文章

ALYAHYA, Sultan, 2020. Crowdsourced Software Testing: A Systematic Literature Review. Information and Software Technology. Vol. 127, p. 106363. Available from doi: 10.1016/j.infsof.2020.106363.

ANTONIOL, Giuliano; BRIAND, Lionel; DI PENTA, Massimiliano; LABICHE, Yvan, 2002. A case study using the round-trip strategy for state-based class testing. Proceedings 13th International Symposium on Software Reliability Engineering, pp. 269 - 279. isbn 0-7695-1763-

3. Available from doi: 10.1109/ ISSRE.2002.1173268.

ARCURI, Andrea; IQBAL, Muhammad Zohaib; BRIAND, Lionel, 2012. Random Testing: Theoretical Results and Practical Implications. IEEE Transactions on Software Engineering. Vol. 38, pp. 258－277. Available from doi: 10.1109/TSE.2011.121.

BARR, Earl; HARMAN, Mark; MCMINN, Phil; SHAHBAZ, Muzammil; YOO, Shin, 2014. The Oracle Problem in Software Testing: A Survey. IEEE Transactions on Software Engineering. Vol. 41, no. 5,pp. 507－525. Available from doi: 10.1109/TSE.2014.2372785.

BOCHMANN, Gregor; PETRENKO, Alexandre, 1994. Protocol Testing: Review of Methods and Relevance for Software Testing. ISSTA '94: Proceedings of the 1994 ACM SIGSOFT international symposium on Software testing and analysis, pp. 109－124. Available from doi: 10.1145/186258.187153.

CATOLINO, Gemma; PALOMBA, Fabio; ZAIDMAN, Andy; FERRUCCI, Filomena, 2019. Not All Bugs Are the Same: Understanding, Characterizing, and Classifying Bug Types. Journal of Systems and Software. Vol. 152, pp. 165－181. Available from doi: 10.1016/j.jss.2019.03.002.

CHILLAREGE, R. et al., 1992. Orthogonal Defect Classification - A Concept for In-Process Measurements.IEEE Transactions on Software Engineering. Vol. 18, pp. 943－956. Available from doi: 10.1109/ 32. 177364.

CHOW, Tsun, 1978. Testing Software Design Modeled by Finite-State Machines. IEEE Transactions on Software Engineering. Vol. 4, pp. 178－187. Available from doi: 10.1109/TSE.1978.231496.

COHEN, D.M.; DALAL, Siddhartha; KAJLA, A.; PATTON, G.C., 1994. The Automatic Efficient Test Generator (AETG) system. Proceedings of 1994 IEEE International Symposium on Software Reliability Engineering, pp. 303－309.isbn 0-8186-6665-X.Available from doi: 10.1109/ISSRE.1994.341392.

ENGSTRÖM, Emelie; RUNESON, Per; SKOGLUND, Mats, 2010. A systematic review on regression test selection techniques. Information and Software Technology. Vol. 52, pp. 14－30. Available from doi: 10.1016/j.infsof.2009.07.001.

GHAZI, Ahmad Nauman; GARIGAPATI, Ratna; PETERSEN, Kai, 2017. Checklists to Support Test

Charter Design in Exploratory Testing. Agile Processes in Software Engineering and Extreme Programming. XP 2017. isbn 978-3-319-57632-9. Available from doi: 10.1007/978-3-319-57633-6_17.HAREL, David, 1987. Statecharts: A Visual Formalism For Complex Systems. Science of Computer Programming. Vol. 8, pp. 231－274. Available from doi: 10.1016/0167-6423(87)90035-9.

HUANG, Rubing; SUN, Weifeng; XU, Yinyin; CHEN, Haibo; TOWEY, Dave; XIA, Xin, 2019. A Survey on Adaptive Random Testing. IEEE Transactions on Software Engineering. Vol. 47, no. 10, pp. 2052－2083. Available from doi: 10.1109/TSE.2019.2942921.

JENG, Bingchiang; WEYUKER, Elaine, 1994. A Simplified Domain-Testing Strategy. ACM Transactions on Software Engineering and Methodology. Vol. 3, pp. 254－270. Available from doi: 10.1145/196092. 193171.

JUERGENS, Elmar; PAGANO, Dennis; GOEB, Andreas, 2018. Test Impact Analysis: Detecting Errors Early Despite Large, Long-Running Test Suites. Whitepaper, CQSE GmbH.

KUHN, D.; WALLACE, Dolores; JR, A.M., 2004. Software Fault Interactions and Implications for Software Testing. IEEE Transactions on Software Engineering. Vol. 30, pp. 418－421. Available from doi: 10.1109/ TSE.2004.24.

LEICHT, Niklas; BLOHM, Ivo; LEIMEISTER, Jan Marco, 2017. Leveraging the Power of the Crowd for Software Testing. IEEE Software. Vol. 34, pp. 62－69. Available from doi: 10.1109/MS.2017.37.

OFFUTT, A. Jefferson, 1992. Investigations of the software testing coupling effect. ACM Transactions on Software Engineering and Methodology. Vol. 1, pp. 5－20.

RECHTBERGER, Vaclav; BURES, Miroslav; AHMED, Bestoun, 2022. Overview of Test Coverage Criteria for Test Case Generation from Finite State Machines Modelled as Directed Graphs. IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), Valencia, Spain,pp. 207－214.

RWEMALIKA, Renaud; KINTIS, Marinos; PAPADAKIS, Mike; LE TRAON, Yves; LORRACH, Pierre, 2019.On the Evolution of Keyword-Driven Test Suites. 12th IEEE Conference on Software Testing, Validation and Verification (ICST), pp. 335－345.

SEGURA, Sergio; FRASER, Gordon; SANCHEZ, Ana; RUIZ-CORTÉS, Antonio, 2016. A Survey on

Metamorphic Testing. IEEE Transactions on Software Engineering, pp. 46 - 53.

SEGURA, Sergio; TOWEY, Dave; ZHOU, Zhi Quan; CHEN, Tsong Yueh, 2020. Metamorphic Testing: Testing the Untestable. IEEE Software. Vol. 42, no. 9, pp. 805 - 824.

WU, Huayao; NIE, Changhai; PETKE, Justyna; JIA, Yue; HARMAN, Mark, 2020. An Empirical Comparison of Combinatorial Testing, Random Testing and Adaptive Random Testing. IEEE Transactions on Software Engineering. Vol. 46, no. 3, pp. 302 - 320.

网页

COMMISSION, European, 2016. General Data Protection Regulation: Regulation (EU) 2016/679 [online]. [visited on 2024-09-15]. Available from: https://commission.europa.eu/law/law-topic/data-protection/ legal-framework-eu-data-protection_en?.

COMPUTER SOCIETY, IEEE; JTC 1/SC7, ISO/IEC, 2024. SE VOCAB: Software and Systems

Engineering Vocabulary [online]. [visited on 2024-09-15]. Available from: https://pascal.computer.org/.

CZERWONKA, Jacek, 2004. Pairwise Testing: Combinatorial Test Case Generation [online]. [visited on 2024-09-15]. Available from: www.pairwise.org.

HEALTH, U.S. Department of; SERVICES, Human, 2024. Health Insurance Portability and Accountability Act: Standards for Privacy of Individually Identifiable Health Information (Privacy Rules) [online]. [visited on 2024-09-15]. Available from: https://www.hhs.gov/hipaa/for-professionals/privacy/laws-regulations/ index.html.

IREB-GLOSSARY, IREB®, 2024. The CPRE Glossary: Core terms of Requirements Engineering [online]. [visited on 2024-09-15]. Available from: https://glossary.istqb.org.

ISTQB-GLOSSARY, ISTQB®, 2024. Standard Glossary of Terms Used in Software Testing [online]. [visited on 2024-09-15]. Available from: https://glossary.istqb.org.

Site of Software Test Design, 2020 [online]. [visited on 2024-09-15]. Available from: https://test-design. org/.

SUMI, 1991. Software Usability Measurement Inventory: Standard evaluation questionnaire

for assessing quality of use [online]. [visited on 2024-09-15]. Available from: sumi.uxp.ie.

U.S. DEPARTMENT OF JUSTICE, Civil Rights Division, 2010. Americans with Disabilities Act: Guidance & Resource Materials [online]. [visited on 2024-09-15]. Available from: www.ada.gov/resources/.

UK GOVERNMENT, Equalities Office, 2010. Equality Act 2010: guidance: Information and guidance on the Equality Act 2010, including age discrimination and public sector Equality Duty. [online]. [visited on 2024-09-15]. Available from: www.gov.uk/guidance/equality-act-2010-guidance.

WAMMI, 1999. Website Analysis and Measurement Inventory: Professional service for analyzing user experience [online]. [visited on 2024-09-15]. Available from: www.wammi.com.

WCAG, 2023. Web Content Accessibility Guidelines 2.2: W3C Recommendation [online]. [visited on 2024- 09-15]. Available from: www.w3.org/TR/WCAG22/.

The previous references point to information available on the Internet and elsewhere. Even though those references were checked at the time of publication of this syllabus, the ISTQB® cannot be held responsible if the references are unavailable anymore.

7. 附录 A-学习目标/知识认知级别

本教学大纲中的具体学习目标在每章开头均有列出。教学大纲中的每个主题都将根据其学习目标进行考试。

学习目标均以与知识认知级别相对应的行为动词开头，具体如下所示。

级别 1：牢记（K1）

考生应牢记、认识和回顾术语或者概念的内容。

行为动词：回顾、认识

注意：高级大纲中没有针对 K1 级别的具体学习目标。然而，第 1 至 5 章的大纲内容以及各章标题下方列出的所有作为关键词的术语仍需牢记（适用于 K1 级别），即便学习目标中未明确提及这些内容。

级别 2：理解（K2）

考生应能选择与大纲主题相关的陈述理由或解释，同时对测试概念能够进行总结、比较、分类和举例说明。

行为动词：分类、比较、区分、区别、解释、举例说明、解读、总结

例子	说明
区分功能正确性、功能适合性以及功能完备性测试。	寻找概念之间的差异。
解释CRUD测试。	
举例说明测试环境需求。	
总结测试分析师在各软件开发生存周期中的参与情况。	

级别 3：应用（K3）

考生能在遇到熟悉的任务时执行规程，或者选择正确的规程并应用于给定的周境中。

行为动词：应用、实施、准备、使用

例子	说明
应用域测试。	应参照某一规程/技术/过程等。
为基于会话的测试准备测试章程。	

使用关键词驱动测试来开发测试脚本。	可用于一个学习目标中，该目标要求考生能够使用一种技术或规程，与“应用”相似。

级别 4：分析(K4)

考生能够将与某个规程或技术相关的信息分解为各个组成部分，以便于更好地理解这些内容，并能区别事实与推论。一个典型的应用是分析文档、软件或项目情况，并提出适当的行动方案来解决某个问题或完成一项任务。

行为动词：分析、解构、概述、优先级排序、选择

例子	说明
分析变更的影响，以确定回归测试的范围。	只有与分析的可衡量目标相结合时才能进行检验。其表述形式应为“对 X 进行分析以得出 Y 结果”（或类似表述）。
选择合适的测试技术来缓解给定场景的产品风险。	在需要进行分析的场景中适用。

参考文献

（关于学习目标的认知级别）

Anderson, L. W. and Krathwohl, D. R. (eds) (2001) A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom’s Taxonomy of Educational Objectives, Allyn & Bacon

8. 附录 B- 商业价值与学习目标的可追溯性矩阵

本节列出了高级测试分析师大纲中商业价值与学习目标之间的可追溯性。

商业价值：高级测试分析师		TA-B01	TA-B02	TA-B03	TA-B04	TA-B05	TA-B06	TA-B07	TA-B08	TA-B09
TA-B01	按照所遵循的软件生存周期模型，支持并执行适当的测试活动。	6								
TA-B02	应用基于风险的测试原则。		3							
TA-B03	选择并应用适当的测试技术，以支持测试目标的实现。			13						
TA-B04	提供详细程度适当且质量达标的文档。				5					
TA-B05	确定需要执行的功能测试类型。					1				
TA-B06	参与非功能测试。						3			
TA-B07	帮助预防缺陷。							5		
TA-B08	利用工具提高测试过程的效率。								4	
TA-B09	明确测试环境和测试数据的要求。									2

商业价值：高级测试分析师		TA-B01	TA-B02	TA-B03	TA-B04	TA-B05	TA-B06	TA-B07	TA-B08	TA-B09
学习目标编号	学习目标(K-级别)									
1	测试分析师在测试过程中的任务 - 225 分钟									
1.1	软件开发生存周期中的测试									
TA-1.1.1	总结测试分析师在不同软件开发生存周期中的参与（K2）	×								
1.2	参与测试活动									
TA-1.2.1	总结测试分析师在测试分析中的任务（K2）	×								
TA-1.2.2	总结测试分析师在测试设计中的任务（K2）	×								
TA-1.2.3	总结测试分析师在测试实施中的任务（K2）	×								
TA-1.2.4	总结测试分析师在测试执行中的任务（K2）	×								
1.3	与工作产品相关的任务									
TA-1.3.1	区分概要测试用例与详细测试用例				×					
TA-1.3.2	解释测试用例的质量准则（K2）				×					
TA-1.3.3	举例说明测试环境需求（K2）				×					×
TA-1.3.4	解释测试结果参照物问题及其潜在解决方案（K2）				×					
TA-1.3.5	举例说明测试数据需求（K2）				×					×
TA-1.3.6	使用关键词驱动测试开发测试脚本（K2）	×							×	

商业价值：高级测试分析师		TA-B01	TA-B02	TA-B03	TA-B04	TA-B05	TA-B06	TA-B07	TA-B08	TA-B09
TA-1.3.7	总结用于管理测试件的工具类型（K2）								X	
2	基于风险的测试中测试分析师的任务 - 90 分钟									
2.1	风险分析									
TA-2.1.1	总结测试分析师对产品风险分析的贡献（K2）		X							
2.2	风险控制									
TA-2.2.1	分析变更的影响，以确定回归测试的范围（K4）		X						X	
3	测试分析与测试设计 - 615 分钟									
3.1	基于数据的测试技术									
TA-3.1.1	应用域测试（K3）			X						
TA-3.1.2	应用组合测试（K3）			X						
TA-3.1.3	总结随机测试的收益和局限性（K2）			X						
3.2	基于行为的测试技术									
TA-3.2.1	解释CRUD测试（K2）			X						
TA-3.2.2	应用状态转移测试（K3）			X						
TA-3.2.3	应用基于场景的测试（K3）			X						
3.3	基于规则的测试技术									

商业价值：高级测试分析师		TA-B01	TA-B02	TA-B03	TA-B04	TA-B05	TA-B06	TA-B07	TA-B08	TA-B09
TA-3.3.1	应用判定表测试（K3）			X						
TA-3.3.2	应用蜕变测试（K3）			X						
3.4	基于经验的测试技术									
TA-3.4.1	为基于会话的测试准备测试章程（K3）			X						
TA-3.4.2	准备支持基于经验的测试的检查表（K3）			X						
TA-3.4.3	举例说明众测的收益和局限性（K2）			X						
3.5	应用最合适的测试技术									
TA-3.5.1	选择适当的测试技术来缓解给定场景的产品风险（K4）		X	X						
TA-3.5.2	解释自动化测试设计的收益和风险（K2）								X	
4	测试质量特性 - 60分钟									
4.1	功能测试									
TA-4.1.1	区分功能正确性、功能适合性和功能完备性测试（K2）					X				
4.2	易用性测试									
TA-4.2.1	解释测试分析师如何为易用性测试做出贡献（K2）						X			
4.3	灵活性测试									
TA-4.3.1	解释测试分析师如何为适应性和易安装性测试做出贡献（K2）						X			

商业价值：高级测试分析师		TA-B01	TA-B02	TA-B03	TA-B04	TA-B05	TA-B06	TA-B07	TA-B08	TA-B09
4.4	兼容性测试									
TA-4.4.1	解释测试分析师如何为互操作性测试做出贡献（K2）						×			
5	软件缺陷预防 - 225 分钟									
5.1	缺陷预防实践									
TA-5.1.1	解释测试分析师如何为缺陷预防做出贡献(K2)							×		
5.2	支持阶段遏制									
TA-5.2.1	使用测试对象的模型来检测规格说明中的缺陷（K3）							×		
TA-5.2.2	应用评审技术从测试依据中发现缺陷（K3）							×		
5.3	减少缺陷再次发生									
TA-5.3.1	分析测试结果以识别缺陷检测方面的潜在改进措施（K4）							×		
TA-5.3.2	解释缺陷分类如何支持根本原因分析（K2）							×		

9. 附录 C - 发布说明

ISTQB® 高级测试分析师大纲（4.0 版）进行了重大更新。因此，各章节均未提供详细的发布说明。不过，以下提供了主要变化的概要。此外，在单独的发布说明文档中，ISTQB® 提供了高级测试分析师大纲（3.1.2 版和 4.0 版）中学习目标之间的详细可追溯性。

此次重大更新带来了以下变化：

教学大纲结构

所有学习目标均已进行修改，使其具备原子性，并实现从学习目标到内容的一对一可追溯性，以避免出现内容没有对应学习目标的情况。每个学习目标均在具有相同编号的章节中进行描述（例如，TA-1.1.1 在第 1.1.1 节中进行讨论）。此次修订的目的是使该版本更易于阅读、理解、学习和翻译，重点在于提高实用价值并兼顾知识与技能之间的平衡。

在 v4.0 版本中，有 36 个学习目标，而在 v3.1.2 版本中则有 31 个。有几个 K4 级别的学习目标被调整为 K2 或 K3 级别。对于与测试技术相关的学习目标，其动机在于应将重点放在应用测试技术上，而非分析文档以进行进一步的测试设计。有些学习目标被合并为一个单一的学习目标。下面的表格展示了关于学习目标数量和培训时间的详细统计数据。

大纲版本	# K2学习目标（时间）	# K3学习目标（时间）	# K4学习目标（时间）	# 合计学习目标（时间）
现行版本（4.0）	22 (330 分钟)	11 (660分钟)	3 (225分钟)	36 (1215 分钟)
先前版（3.1.2）	16 (240 分钟)	5 (300 分钟)	10 (750 分钟)	31 (1290 分钟)

测试技术分类

第 3 章的结构与 v3.1.2 版相比，对测试技术的分类更加详细。现在，黑盒测试技术根据其底层测试对象模型的类型被分为基于数据的、基于行为的和基于规则的类型。

扩大第 5 章的范围

旧的第五章（仅专注于评审）已被扩展，以讨论测试分析师采用其他重要的缺陷预防实践的形式，例如使用模型来检测规格说明中的缺陷、分析测试结果以提高缺陷检测能力以及使用缺陷分类来支持根本原因分析。

学习目标的变化

- 第1章重新划分为一个关于测试活动的板块和一个关于测试件的板块。
- 关于概要测试用例和详细测试用例的主题已从（旧版 TA-1.4.2）K4 级别降级至（新版 TA-1.3.1）K2级别。
- 将基于风险的测试（旧版 TA-2.1.1）K3 LO分成了两个部分，分别是风险分析相关的 K2 TA-2.1.1以及与风险控制相关的 K4 TA-2.2.1。
- 等价类划分（旧版 K4 TA-3.2.1）和边界值分析（旧版 K4 TA-3.2.2）已被更通用的、用于域测试的K3 TA-3.1.1 取代。
- 成对测试（旧版 K4 TA-3.2.6）和分类树图（旧版 K2 TA-3.2.5）被合并为一个更通用的、用于组合测试的 K3 TA-3.1.2。
- 状态转移测试（旧K4 TA-3.2.4）已被K3 TA-3.2.2取代，后者专注于N-切换覆盖率和往返覆盖率。与CTFL的冗余部分已被移除。
- 用例测试（旧版 K4 TA-3.2.7）已被更通用的、基于场景的测试的 K3 TA-3.2.3 取代。
- 判定表测试（旧版 K4 TA-3.2.3 版本）已被降级至K3（TA-3.3.1 ）。
- 探索性测试（旧版 K3 TA-3.3.2）已被两个独立的学习目标取代：准备测试章程（K3 TA-3.4.1），以及准备支持基于经验的测试的检查表（K3 TA-3.4.2）。与当前CTFL v4.0版本中的冗余内容已被移除。
- 与比较测试技术及应用最适当技术有关的四个学习目标（旧版 K4 TA-3.2.8、K2 TA-3.3.3、K2 TA-3.4.1、K2 TA-3.3 .1）已合并为一个K4 TA-3.5.1。
- 关于功能测试的四个学习目标（旧版：K2 TA-4.2.1、K2 TA-4.2.2、K2 TA-4.2.3 和 K4 TA-4.2.7）被合并为一个K2 TA-4.1.1。由于功能测试在第 3 章“测试技术”这一部分中已有大量描述，因此该主题被简化处理。
- 第6章（测试工具）中的主题已移至第 1.3 节，重点在于更实际的问题探讨。旧版 K3 TA-6.2.1 中“对于给定场景，确定测试分析师在关键词驱动的测试项目中的适当活动”已稍作修改，变为 K3 TA-1.3.6 “使用关键词驱动测试来开发测试脚本”。旧版 K2 TA-6.3.1“解释测试设计、测试数据准备及测试执行期间所使用测试工具的方法和类型”这一内容稍作修改，变为 K2 TA-1.3.7“总结用于管理测试件的工具类型”。

- 在评审的部分存在两个类似的学习目标（旧版 K3 TA-5.2.1 和 K3 TA-5.2.2）被合并为一个 K3 TA-5.2.2。

新的议题

- 测试用例的质量准则(K2 TA-1.3.2)。
- 测试环境需求(K2 TA-1.3.3)。
- 确定测试结果参照物(K2 TA-1.3.4)。
- 测试数据需求(K2 TA-1.3.5)。
- 随机测试(K2 TA-3.1.3)。
- CRUD测试(K2 TA-3.2.1)。
- 蜕变测试(K3 TA-3.3.2)。
- 众测(K2 TA-3.4.3)。
- 自动化测试设计的收益与风险(K2 TA-3.5.2)。
- 测试分析师对缺陷预防的贡献(K2 TA-5.1.1)。
- 使用模型来检测规格说明中的缺陷。(K3 TA-5.2.1)。
- 分析测试结果以提高缺陷检测能力(K4 TA-5.3.1)。
- 通过缺陷分类支持根本原因分析(K2 TA-5.3.2)。

标准更新

新版本的软件质量国际标准（*ISO/IEC 25010*，2023）以及测试技术标准（*ISO/IEC/IEEE 29119-4*，2021）中的变化促成了课程大纲相应内容的调整。

10. 附录 D - 缩略语列表

缩略语	含义
AI	人工智能（artificial intelligence）
BPMN	业务流程模型和标注（Business Process Model and Notation）
BVA	边界值分析（boundary value analysis）
CRUD	创建（create）、读取（read）、更新（update）和删除（delete）
CSV	逗号分隔值（comma-separated value）
DDP	缺陷检测率（defect detection percentage）
DRE	缺陷清除效率（defect removal efficiency）
EP	等价类划分（equivalence partitioning）
GDPR	通用数据保护条例（General Data Protection Regulation）
JSON	JavaScript对象表示法（JavaScript Object Notation）
MBT	基于模型的测试（model-based testing）
MR	蜕变关系（metamorphic relation）
MT	蜕变测试（metamorphic testing）
ODC	正交缺陷分类法（orthogonal defect classification）
PCE	阶段遏制有效性（phase containment effectiveness）
RCA	根本原因分析（root cause analysis）
SDLC	软件开发生存周期（software development lifecycle）
TA	测试分析师（test analyst）
TTA	技术测试分析师（technical test analyst）
UML	统一建模语言（Unified Modeling Language）
UX	用户体验（user experience）
XML	可扩展标记语言（Extensible Markup Language）

11. 附录 E - 领域特定术语

术语	解释
CRUD矩阵	一种用于表明系统内各功能对实体所执行操作类型的矩阵。
数据语义	数据的含义与解释。
五问法	一种迭代式追问技术，通过连续五次追问“为什么”，每次都将当前的“为什么”指向上一次“为什么”的答案，以此来确定缺陷或问题的根本原因。
帕累托分析	一种用于识别能带来最大收益的问题领域或任务的方法。
用户旅程图	用户体验可视化文档，展示用户在实现目标的过程中所采取的步骤。
用户研究	一门通过研究用户执行任务的方式、观察他们与组件或系统的交互情况，或通过数据分析和解读，来了解用户需求和思维过程的学科。

12. 附录 F - 软件质量模型

下表展示了 ISO/IEC 25010 产品质量模型（ISO/IEC 25010，2023 年）中的质量特性。表中注明了本教学大纲（TA）中涉及到的特性/子特性，以及 ISTQB 的其他教学大纲中（包括技术测试分析师（TTA）、性能测试（PT）、易用性测试（UT）、汽车软件测试工程师（AuT）和信息安全性测试员（SEC））涵盖的特性/子特性。若某一质量特性被多个教学大纲覆盖，则按覆盖详细程度从高到低排列。此外，表中还将现行的 ISO/IEC 25010 模型与本教学大纲上一版所采用的 2011 年版进行了对比。

ISO/IEC 25010:2023 （现行标准）	ISO/IEC 25010:2011 （先前的标准）	注释	ISTQB®大纲
功能性	**功能性**		**TA**
功能完备性	功能完备性		
功能正确性	功能正确性		
功能适合性	功能适合性		
性能效率	**性能效率**		PT，TTA
时间特性	时间特性		
资源利用性	资源利用性		
容量	容量		
兼容性	**兼容性**		**TA**，TTA
共存性	共存性		TTA
互操作性	互操作性		**TA**
交互能力	**易用性**	**重命名**	UT，**TA**
可辨识性	可辨识性		
易学性	易学性		
易操作性	易操作性		
用户差错防御性	用户差错防御性		
用户粘性	用户界面舒适性	重命名	
包容性	易访问性	拆分并重命名	
用户协助			
自描述性		新增	
可靠性	**可靠性**		TTA
无故障性	成熟性	重命名	
可用性	可用性		
容错性	容错性		

ISO/IEC 25010:2023 （现行标准）	ISO/IEC 25010:2011 （先前的标准）	注释	ISTQB®大纲
易恢复性	易恢复性		
信息安全性	**信息安全性**		SEC，TTA
保密性	保密性		
完整性	完整性		
抗抵赖性	抗抵赖性		
可核查性	可核查性		
真实性	真实性		
耐受性		新增	
维护性	**维护性**		TTA
模块化	模块化		
可重用性	可重用性		
易分析性	易分析性		
易修改性	易修改性		
易测试性	易测试性		
灵活性	**可移植性**	**重命名**	TTA，TA，PT
适应性	适应性		TA，TTA
可扩展性		新增	PT
易安装性	易安装性		TA，TTA
易替换性	易替换性		TTA
功能安全性/无危害性		**新增**	AuT
运行限制性		新增	
风险识别		**新增**	
故障安全性		**新增**	
危险警告性		**新增**	
安全集成性		**新增**	

13. 附录 G - 商标

ISTQB® 是国际软件测试认证委员会的注册商标。

UML® 是对象管理组织（OMG）的注册商标。

BPMN™ 是对象管理组织（OMG）的商标。

14. 索引