Certified Tester

Advanced Level Syllabus

Test Analyst Syllabus

versão 4.0

International Software Testing Qualifications Board

BSTQB®
CTAL-TA 4.0

Direitos autorais

Aviso de direitos autorais© International Software Testing Qualifications Board (doravante denominado ISTQB®)
ISTQB® é uma marca registrada do International Software Testing Qualifications Board.

Copyright © 2025 Os autores da v4.0: Armin Born, Filipe Carlos, Wim Decoutere, István Forgács, Matthias Hamburg (proprietário do produto), Attila Kovács, Sandy Liu, François Martin, Stuart Reid, Adam Roman, Jan Sabak, Murian Song, Tanja Tremmel, Marc-Florian Wendland, Tao Xian Feng.

Direitos autorais © 2021-2022. Os autores da atualização v3.1.0, errata 3.1.1 e errata 3.1.2: Wim Decoutere, István Forgács, Matthias Hamburg, Adam Roman, Jan Sabak, Marc-Florian Wendland.

Copyright © 2019 Os autores da atualização de 2019: Graham Bath, Judy McKay, Jan Sabak, Erik van Veenendaal.

Copyright© 2012. Os autores: Judy McKay, Mike Smith, Erik van Veenendaal.

Todos os direitos reservados. Os autores transferem os direitos autorais para o ISTQB®. Os autores (como atuais detentores dos direitos autorais) e o ISTQB® (como futuro detentor dos direitos autorais) concordaram com as seguintes condições de uso:

- Extratos deste documento para uso não comercial podem ser copiados se a fonte for mencionada. Qualquer Provedor de Treinamento Credenciado pode usar este syllabus como base para um curso de treinamento se os autores e o ISTQB® forem reconhecidos como a fonte e os proprietários dos direitos autorais do syllabus e desde que qualquer anúncio de tal curso de treinamento possa mencionar o syllabus somente após o credenciamento oficial dos materiais de treinamento ter sido recebido de um Conselho de Membros reconhecido pelo ISTQB®.
- Qualquer indivíduo ou grupo de indivíduos pode usar este syllabus como base para artigos e livros, desde que os autores e o ISTQB® sejam reconhecidos como a fonte e os proprietários dos direitos autorais do syllabus.
- É proibido qualquer outro uso deste syllabus sem antes obter a aprovação por escrito do ISTQB®
- Qualquer Conselho Membro reconhecido pelo ISTQB® pode traduzir este syllabus desde que reproduza o Aviso de Direitos Autorais acima mencionado na versão traduzida do syllabus

Histórico da Revisão

Versão	Data	Observações
v4.0	2025/05/02	Grande atualização com revisão geral e atualização do escopo
v3.1.2	2022/01/31	Errata: Pequenos defeitos de formatação, gramática e redação corrigidos
v3.1.1	2021/05/15	Errata: O aviso de direitos autorais foi adaptado para os padrões atuais do ISTQB®
v3.1	2021/03/03	Pequena atualização com a seção 3.2.3 reescrita e vários aprimoramentos de redação
v3.0 (2019)	2019/10/19	Grande atualização com revisão geral e redução do escopo
v2.0 (2012)	2012/10/19	Primeira versão como um syllabus CTAL-TA separado

Histórico da versão de tradução do BSTQB

Data	Observações
04/06/2026	Lançamento da versão na Língua Portuguesa

©International Software Testing Qualifications Board

Índice

©International Software Testing Qualifications Board

©International Software Testing Qualifications Board

©International Software Testing Qualifications Board

Agradecimentos

A Assembleia Geral do ISTQB® divulgou formalmente este documento em 2 de maio de 2025.

Ele foi produzido por uma equipe do International Software Testing Qualifications Board: Armin Born, Filipe Carlos, Wim Decoutere, István Forgács, Matthias Hamburg (proprietário do produto), Attila Kovács, Sandy Liu, François Martin, Stuart Reid, Adam Roman, Jan Sabak, Murian Song, Tanja Tremmel, Marc-Florian Wendland, Tao Xian Feng.

A equipe agradece a Julia Sabatine por sua revisão, a Gary Mogyorodi por sua revisão técnica, a Daniel Pol'an por seu constante suporte técnico, à equipe de revisão e aos Conselhos de Administração por suas sugestões e contribuições.

As seguintes pessoas participaram da revisão, dos comentários e da votação deste syllabus:

Edição atual (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.

Edição 2021-2022 (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.

Posteriormente, Tal Pe'er, Stuart Reid, Marc-Florian Wendland e Matthias Hamburg sugeriram melhorias formais, gramaticais e de redação que foram implementadas e publicadas nas Erratas 3.1.1 e 3.1.2.

Edição 2019 (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.

Edição 2012 (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

O BSTQB® agradece aos voluntários do WGT BSTQB® pelo empenho e esforço na tradução e revisão deste material: Eduardo Rodrigues Medeiros, George Fialkovitz, Irene Nagase, Osmar Higashi, Paula Oliveira, Rogério Athaide de Almeida, Stênio Viveiros.

O BSTQB® também agradece aos ISTQB® Accredited Training Providers no Brasil que contribuíram na revisão da tradução com sugestões e questionamentos. No momento deste trabalho os seguintes provedores estavam credenciados: Conecteseaqui, Exseed, Iterasys.

O BSTQB® também agradece às empresas brasileiras credenciadas no ISTQB® Partner Program que contribuíram na revisão da tradução com sugestões e questionamentos. No momento deste trabalho as seguintes empresas estavam credenciadas: Deltapoint, Keeggo, Venturus.

©International Software Testing Qualifications Board

0 Introdução

0.1 Objetivo deste syllabus

Este syllabus forma a base para a International Software Testing Qualification para o Advanced Level Test Analyst. O ISTQB® fornece esse syllabus da seguinte forma:

1. Aos Conselhos Membros, para traduzir para seu idioma local e credenciar os provedores de treinamento. Os Conselhos Membros podem adaptar o syllabus às suas necessidades específicas de idioma e modificar as referências para adaptá-las às suas publicações locais.
2. Aos Órgãos de Certificação, para que elaborem questões de exame em seu idioma local, adaptadas aos objetivos de aprendizagem deste syllabus.
3. Aos Provedores de Treinamento, para produzir material de treinamento e determinar métodos de ensino adequados.
4. Para os candidatos à certificação, para se preparar para o exame de certificação (como parte de um curso de treinamento ou de forma independente).
5. À Comunidade Internacional de Engenharia de Software e sistemas para promover a profissão de teste de software e sistemas e como base para livros e artigos.

0.2 O Advanced Level Test Analyst

A certificação ISTQB® Advanced Level Test Analyst (CTAL-TA) fornece as habilidades necessárias para realizar testes de software estruturados e completos em todo o ciclo de vida de desenvolvimento de software. Ela detalha a função e as responsabilidades do Analista de Teste em cada etapa de um processo de teste padrão e amplia as técnicas de teste importantes. A certificação ISTQB® Advanced Level Test Analyst destina-se a pessoas que possuem um certificado de Nível Fundamental e que desejam desenvolver ainda mais seus conhecimentos em análise e técnicas de teste.

Neste syllabus, o Analista de Teste é entendido como uma função que:

- concentra-se mais nas necessidades de negócio do cliente do que nos aspectos técnicos dos testes.
- realiza principalmente testes funcionais, mas também contribui para testes não funcionais com foco no usuário, como testes de usabilidade, adaptabilidade, instalabilidade ou interoperabilidade.
- usa técnicas de teste caixa-preta e testes baseados na experiência em vez de técnicas de teste caixa-branca.
- melhora a eficácia dos testes usando técnicas de prevenção de defeitos.

0.3 Carreira para testadores

O esquema do ISTQB® apoia profissionais de teste em todos os estágios de suas carreiras. Os indivíduos que obtiverem a certificação ISTQB® Advanced Level Test Analyst também podem se interessar pelos outros Core Advanced Levels (Technical Test Analyst e Test Management) e, posteriormente, pelo Expert Level (Test Management ou Improving the Test Process). Qualquer pessoa que queira desenvolver habilidades em práticas de teste em uma área de ambiente ágil pode considerar as certificações Agile Technical Tester ou Agile Test Leadership at Scale. Além disso, os fluxos especializados oferecem produtos de certificação com foco em tecnologias e abordagens de teste específicas, características de qualidade e níveis de teste específicos ou testes em domínios específicos do setor. Acesse www.istqb.org para obter as informações mais recentes sobre o Esquema de Certificação de Testes do ISTQB®, ou acesse https://bstqb.qa para conhecer as certificações disponíveis na língua portuguesa.

©International Software Testing Qualifications Board

0.4 Resultados de Negócio

Esta seção lista os Resultados de Negócios esperados de um candidato que tenha obtido a certificação Advanced Level Test Analyst.

Um ISTQB® Advanced Level Test pode...

Código	Descrição
TA-BO1	Apoiar e realizar testes adequados com base no ciclo de vida de desenvolvimento de software seguido
TA-BO2	Aplicar os princípios de testes baseados em riscos
TA-BO3	Selecionar e aplicar técnicas de teste adequadas para apoiar a realização dos objetivos do teste
TA-BO4	Fornecer documentação com níveis adequados de detalhes e qualidade
TA-BO5	Determinar os tipos apropriados de testes funcionais a serem realizados
TA-BO6	Contribuir para testes não funcionais
TA-BO7	Contribuir para a prevenção de defeitos
TA-BO8	Melhorar a eficiência do processo de teste com o uso de ferramentas
TA-BO9	Especificar os requisitos para ambientes de teste e dados de teste

0.5 Objetivos de aprendizagem (LO) e Nível Cognitivo de Conhecimento (K)

Os objetivos de aprendizagem apoiam os resultados de negócio e são usados para criar os exames Certified Tester Advanced Level Test Analyst.

Em geral, todo o conteúdo deste syllabus é passível de exame, exceto a Introdução e os Apêndices. As questões do exame confirmarão o conhecimento das palavras-chave no nível K1 (veja abaixo) ou os objetivos de aprendizagem no respectivo nível de conhecimento.

Os objetivos específicos de aprendizado e seus níveis de conhecimento são mostrados no início de cada capítulo e classificados da seguinte forma:

- K2: Compreender
- K3: Aplicar
- K4: Analisar

Para todos os termos listados como palavras-chave logo abaixo dos títulos dos capítulos, o nome e a definição corretos do glossário do ISTQB® devem ser lembrados (K1), mesmo que não sejam explicitamente mencionados em nenhum objetivo de aprendizado.

Mais detalhes e exemplos de objetivos de aprendizagem são fornecidos na Seção 7.

0.6 Exame do Certified Tester Advanced Level Test Analyst

O exame para obtenção do certificado de Certified Tester Advanced Level Test Analyst será baseado neste syllabus. As respostas às perguntas do exame podem exigir o uso de material baseado em mais de uma seção deste syllabus. Todas as seções do syllabus são passíveis de exame, exceto a Introdução e os Apêndices. Normas e livros são incluídos como referências, mas seu conteúdo não é passível de exame além do que está resumido no próprio syllabus a partir dessas normas e livros.

Consulte o documento Exam Structures and Rules compatível com o Advanced Level Test Analyst v4.0 para obter mais detalhes.

©International Software Testing Qualifications Board

O critério de entrada para fazer o exame ISTQB® Certified Tester Advanced Level Test Analyst é que os candidatos estejam interessados em testes de software. Entretanto, é altamente recomendável que os candidatos tenham:

- Ter pelo menos uma formação mínima em desenvolvimento ou teste de software, como seis meses de experiência como testador de sistemas ou de aceite de usuários ou como desenvolvedor de software;
- Fazer um curso que tenha sido credenciado de acordo com os padrões do ISTQB® (por um dos Conselhos Membros reconhecidos pelo ISTQB®).

Observação sobre os requisitos de entrada: o certificado ISTQB® Foundation Level deve ser obtido antes do exame de certificação Advanced Level Test Analyst.

0.7 Credenciamento

Um Conselho Membro ISTQB®, como o BSTQB®, pode credenciar provedores de treinamento cujo material do curso siga este syllabus. Os provedores de treinamento devem obter as diretrizes de credenciamento do Conselho Membro ou do órgão que realiza o credenciamento. Um curso credenciado é reconhecido como estando em conformidade com este syllabus e permite que um exame ISTQB® seja incluído no curso. As diretrizes de credenciamento para este syllabus seguem as diretrizes gerais de credenciamento publicadas pelo ISTQB® Processes Management and Compliance Working Group

0.8 Manuseio de padrões

Organizações internacionais de normas e padrões, como IEEE e ISO, emitiram padrões associados a características de qualidade e testes de software. Essas normas são referenciadas neste syllabus. O objetivo dessas referências é fornecer uma estrutura (como nas referências à ISO/IEC 25010 com relação às características de qualidade) ou fornecer uma fonte de informações adicionais, se o leitor desejar. Observe que os syllabi do ISTQB® usam normas como referência. As normas não se destinam a exame. Consulte a seção 6 - Referências para obter mais informações sobre normas.

0.9 Nível de detalhe

O nível de detalhamento desse syllabus permite cursos e exames consistentes em nível internacional. Para atingir esse objetivo, o syllabus consiste em:

- Objetivos gerais de instrução que descrevem a intenção do Advanced Level Test Analyst;
- Uma lista de termos que os alunos devem ser capazes de lembrar;
- Objetivos de aprendizagem para cada área de conhecimento, descrevendo os resultados cognitivos de aprendizagem a serem alcançados;
- Uma descrição dos principais conceitos, incluindo referências a fontes, como literatura ou normas aceitas.

O conteúdo do syllabus não descreve toda a área de conhecimento de testes de software; ele reflete o nível de detalhes a ser abordado nos cursos de treinamento de Advanced Level Test Analyst.

O syllabus usa a terminologia (ou seja, o nome e o significado) dos termos usados em testes de software e garantia de qualidade de acordo com o ISTQB® Glossary (ISTQB-Glossary, 2024).

Para conhecer a terminologia de disciplinas relacionadas, consulte os respectivos glossários: IREB-CPRE para Engenharia de Requisitos (IREB-Glossary, 2024) e IEEE-Pascal para Engenharia de Software (Computer Society et al., 2024).

0.10 Como syllabus está organizado

Há cinco capítulos com conteúdo passível de exame. O cabeçalho de nível superior de cada capítulo especifica o tempo alocado para o capítulo; o tempo não é fornecido abaixo do nível do capítulo. Para cursos de treinamento

©International Software Testing Qualifications Board

credenciados, o syllabus exige um mínimo de 20,25 horas de instrução (1215 minutos), distribuídas pelos cinco capítulos da seguinte forma:

- Capítulo 1 (225 minutos): As tarefas do Analista de Teste no Processo de Teste.
  - O aluno aprende como o Analista de Teste está envolvido em vários ciclos de vida de desenvolvimento de software.
  - O aluno aprende como o Analista de Teste está envolvido em várias atividades de teste.
  - O aluno aprende sobre as tarefas realizadas pelo Analista de Teste relacionadas aos produtos de trabalho.
- Capítulo 2 (90 minutos) As tarefas do Analista de Teste em Testes Baseados em Riscos.
  - O aluno aprende como o Analista de Teste contribui para a análise de risco do produto.
  - O aluno aprende a analisar o impacto das mudanças para determinar o escopo do teste de regressão.
- Capítulo 3 (615 minutos) Análise e Projeto de Testes
  - O aluno aprende sobre técnicas de teste baseadas em dados, como teste de domínio, teste combinatório e teste aleatório.
  - O aluno aprende sobre técnicas de teste baseado no comportamento, como teste CRUD, teste de transição de estado e teste baseado em cenário.
  - O aluno aprende sobre técnicas de teste baseadas em regras, como teste de tabela de decisão e teste metamórfico.
  - O aluno aprende sobre testes baseados na experiências, como testes baseados em sessões e testes de multidões.
  - O aluno aprende a selecionar técnicas de teste adequadas para reduzir os riscos do produto.
- Capítulo 4 (60 minutos) Teste de características de qualidade
  - O aluno aprende a realizar vários tipos de testes funcionais.
  - O aluno aprende a usar o conhecimento específico de funcionalidade para contribuir com tipos de teste não funcionais, como teste de usabilidade, teste de flexibilidade e teste de compatibilidade.
- Capítulo 5 (225 minutos) Prevenção de defeitos de software
  - O aluno aprende sobre várias práticas de prevenção de defeitos.
  - O aluno aprende sobre várias abordagens que apoiam a contenção de fases.
  - O aluno aprende a mitigar a recorrência de defeitos.

©International Software Testing Qualifications Board

1 As tarefas do Analista de Teste no Processo de Teste – 225 min

Palavras-chave

caso de teste de alto nível, palavra-chave, teste orientado por palavra-chave, caso de teste de baixo nível, ciclo de vida de desenvolvimento de software, análise de teste, analista de teste, caso de teste, condição de teste, dados de teste, projeto de teste, ambiente de teste, execução de teste, implementação de teste, oráculo de teste, script de teste, testware

Objetivos de aprendizagem para o Capítulo 1:

1.1 Testes no ciclo de vida do desenvolvimento de software

TA-1.1.1 (K2) Resumir o envolvimento do Analista de Teste em vários ciclos de vida de desenvolvimento de software.

1.2 Envolvimento em atividades de teste

TA-1.2.1 (K2) Resumir as tarefas executadas pelo Analista de Teste como parte da análise de testes.

TA-1.2.2 (K2) Resumir as tarefas realizadas pelo Analista de Teste como parte do projeto do teste.

TA-1.2.3 (K2) Resumir as tarefas realizadas pelo Analista de Teste como parte da implementação do teste.

TA-1.2.4 (K2) Resumir as tarefas realizadas pelo Analista de Teste como parte da execução do teste.

1.3 Tarefas relacionadas a produtos de trabalho

TA-1.3.1 (K2) Diferenciar entre casos de teste de alto nível e casos de teste de baixo nível.

TA-1.3.2 (K2) Explicar os critérios de qualidade para casos de teste.

TA-1.3.3 (K2) Dar exemplos de requisitos de ambiente de teste.

TA-1.3.4 (K2) Explicar o problema do oráculo de teste e as possíveis soluções.

TA-1.3.5 (K2) Dar exemplos de requisitos de dados de teste.

TA-1.3.6 (K3) Usar testes orientados por palavras-chave para desenvolver scripts de teste.

TA-1.3.7 (K2) Resumir os tipos de ferramentas para gerenciar o material de teste.

©International Software Testing Qualifications Board

Introdução

O Foundation Level Syllabus (ISTQB-CTFL, v4.0.1) descreve duas funções principais no teste: uma função de Gerenciamento de Teste e uma função de Testador. Nesse syllabus, a pessoa em uma função de testador responsável por testar os aspectos de negócio do software é chamada de Analista de Teste (TA). Embora essa responsabilidade raramente seja atribuída a um cargo ou função dedicada, o TA tem tarefas claramente definidas e competências necessárias. Em termos de níveis de teste, o TA se concentra em testes de sistema, testes de aceite e testes de integração de sistema. Em termos de características de qualidade, as competências do TA concentram-se na adequação funcional e abrangem determinadas características de qualidade não funcionais voltadas para o usuário, como usabilidade, adaptabilidade, instalabilidade e interoperabilidade.

1.1 Testes no ciclo de vida do desenvolvimento de software

1.1.1 Envolvimento do Analista de Teste em vários ciclos de vida de desenvolvimento de software

A organização das atividades de teste pode variar de acordo com o ciclo de vida de desenvolvimento de software (SDLC) que está sendo seguido. Portanto, o envolvimento do TA nas atividades de teste pode variar de acordo com o modelo de SDLC adotado.

Nos **modelos de desenvolvimento sequencial**, as atividades de desenvolvimento são realizadas em fases e começam quando a fase anterior é concluída. Normalmente, há pouca sobreposição entre essas atividades. Portanto, as tarefas do TA geralmente mudam com o tempo. Nas fases iniciais do SDLC, o TA concentra-se no suporte ao planejamento de testes. O TA inicia a análise do teste quando a base do teste está sendo produzida. O projeto e a implementação do teste seguem em paralelo com o projeto e a implementação do software. Por fim, o TA executa os testes e dá suporte à conclusão dos testes durante as fases finais do SDLC.

Os **modelos de desenvolvimento incremental** dividem o software em incrementos menores e gerenciáveis. Cada incremento é desenvolvido e testado de forma independente. Portanto, o TA executa as mesmas atividades para cada incremento (ou seja, análise de teste, modelagem de teste, implementação de teste, execução de teste e suporte ao gerenciamento de teste). Entretanto, o trabalho do TA pode ser organizado de forma diferente para cada incremento. Os testes se concentram nos recursos novos ou modificados. Além disso, devido ao aumento do risco de regressão, o TA deve prestar atenção especial à refatoração e à montagem de conjuntos de testes de regressão.

Nos **modelos de desenvolvimento iterativo**, o processo de desenvolvimento é cíclico. O projeto passa por ciclos repetidos de prototipagem, teste, refinamento e implementação. A função do TA é dinâmica e adaptável. O TA colabora estreitamente com os desenvolvedores e representantes do negócio, adaptando-se ao produto em evolução. O TA adapta e modifica as condições de teste e os casos de teste à medida que o software evolui e fornece feedback para melhorar o processo de teste a cada iteração. Quanto mais frequentes forem as iterações, mais críticos serão a manutenção e o desenvolvimento contínuos dos testes de regressão pelo TA.

Um SDLC pode combinar elementos de vários modelos e técnicas e abordagens específicas (p. ex., o desenvolvimento ágil de software combina aspectos dos modelos iterativo e incremental). Nesses casos, o envolvimento do TA dependerá das características específicas do SDLC e de como elas são combinadas. Uma boa prática comum a todos os modelos de SDLC é que o TA deve estar envolvido desde as fases iniciais do SDLC.

1.2 Envolvimento em atividades de teste

O Foundation Level Syllabus (ISTQB-CTFL, v4.0.1) descreve sete atividades dentro do processo de teste. A TA se concentra principalmente em quatro delas: análise de teste, modelagem de teste, implementação de teste e execução de teste.

As tarefas do TA nessas quatro atividades são descritas em detalhes nas seções a seguir.

©International Software Testing Qualifications Board

1.2.1 Análise de Teste

Durante a análise do teste, o TA verifica a integridade da base do teste e coleta qualquer informação adicional relevante para o teste. Isso inclui não apenas a documentação, mas também informações verbais, por exemplo, conversas na escrita colaborativa de histórias de usuários (consulte ISTQB-CTFL (v4.0.1), Seção 4.5.1). As alterações na base do teste podem levar a ajustes no escopo do teste em coordenação com o gerenciamento de testes.

Para prosseguir efetivamente com a análise do teste, o TA verifica os seguintes critérios de entrada:

- O planejamento do teste foi realizado e o escopo do teste, os objetivos do teste e a abordagem do teste estão claros.
- A base de teste (que contém informações como requisitos ou histórias de usuários) é definida.
- Os riscos do produto já identificados foram avaliados e documentados, se necessário.

O TA avalia a base de teste para identificar quaisquer defeitos que ela possa conter e avaliar sua testabilidade, fornecendo assim um feedback antecipado aos Product Owners (PO). Isso pode incluir a modelagem do comportamento do sistema de acordo com as técnicas de teste a serem aplicadas (consulte a Seção 3, Seção 5.2.1). As técnicas de revisão também são aplicadas como parte do processo (consulte a Seção 5.2.2). Se não forem corrigidos diretamente, os defeitos relacionados à base do teste deverão ser documentados. Além disso, o TA determina os oráculos de teste necessários (consulte a Seção 1.3.4).

O TA define e prioriza as condições de teste para cada item de teste no escopo. As condições de teste abordam os objetivos do teste (consulte ISTQB-CTFL (v4.0.1), Seção 1.1.1) e devem ser rastreáveis aos elementos da base de teste. O escopo e o foco das condições de teste levam em consideração os riscos do produto. Nos modelos de desenvolvimento incremental ou iterativo, isso inclui a determinação do escopo do teste de regressão com base em uma análise de impacto. No desenvolvimento ágil de software, as condições de teste podem ser expressas como critérios de aceite que refletem os riscos das histórias de usuários.

O TA pode prosseguir em etapas, começando com condições de teste de alto nível, como "funcionalidade da tela x". Em seguida, o TA define condições de teste mais detalhadas, como "A tela x rejeita números de conta com um dígito a menos". Essa abordagem oferece suporte a uma cobertura suficiente e permite um início antecipado do projeto de teste,

Por exemplo, para histórias de usuários que ainda precisam ser refinadas.

O TA envolve os stakeholders na revisão das condições de teste para garantir que a base do teste seja claramente compreendida e que o teste esteja alinhado com os objetivos do teste.

1.2.2 Modelagem de Teste

A modelagem de teste descreve como realizar o teste para atingir os objetivos de teste declarados. Normalmente, isso é feito com casos de teste. A maneira como a modelagem de teste é realizada depende de muitos fatores, incluindo a cobertura necessária, a base do teste, o SDLC, as restrições do projeto e o conhecimento e a experiência dos testadores.

Durante a modelagem de teste, o TA determina em quais áreas os casos de teste de baixo nível ou os casos de teste de alto nível são apropriados (consulte a Seção 1.3.1). Em ambos os casos, o TA deve identificar critérios claros de aprovação/reprovação. O TA projeta os casos de teste para as condições de teste novas ou alteradas de acordo com os critérios de qualidade (consulte a Seção 1.3.2). Para testes de regressão, a seleção de casos de teste de alto nível existentes ou a adaptação de casos de teste de baixo nível existentes com base em sua priorização geralmente é suficiente.

O TA captura a rastreabilidade entre a base do teste, as condições do teste e os casos de teste. Nos testes baseados na experiência, os casos de teste nem sempre são documentados; em vez disso, as condições de teste (entre outras) podem orientar a execução do teste. O TA pode construir alguns casos de teste com base em objetivos de teste de alto nível.

Além dessas tarefas, o TA define os requisitos do ambiente de teste (consulte a Seção 1.3.3) e identifica, cria e especifica os requisitos para os dados de teste (consulte a Seção 1.3.5).

©International Software Testing Qualifications Board

O TA usa os critérios de saída definidos no planejamento de testes para determinar quando foram modelados casos de teste suficientes. Entretanto, outros critérios de saída, como níveis de risco residual ou restrições do projeto (p. ex., orçamento ou tempo), também indicam quando o projeto de teste pode terminar.

O projeto de teste pode ser apoiado por ferramentas, mas deve ser independente de ferramentas e tecnologias para permanecer independente de ferramentas. O projeto de teste aplica uma abordagem sistemática usando técnicas de teste ou é ad hoc.

Os casos de teste têm uma função comunicativa e devem ser compreensíveis para os stakeholders. Como um caso de teste nem sempre pode ser executado por seu autor, outros testadores precisam entender como executá-lo, seus objetivos de teste (ou seja, suas condições de teste subjacentes) e sua importância. Os casos de teste também devem ser compreensíveis para os desenvolvedores, que podem implementar os testes ou executá-los novamente em caso de falha, e para os auditores, que podem ter de aprová-los.

1.2.3 Implementação do Teste

Durante a implementação do teste, o TA fornece o software de teste necessário para a execução do teste. O TA pode organizar os procedimentos de teste e os scripts de teste em conjuntos de testes ou sugerir casos de teste para automação. A definição dos procedimentos de teste requer a identificação cuidadosa das restrições e dependências que podem influenciar a ordem de execução do teste. Além das etapas contidas nos casos de teste, os procedimentos de teste incluem etapas para configurar quaisquer pré-condições iniciais (p. ex., carregar dados de teste de um repositório de dados), verificar os resultados esperados e as pós-condições e redefinir as etapas após a execução (p. ex., redefinir o banco de dados, o ambiente e o sistema).

O TA prioriza os procedimentos de teste e os scripts de teste para a execução do teste com base nos critérios de priorização identificados durante a análise de risco e o planejamento do teste e identifica os procedimentos de teste ou scripts de teste que devem ser executados na versão atual do objeto de teste. Isso permite que testes relacionados (p. ex., para novos recursos ou testes de regressão) sejam executados juntos em uma execução de teste específica. O TA atualiza a rastreabilidade entre a base de teste e outros materiais de teste, como procedimentos de teste, scripts de teste e conjuntos de testes.

O TA pode auxiliar a TM (Gerenciamento de Teste) na definição de um cronograma de execução de teste, incluindo a alocação de recursos, para permitir a execução eficiente, definindo a ordem de execução (consulte ISTQB-CTFL, v4.0.1, Seção 5.1.5).

O TA cria dados de entrada e de ambiente para carregar em bancos de dados e outros repositórios (consulte a Seção 1.3.5). Esses dados devem ser "adequados à finalidade" para apoiar os objetivos específicos do teste.

O TA também deve verificar se o ambiente de teste está totalmente configurado e pronto para a execução do teste (consulte a Seção 1.3.3). A melhor maneira de fazer isso é modelar e executar um *smoke test*. O ambiente de teste deve revelar defeitos no objeto de teste por meio da execução do teste, operar normalmente quando não ocorrerem falhas e replicar adequadamente, se necessário, o ambiente de produção ou do usuário final.

O nível de detalhe e a complexidade associada do trabalho realizado durante a implementação do teste podem ser influenciados pelo nível de detalhe das condições de teste e dos casos de teste. Em alguns casos, aplicam-se regras regulatórias, e o software de teste deve fornecer evidências de conformidade com os padrões aplicáveis (p. ex., RTCA DO- 178C, 2011).

1.2.4 Execução do Teste

A execução do teste é realizada de acordo com o cronograma de execução do teste. As tarefas típicas realizadas pelo TA são a execução de testes, a comparação dos resultados reais com os resultados esperados, a análise de anomalias, a comunicação de defeitos e o registro dos resultados dos testes.

O TA executa os testes manualmente. Isso inclui testes exploratórios, execução de procedimentos de teste, testes de regressão e testes de confirmação. Para testes exploratórios, o TA pode usar testes baseados em sessões com cartas de teste (consulte a Seção 3.4.1). O TA também pode executar scripts de testes automatizados, mas pode ser tarefa dos desenvolvedores, dos Engenheiros de Automação de Testes (TAE) ou dos Analista Técnico de Teste (TTA) executar scripts de testes automatizados e analisá-los em caso de falhas.

O TA analisa as anomalias que ocorrem nas execuções de testes manuais ou automatizados para estabelecer suas causas prováveis. Uma anomalia pode ser consequência de um defeito em um objeto de teste. Ainda assim,

©International Software Testing Qualifications Board

outros motivos também podem existir, incluindo pré-condições ausentes, dados de teste incorretos, defeitos nos scripts de teste ou no ambiente de teste, ou má compreensão das especificações. O TA registra os resultados reais da execução do teste, comunica os defeitos com base nas falhas observadas e faz relatórios sobre eles, se necessário.

O TA atualiza a rastreabilidade entre a base de teste e outros equipamentos de teste, considerando os resultados do teste. Essas informações possibilitam a transformação dos resultados dos testes em informações de alto nível sobre riscos ou cobertura, o que permite que os stakeholders tomem decisões. Por exemplo, isso esclarece aos stakeholders quantos casos de teste relacionados a uma condição de teste foram aprovados ou reprovados.

Além dessas tarefas típicas, o TA avalia os resultados dos testes, incluindo as seguintes tarefas:

- Reconhecer grupos de defeitos, o que pode indicar a necessidade de mais testes de uma parte específica do objeto de teste (consulte a Seção 5.3.1).
- Reexecutar manualmente os testes automatizados que falharam para garantir que a automação do teste não produziu um resultado falso-positivo.
- Sugerir testes adicionais com base no que foi aprendido nos testes anteriores.
- Identificar novos riscos a partir de informações obtidas durante a execução do teste.
- Sugerir melhorias no projeto ou na implementação do teste (p. ex., melhorias nos procedimentos de teste) ou até mesmo no sistema em teste.
- Sugerir melhorias nos conjuntos de testes de regressão, incluindo refatoração, ajustes de escopo e automação de testes (consulte a Seção 2.2.1).

1.3 Tarefas relacionadas aos produtos de trabalho

O TA deve garantir a qualidade dos produtos de trabalho pelos quais é responsável. Isso inclui casos de teste, ambientes de teste, dados de teste, oráculos de teste e scripts de teste. Nesta seção, o syllabus discute as tarefas do TA relacionadas ao material de teste e os tipos de ferramentas para gerenciar o material de teste.

1.3.1 Casos de Teste de Alto Nível e Casos de Teste de Baixo Nível

Um caso de teste de alto nível (também chamado de caso de teste abstrato ou caso de teste lógico) descreve as circunstâncias em que o objeto de teste é examinado, indicando quais condições de teste são cobertas pelo caso de teste. Os casos de teste de alto nível são, portanto, adequados para garantir que os testes abranjam todas as condições de teste relevantes. Os casos de teste de alto nível não contêm informações concretas sobre pré-condições, dados de entrada, resultados esperados ou pós-condições. Todos eles são expressos em um nível abstrato (p. ex., "encomendar mais de um livro, com o preço do pedido resultando em um desconto; resultado esperado: o desconto é atribuído").

Um caso de teste de baixo nível (também chamado de caso de teste concreto ou caso de teste físico) é o refinamento detalhado de um caso de teste de alto nível. Os casos de teste de baixo nível descrevem quais dados precisam ser preparados, quais ações o testador deve realizar (se necessário) e qual é o resultado concreto esperado. Os casos de teste de baixo nível contêm pré-condições específicas, dados de entrada, resultados esperados e pós-condições (p. ex., "encomendar livros B1 (R$20) e B2 (R$30), com preço total do pedido de R$50; resultado esperado: 10% de desconto atribuído, preço total: R$45").

Normalmente, um TA inicialmente modela casos de teste de alto nível, que são a base para o desenvolvimento de casos de teste de baixo nível. Um caso de teste de alto nível pode ser implementado em um ou mais casos de teste de baixo nível. Às vezes, o caso de teste pode permanecer em alto nível, com as informações concretas determinadas pelo TA durante a execução do teste.

Por exemplo, os casos de teste de alto nível podem orientar o TA ao criar objetivos de teste em uma carta de teste, para testes baseados em sessão, permitindo que os TA elaborem esses objetivos durante a execução do teste (consulte a Seção 3.4.1).

A transição de casos de teste de alto nível para baixo nível é mais do que apenas preencher valores concretos. É também uma etapa conceitual para o técnico. Geralmente, ela é adiada do projeto de teste para a implementação do teste, especialmente se forem necessários dados de teste específicos. O TA deve garantir que tudo o que for

©International Software Testing Qualifications Board

necessário para executar os casos de teste de baixo nível seja conhecido (Koomen et al., 2006). Na prática, muitos casos de teste são híbridos, sendo concretos em alguns aspectos e abstrato em outros. Isso geralmente se deve a uma compensação entre a capacidade de manutenção e a compreensibilidade dos casos de teste.

1.3.2 Critérios de Qualidade para Casos de Teste

Negligenciar a qualidade do caso de teste pode levar a muitos problemas, como altos custos de manutenção, compreensão reduzida ou atrasos na execução. Os critérios de qualidade para os casos de teste são a primeira etapa para a obtenção de casos de teste mais fáceis de manter. Eles incluem:

- **Correção**. Um caso de teste deve facilitar a verificação precisa das condições de teste nas quais ele se baseia.
- **Viabilidade**. Deve ser possível executar um caso de teste.
- **Necessidade**. Cada caso de teste deve abranger um objetivo de teste claro, conforme expresso em seu título ou resumo. As duplicidades devem ser evitadas. Coisas que não devem ser testadas não devem ter casos de teste modelados.
- **Compreensibilidade**. Os casos de teste podem ser revisados, modificados e executados por pessoas que não sejam o autor. O TA deve escrever casos de teste em uma linguagem e um formato compreensíveis para todos os stakeholders envolvidos, sem explicar o óbvio. Os casos de teste complexos devem ser simplificados ou divididos.
- **Rastreabilidade**. Os casos de teste devem ser rastreáveis às condições de teste, aos requisitos e aos riscos para permitir que o TA os mantenha atualizados (consulte ISTQB-CTFL (v4.0.1), Seção 1.4.4).
- **Consistência**. A consistência na linguagem, na formatação e na estrutura facilita a compreensão e a manutenção dos casos de teste. O TA pode usar um glossário para essa finalidade.
- **Precisão**. Deve haver apenas uma interpretação de um caso de teste para evitar resultados de teste falso-negativos e falso-positivos. Termos ambíguos como "adequado", "conforme necessário" ou "vários" devem ser evitados.
- **Completude**. Todos os atributos necessários (p. ex., consulte ISO/IEC/IEEE 29119-3, 2021) devem estar presentes, inclusive os dados de teste necessários (consulte a Seção 1.3.5) e um resultado esperado claro para evitar dúvidas na comparação com o resultado real.
- **Concisão**. A granularidade dos casos de teste (ou seja, um caso de teste grande com muitas ações de teste versus vários casos de teste menores) deve corresponder à base e às condições do teste. É preferível ter casos de teste menores focados em alguns itens de cobertura, pois eles facilitam a localização das causas das falhas, podem ser combinados de forma flexível em procedimentos e conjuntos de testes, e uma falha durante a execução do teste não bloquearia nenhum outro teste.

O formato e o nível de detalhes dos casos de teste dependem do contexto do projeto e do produto e devem ser acordados entre a equipe de teste.

1.3.3 Requisitos do Ambiente de Teste

O ambiente de teste é um fator crítico de sucesso para a execução de testes manuais e automatizados. A implementação do ambiente de teste afeta a capacidade de teste, a detecção de defeitos, os custos gerais de teste e a confiabilidade dos resultados do teste. Um caso de teste que passa ou falha no ambiente de teste deve ter o mesmo resultado quando executado na produção. O ideal é que um ambiente de teste seja robusto, previsível e integrado à estrutura de automação de testes, se necessário. O TA pode definir os requisitos do ambiente de teste durante o projeto do teste com base na análise de:

- condições de teste, casos de teste e requisitos de dados de teste, de modo que os requisitos do ambiente de teste descrevam as condições necessárias para configurar e manter o ambiente de teste para garantir que as condições prévias da execução do teste sejam atendidas
- níveis de teste e tipos de teste, que influenciam o equilíbrio entre a flexibilidade do ambiente de teste e a semelhança com o ambiente de produção

©International Software Testing Qualifications Board

- disponibilidade e independência de componentes e sistemas, o que pode indicar a necessidade de usar dublês de teste (p. ex., stubs ou drivers)

Os requisitos do ambiente de teste descrevem os itens do ambiente de teste, que podem ser categorizados em vários tipos, como hardware, middleware, software, serviços virtualizados, rede, interfaces, ferramentas, segurança, configuração e local. Para cada item do ambiente de teste, os requisitos devem incluir as seguintes informações (ISO/IEC/IEEE 29119-3, 2021):

- Identificador exclusivo - usado para fins de rastreabilidade.
- Descrição - com detalhes suficientes para implementá-la conforme necessário.
- Responsabilidade - descreve quem é responsável por disponibilizá-lo.
- Período necessário - identifica quando e por quanto tempo o item é necessário.
- Fidelidade - o grau em que esse item representa ou se desvia do ambiente de produção.

Os requisitos também devem abordar as necessidades gerais do ambiente de teste, incluindo configuração, backup e restauração, necessidades de segurança, capacidade de alterar o ambiente de teste e funções e autorização (Koomen et al., 2006).

O TA deve documentar os requisitos do ambiente de teste de forma clara, compacta e coerente. O TA pode usar diagramas ou tabelas. Para evitar redundâncias e documentação desnecessária, o TA pode fazer referência a ambientes de teste existentes e se concentrar nas necessidades específicas do nível de teste. Os stakeholders relevantes (p. ex., desenvolvedores, TTA, engenheiros de automação de testes, analistas de negócios, patrocinadores e Product Owner (PO)) também devem revisar, aprovar e atualizar os requisitos do ambiente de teste.

1.3.4 Determinação de Oráculos de Teste

É necessário um oráculo de teste para determinar os resultados esperados nos testes dinâmicos. O ideal é que a base do teste forneça os oráculos de teste (p. ex., em uma especificação textual ou formal). A experiência humana ou o conhecimento sobre o objeto de teste também podem servir como oráculo de teste. Um oráculo de teste automatizado pode ser necessário por motivos de custo-benefício (p. ex., se as entradas forem geradas automaticamente ou se os oráculos humanos forem caros).

Dependendo da qualidade e da integridade da base de teste ou das características do sistema, um oráculo de teste econômico pode não estar disponível. Isso é conhecido como o problema do oráculo de teste. Os fatores típicos que contribuem para o problema do oráculo de teste incluem complexidade relacionada aos dados, não determinismo (p. ex., sistemas baseados em IA), comportamento probabilístico e requisitos ausentes ou ambíguos.

Algumas soluções conhecidas para o problema do oráculo de teste são (Barr et al., 2014):

- Os pseudo-oráculos são sistemas desenvolvidos de forma independente que atendem à mesma especificação do objeto de teste (p. ex., sistemas legados, versões simplificadas de objetos de teste). O desenvolvimento de um pseudo-oráculo dedicado para um objeto de teste complexo pode ser caro, mas é comum em sistemas críticos.
- O Teste Baseado em Modelo pode formalizar o oráculo de teste como parte do modelo de teste. Ele permite a geração de resultados esperados e a derivação de testes a partir do modelo. As técnicas de teste baseado no comportamento geralmente envolvem a implementação de um oráculo de teste automatizado (consulte a Seção 3.2.2 e a Seção 3.2.3).
- O Teste Baseado em Propriedade usa propriedades do objeto de teste especificadas para verificar as relações entre a entrada e o resultado esperado de casos de teste individuais. Se essa relação não for atendida, o caso de teste falha. Essa solução é adequada para a automação de testes, mas sua eficácia depende das relações, que podem ser difíceis de determinar.
- Testes Metamórficos (consulte a Seção 3.3.2).
- Os oráculos humanos usam a capacidade dos seres humanos para determinar os resultados esperados. Os recursos humanos podem ser caros e escassos. Entretanto, os oráculos humanos são preferidos em algumas abordagens de teste (p. ex., teste exploratório).

©International Software Testing Qualifications Board

As asserções podem ser incorporadas ao código de automação de teste ou ao próprio objeto de teste para implementar um oráculo de teste automatizado. Elas são declarações executáveis que verificam o estado ou o comportamento do objeto de teste. Quando incorporadas ao objeto de teste, elas geralmente verificam apenas o que é necessário para a continuação da tarefa.

1.3.5 Requisitos de Dados de Teste

Durante o projeto do teste, o TA identifica e solicita dados de teste que podem precisar de preparação ou provisionamento para a execução do teste. As considerações incluem finalidade, formato e contexto de uso. (ISO/IEC/IEEE 29119-3, 2021, Seção 8.5) também descreve os requisitos de dados de teste. Os principais aspectos são:

- **Similaridade com os dados de produção**. Os dados de produção refletem os dados do mundo real, mas podem não ter variabilidade. Os dados sintéticos permitem a variabilidade controlada. Eles devem refletir os principais aspectos dos padrões, distribuições e valores atípicos dos dados de produção, mas podem ignorar certos defeitos que só ocorrem com dados do mundo real. Para sistemas que não possuem dados de produção, os dados sintéticos devem refletir cenários realistas técnicos e de negócio. As personas ajudam fornecendo perfis realistas e centrados no usuário que orientam a criação de dados que refletem diversos cenários e comportamentos de usuários.
- **Confidencialidade**. Dados de teste confidenciais (p. ex., informações pessoais) exigem proteção. Os dados pseudonimizados substituem as informações pessoais por identificadores artificiais. Os dados anônimos removem informações identificáveis sobre indivíduos. Se necessário, o TA deve observar os regulamentos de proteção de dados, como o GDPR na União Europeia (Comissão, 2016), o HIPAA nos EUA (Health et al., 2024), ou a LGPD no Brasil (ANPD, 2023)
- **Objetivo**. Os dados de teste são essenciais para determinar as condições prévias e os resultados esperados que afetam o estado do sistema e sua configuração (p. ex., data e hora do sistema). Também incluem a especificação de permissões de usuário e o estabelecimento de relações entre produtos, departamentos e categorias.
- **Critérios de cobertura**. Os dados de teste devem estar alinhados com os critérios de cobertura da técnica de teste escolhida. Além de dados de teste válidos, isso também pode exigir dados de teste inválidos, como para testes negativos.
- **Formato dos dados**. O gerenciamento sistemático de dados (p. ex., em testes de API) pode exigir dados estruturados (p. ex., CSV, JSON, XML ou banco de dados).
- **Rastreabilidade**. A rastreabilidade garante a manutenção dos dados de teste quando são feitas alterações nos casos de teste.
- **Capacidade de manutenção**. Os dados de teste codificados em casos de teste de baixo nível devem ser evitados para facilitar a detecção de defeitos e a manutenção. Os casos de teste devem separar a lógica de teste dos dados de teste (consulte a Seção 1.3.2).
- **Dependências**. Os dados dependentes requerem uma série de etapas para serem criados.
- **Disponibilidade**. A virtualização de serviços pode tratar de dados ausentes simulando serviços ausentes ou inacessíveis para interagir com sistemas ou serviços externos.
- **Sensibilidade ao tempo e envelhecimento dos dados**. Os casos de teste que envolvem dados desatualizados ou sensíveis ao tempo podem afetar o comportamento do sistema de forma inesperada ou imprecisa.

1.3.6 Desenvolvimento de Scripts de Teste usando Teste Orientado por Palavras-Chave

Se o Teste Orientado por Palavras-Chave for usado, o TA criará scripts de teste usando palavras-chave. A implementação dos scripts de teste é tarefa do TTA, do TAE ou de um desenvolvedor.

O TA identifica e especifica as palavras-chave analisando a base do teste ou colaborando com os stakeholders. As palavras-chave podem ser classificadas em duas categorias: ação e verificação. As palavras-chave de ação devem interagir com o objeto de teste (p. ex., execução de funções, envio de dados, navegação dentro do objeto

©International Software Testing Qualifications Board

de teste pode ser usado para verificar se o resultado real produzido pelo objeto de teste corresponde ao resultado esperado. As palavras-chave de verificação representam afirmações para avaliar se o resultado real produzido pelo objeto de teste corresponde ao resultado esperado.

As palavras-chave residem em pelo menos duas camadas de abstração: a camada de domínio e a camada de interface de teste. As palavras-chave da camada de domínio correspondem a ações relacionadas aos negócios e refletem a terminologia do domínio do aplicativo. Elas abstraem os detalhes técnicos da interface do objeto de teste. As palavras-chave da camada de interface de teste se comunicam com os objetos de teste, itens do ambiente de teste ou outros componentes ou sistemas por meio de sua interface de teste. Elas residem na camada de abstração mais baixa. Camadas intermediárias adicionais podem dar suporte à capacidade de manutenção das palavras-chave.

As palavras-chave podem ser atômicas ou compostas de outras palavras-chave. A estrutura e a camada de abstração de uma palavra-chave são atributos independentes. No entanto, as palavras-chave compostas geralmente residem em um nível mais alto de abstração, enquanto as palavras-chave atômicas tendem a residir na camada de interface de teste.

As tarefas do TA em Teste Orientado por Palavras-Chave incluem:

- Especificar de palavras-chave e seus parâmetros.
- Especificar os casos de teste de palavras-chave, ou seja, scripts de teste que usam palavras-chave.
- Especificar as etapas adicionais para os scripts de teste usando palavras-chave como condições prévias, ações de verificação e limpeza do ambiente de teste após o teste.
- Manutenção de casos de teste de palavras-chave para refletir as alterações no objeto de teste.
- Execução de scripts de teste de palavras-chave, automatizados ou manuais.
- Análise de casos de teste de palavras-chave com falha para determinar a causa da falha.

Ao analisar a base de teste, o TA procura interações entre o objeto de teste e seu ambiente (p. ex., usuários, outros sistemas e dispositivos). Considere, por exemplo, a história do usuário "*Como membro, quero me autenticar para ter acesso às instalações*", com os critérios de aceite "*Cartões de membro válidos podem ser usados para autenticação*". O TA pode especificar uma palavra-chave (ação) '*autenticar membro*' com um parâmetro '*cartão de membro*' e uma palavra-chave de verificação '*verificar acesso*'. A TA verifica se as palavras-chave identificadas residem na camada de abstração apropriada.

Ao especificar palavras-chave, o TA deve ter em mente que as palavras-chave devem:

- conter um verbo (+ substantivo);
- usar a forma imperativa do verbo (+ substantivo);
- ser únicos em seu significado;
- ser adequadamente documentado;
- refletir o vocabulário do domínio do aplicativo;
- ser reutilizável.

Ao compor casos de teste com palavras-chave, o TA pode reconhecer algumas palavras-chave ausentes e especificá-las imediatamente. Os casos de teste de palavras-chave podem ser escritos em vários formatos, como listas ou tabelas.

As palavras-chave podem mudar durante a vida útil de um projeto e são propensas a serem especificadas de forma redundante (Rwemalika et al., 2019). As recomendações mencionadas nesta seção visam evitar a redundância e reduzir os esforços de manutenção.

Embora o Teste Orientado por Palavras-Chave seja uma abordagem de automação de teste, o teste manual também pode se beneficiar dele. Se for aplicado para testes manuais, ele apoia efetivamente uma transição posterior do teste manual para o automatizado.

Mais detalhes sobre automação da execução de testes e testes orientados por palavras-chave estão disponíveis em (ISTQB-TTA, v4.0), (ISTQB-TAE, v2.0) e (ISO/IEC/IEEE 29119-5, 2016).

©International Software Testing Qualifications Board

1.3.7 Ferramentas aplicadas no Gerenciamento do Testware

O gerenciamento adequado do testware por meio de ferramentas pode ajudar o TA a apoiar o processo geral de teste. As ferramentas podem fornecer o status dos produtos de trabalho e apoiar o monitoramento e o controle dos testes. Em caso de falhas no sistema em teste, elas permitem que o TA verifique os resultados de testes anteriores e analise o momento em que os defeitos ocorreram.

Alguns tipos importantes de ferramentas para gerenciar o material de teste incluem:

- **Ferramentas de gerenciamento de testes** para fornecer um repositório de todo o material de teste relevante (p. ex., condições de teste, casos de teste, scripts de teste, conjuntos de teste e execuções de teste). As ferramentas facilitam a rastreabilidade (p. ex., por meio da matriz de rastreabilidade), a recuperação de casos de teste, o agendamento de execuções de teste, o registro dos resultados do teste e o relatório geral do progresso e da qualidade do teste.
- **Ferramentas de gerenciamento de defeitos** para registrar, priorizar e monitorar o processo de resolução de defeitos.
- **Ferramentas de gerenciamento de dados de teste** para criar e manter dados de teste, incluindo a proteção de dados confidenciais (consulte a Seção 1.3.5).
- **Ferramentas de gerenciamento de configuração** para facilitar as atividades relacionadas a testes nos processos de desenvolvimento, lançamento e operação, incluindo o gerenciamento da configuração e da disponibilidade dos ambientes de teste.
- **Ferramentas de gerenciamento de requisitos** para conter e rastrear requisitos de alto nível, garantindo que eles sejam claramente definidos, versionados e rastreados durante todo o SDLC.

O TA oferece suporte ao gerenciamento do testware:

- analisar o projeto e a versão para selecionar o subconjunto correto de testware (p. ex., uma especificação identificada por sua versão para corresponder à versão do SUT)
- definir uma estrutura funcional (p. ex., por recursos ou módulos) ou técnica (p. ex., por tipo de teste ou ambiente) apropriada para a organização dos casos de teste na ferramenta de gerenciamento de testes
- adicionar metadados aos casos de teste (p. ex., informações sobre o esforço relacionado à execução do teste ou ao ambiente de teste específico necessário)
- garantir a rastreabilidade entre requisitos, condições de teste, testes, execuções de teste e defeitos
- selecionar os conjuntos de testes corretos para testes de regressão (para execução manual/automatizada de testes)
- gerenciamento da configuração dos casos de teste, incluindo a identificação de casos de teste desatualizados.

As ferramentas ajudam a organizar, rastrear e garantir a qualidade do processo de teste. O TA apoia esse esforço selecionando, estruturando e mantendo os casos de teste, garantindo a rastreabilidade e gerenciando as versões dos casos de teste para um teste eficiente e controle de teste.

©International Software Testing Qualifications Board

2 As tarefas do Analista de Teste em Teste Baseado em Risco – 90 min

Palavras-chave

análise de impacto, risco do produto, teste de regressão, análise de risco, avaliação de risco, controle de risco, identificação de risco, mitigação de risco, monitoramento de risco, teste baseado em risco

Objetivos de aprendizagem para o Capítulo 2:

2.1 Análise de risco

TA-2.1.1 (K2) Resumir a contribuição do Analista de Teste para a análise de riscos do produto

2.2 Controle de riscos

TA-2.2.1 (K4) Analisar o impacto das mudanças para determinar o escopo do teste de regressão

Introdução

O Teste Baseado em Risco é uma abordagem de teste que prioriza os esforços de teste com base nos níveis de risco dos itens de teste. O gerente de testes determina essa abordagem. O TA desempenha um papel ativo na implementação. O teste baseado em risco é abordado em mais detalhes no (ISTQB-TM, v3.0).

2.1 Análise de Risco

De acordo com o (ISTQB-CTFL, v4.0.1, Seção 5.2), a análise de riscos envolve a identificação e a avaliação de riscos. Este syllabus se concentra em como o TA deve participar das atividades de análise de riscos para garantir que os testes baseados em riscos sejam implementados corretamente.

2.1.1 A contribuição do Analista de Teste para a Análise de Risco do Produto

O TA geralmente possui um conhecimento exclusivo do sistema, além de experiência e conhecimento intuitivo do que geralmente dá errado, do impacto que isso causa e de como os testes podem reduzir o risco. Isso torna o TA um valioso stakeholder para a análise de risco do produto.

Na identificação de riscos, o TA contribui com sua própria experiência e conhecimento para retrospectivas, workshops sobre riscos, brainstorming e criação de listas de verificação. O TA também pode realizar entrevistas com os stakeholders para entender melhor o que elas consideram ser os riscos mais significativos do ponto de vista delas.

Durante a avaliação de riscos, o TA, juntamente com outros stakeholders, contribui para a determinação do nível de risco estimando vários fatores, como:

- frequência de uso e importância crítica dos recursos afetados;
- criticidade dos objetivos comerciais afetados;
- danos financeiros, ambientais e à reputação;
- qualidade da base de teste;
- necessidades legais ou de segurança.

O TA também ajuda a categorizar os riscos do produto pelas características de qualidade afetadas, por exemplo, usando o modelo de qualidade do produto da ISO/IEC 25010 (2023). O nível de risco geralmente não é distribuído

©International Software Testing Qualifications Board

uniformemente pelo objeto de teste. Nesses casos, o TA deve dividir o objeto de teste em itens de teste (p. ex., componentes, interfaces e recursos) e avaliar um determinado risco para cada item de teste separadamente.

Por fim, como parte da avaliação de risco do produto, o TA propõe atividades de teste adequadas para mitigar cada risco identificado do produto. Essas atividades podem incluir testes estáticos e testes dinâmicos. Dependendo de fatores como o nível de risco, o tipo de item de teste associado e a característica de qualidade afetada, o TA indica os níveis de teste necessários, os tipos de teste, as técnicas de teste, os níveis de independência dos testes e o rigor dos testes. No espírito do shift left, o TA indica quais atividades de teste podem mitigar o risco mais cedo para minimizar o esforço de teste.

2.2 Controle de Risco

De acordo com o (ISTQB-CTFL, v4.0.1, Seção 5.2.4), o controle de riscos envolve a mitigação e o monitoramento de riscos. O TA compreende as funcionalidades do sistema e os possíveis riscos relacionados a elas. Portanto, o papel do TA é crucial em várias ações de mitigação de riscos mencionadas no (ISTQB-CTFL, v4.0.1, Seção 5.2.4):

- A realização de revisões é discutida na Seção 5.2.2 deste syllabus.
- A aplicação das técnicas de teste e dos níveis de cobertura adequados é discutida na Seção 3.5.
- A aplicação dos tipos de teste adequados é discutida no Capítulo 4.
- A realização de testes de regressão é discutida a seguir.

Além disso, como os riscos e os níveis de risco não são estáticos e mudam com o tempo, os testes baseados em risco envolvem o monitoramento regular dos riscos. Em um ciclo de vida iterativo, isso é realizado em uma frequência determinada pela equipe (provavelmente uma vez por iteração). Em outros ciclos de vida, sua frequência é definida pela pessoa responsável pelo gerenciamento de riscos do produto, geralmente o gerente de testes. O TA contribui atualizando o registro de riscos com base nas alterações feitas e ajustando as ações de mitigação de riscos mencionadas acima.

2.2.1 Determinação do escopo do Teste de Regressão

O principal objetivo do teste de regressão é garantir a confiança na qualidade do objeto de teste depois que uma alteração é feita. No entanto, pode ser impossível executar todas as regressões durante o Teste de Regressão devido a restrições (p. ex., restrições de tempo, orçamento, ambiente de teste ou dados de teste). Essa é uma preocupação principal com os testes executados manualmente, mas também se aplica aos testes automatizados, por exemplo, quando os ciclos de teste são curtos e há muitos testes automatizados de longa duração. Isso torna necessário selecionar testes de regressão apropriados com base em critérios específicos. É essencial revisar o escopo dos testes de regressão a cada ciclo de teste. A revisão pode concluir que o conjunto de testes de regressão existente precisa ser ajustado.

A técnica mais confiável para selecionar testes automatizados é uma análise de impacto, que pode ser apoiada por ferramentas. Essas ferramentas são baseadas em um sistema de gerenciamento de configuração automatizado. Elas registram quais itens de configuração são ativados durante a execução de cada caso de teste. Quando ocorre uma alteração, elas rastreiam quais itens de configuração foram modificados e selecionam testes de regressão que interagem com os itens alterados. Ao fazer isso, as ferramentas garantem que, após uma alteração em um item de configuração, os testes se concentrem nas áreas com maior probabilidade de encontrar falhas (Juergens et al., 2018).

No que diz respeito à execução de testes manuais, não há evidências conclusivas de uma técnica de seleção de testes de regressão que seja claramente superior, pois os resultados dependem de vários fatores (Engström et al., 2010). Portanto, o TA deve decidir qual técnica usar com base na situação em questão. As técnicas comumente usadas incluem:

- **Teste Baseado em Risco**, em que o TA mantém a rastreabilidade do conjunto de testes de regressão para um registro de riscos. Quando uma alteração é feita e o registro de riscos é atualizado de acordo, o TA ajusta o conjunto de testes de regressão para cobrir os níveis de risco mais altos.
- **Teste Baseado em Histórico**, em que o TA avalia as execuções de testes anteriores e determina quais expuseram defeitos ou foram sensíveis a alterações semelhantes àquelas feitas desde a última execução

©International Software Testing Qualifications Board

do teste. A execução dos testes correspondentes novamente como parte do teste de regressão aumenta a probabilidade de exposição de defeitos semelhantes. O TA também pode incluir alguns testes que não foram executados por um longo período para garantir que ainda sejam aprovados.

- **Teste Baseado na Cobertura**, em que o TA seleciona um pequeno número de testes que atingem o máximo de cobertura possível com base na(s) técnica(s) de teste escolhida(s). A quantidade de testes deve ser cuidadosamente equilibrada com o aumento da cobertura por teste.
- **Matriz de Rastreabilidade de Requisitos**, que é usada para avaliar o impacto das alterações de requisitos nos testes associados. Ela pode ser particularmente valiosa quando requisitos novos ou alterados indiretamente afetam os recursos existentes. O TA seleciona testes de regressão para os recursos diretamente afetados e para os recursos relacionados para cobrir possíveis efeitos colaterais não intencionais. No desenvolvimento ágil de software, isso pode ser feito de forma semelhante, selecionando testes que cubram os critérios de aceite afetados por histórias de usuários novas ou alteradas.
- **Teste Baseado em Perfil Operacional**, em que o TA seleciona os casos de teste de regressão a serem executados com base nos padrões de uso do objeto de teste. Por exemplo, ao testar uma loja on-line, um desses testes pode incluir o login do usuário, a busca de produtos, a adição deles ao carrinho e a realização do pedido. Quando um aplicativo é alterado significativamente, essa técnica oferece uma visão geral rápida da funcionalidade geral do sistema. Se isso resultar em muitos testes, o TA priorizará padrões críticos de uso que ocorrem com frequência e abrangem funcionalidades e processos de negócio essenciais.
- **Análise de Impacto**, que também pode ser usada para selecionar casos de teste de regressão para execução manual se o TA souber quais deles interagem com os itens de configuração alterados.

Muitas vezes, é necessário usar uma combinação de técnicas de seleção para obter um conjunto de testes de regressão mais abrangente e eficaz. No entanto, o TA deve equilibrar cuidadosamente a necessidade de cobertura com um tamanho gerenciável do conjunto de testes. Após cada ciclo de teste, o TA analisa os resultados do teste para determinar a eficácia das técnicas aplicadas (consulte a Seção 5.3.1). Na próxima vez, o TA mantém as técnicas eficazes e substitui as ineficazes. Isso melhora continuamente a seleção de testes de regressão ao longo do tempo. É essencial em modelos de desenvolvimento iterativos e incrementais, em que as alterações são frequentes.

©International Software Testing Qualifications Board

3 Análise e Modelagem de Teste - 615 min

Palavras-chave

teste baseado em lista de verificação, técnica de teste baseado em comportamento, teste combinatório, teste de multidão, teste CRUD, técnica de teste baseado em dados, teste de tabela de decisão, teste de domínio, partição de equivalência, teste baseado na experiência, relação metamórfica, teste metamórfico, teste aleatório, técnica de teste baseado em regra, teste baseado em cenário, teste baseado em sessão, teste de transição de estado, carta de teste

Objetivos de aprendizagem para o Capítulo 3:

3.1 Técnicas de Teste Baseado em Dados

TA-3.1.1 (K3) Aplicar testes de domínio

TA-3.1.2 (K3) Aplicar testes combinatórios

TA-3.1.3 (K2) Resumir os benefícios e as limitações dos testes aleatórios

3.2 Técnicas de Teste Baseado em Comportamento

TA-3.2.1 (K2) Explicar o teste CRUD

TA-3.2.2 (K3) Aplicar testes de transição de estado

TA-3.2.3 (K3) Aplicar teste baseado em cenário

3.3 Técnicas de Teste Baseado em Regras

TA-3.3.1 (K3) Aplicar testes de tabela de decisão

TA-3.3.2 (K3) Aplicar testes metamórficos

3.4 Teste Baseado na Experiência

TA-3.4.1 (K3) Preparar cartas de teste para testes baseados em sessões

TA-3.4.2 (K3) Preparar listas de verificação que apoiem testes baseados em experiência

TA-3.4.3 (K2) Dar exemplos dos benefícios e limitações dos testes de multidão

3.5 Aplicação das técnicas de teste mais apropriadas

TA-3.5.1 (K4) Selecionar técnicas de teste adequadas para mitigar os riscos do produto em uma determinada situação

TA-3.5.2 (K2) Explicar os benefícios e os riscos de automatizar o projeto de teste

©International Software Testing Qualifications Board

Introdução

As técnicas de teste são usadas principalmente na análise e na modelagem de teste. As técnicas de teste discutidas neste capítulo abrangem técnicas de teste caixa-preta e técnicas de teste baseadas na experiência. As técnicas de teste caixa-branca são discutidas em (ISTQB-TTA, v4.0).

Este syllabus subdivide as técnicas de teste caixa-preta em três categorias com base nas condições de teste subjacentes que modelam:

- elementos dos dados (teste baseado em dados)
- elementos do comportamento dinâmico (teste baseado no comportamento)
- elementos de regras de comportamento estático (teste baseado em regras)

3.1 Técnicas de Teste Baseado em Dados

Nesta seção, o termo "domínio" é usado para representar o conjunto de dados de entrada de um item de teste. Os dados de entrada de várias áreas do domínio levam a vários comportamentos do item de teste. As técnicas de teste baseadas em dados visam verificar se a implementação lida corretamente com áreas de domínio específicas. O Foundation Level Syllabus (ISTQB-CTFL, v4.0.1, Seção 4.2) abrange duas técnicas de teste que podem ser categorizadas como baseada em dados: particionamento de equivalência (EP) e análise de valor limite (BVA). Este syllabus apresenta três outras técnicas de teste baseadas em dados:

- teste de domínio - estende o EP e o BVA a domínios com vários parâmetros e partições complexas
- teste combinatório - concentra-se nas interações de vários parâmetros em um domínio multidimensional
- teste aleatório - seleciona entradas aleatórias do domínio com base em uma distribuição de probabilidade especifíca

Observe que os testes aleatórios podem se referir tanto a dados de entrada aleatórios quanto a eventos aleatórios. Este syllabus trata apenas de testes com dados de entrada aleatórios.

3.1.1 Teste de Domínio

O teste de domínio verifica se o item de teste se comporta conforme especificado nas partições de equivalência do domínio e em suas bordas. Nesse caso, as partições de equivalência são definidas usando expressões que combinam condições atômicas por meio de operadores booleanos (p. ex., AND, OR e NOT) e envolvem uma ou mais variáveis de interação. Cada condição atômica define uma borda da partição de equivalência. As bordas fechadas são formadas por expressões relacionais com os operadores **≤**, **≥** ou **=**, e as fronteiras abertas são formadas por expressões relacionais com os operadores **<**, **>** ou **/=**

Por exemplo, a expressão altura **>** 1,29 metros AND largura **/** altura^2 **≥** 30 define uma partição de equivalência de um domínio bidimensional de duas variáveis que tem duas bordas, uma aberta e outra fechada.

O teste de domínio busca identificar defeitos nas implementações da partição de equivalência (p. ex., operador ou constante incorreta), selecionando itens de cobertura apropriados que possam revelar esses defeitos. Os critérios de cobertura nos testes de domínio referem-se aos pontos ON, OFF, IN e OUT:

- Para uma borda fechada, um ponto ON está nela. Para uma borda aberta, um ponto ON está dentro da partição de equivalência e é o mais próximo da borda de acordo com a precisão fornecida.
- Para uma borda fechada, um ponto OFF fica fora da partição de equivalência e é o mais próximo da borda de acordo com a precisão fornecida. Para uma borda aberta, um ponto OFF fica nessa borda.
- Um ponto IN pertence à partição de equivalência e não é um ponto ON.
- Um ponto OUT está fora da partição de equivalência e não é um ponto OFF.

Cada um desses tipos de pontos está relacionado à borda da partição de equivalência. O mesmo ponto pode ter tipos diferentes para bordas diferentes.

Os critérios de cobertura incluem:

©International Software Testing Qualifications Board

Cobertura de domínio simplificada (Jeng et al., 1994), que requer os seguintes itens de cobertura:

- Um ponto ON e um ponto OFF para cada borda definida por um dos operadores **<**, **≤**, **>** ou, **≥**
- Um ponto ON e dois pontos OFF localizados em lados diferentes da fronteira para o operador **=**.
- Um ponto OFF e dois pontos ON localizados em lados diferentes da fronteira para o operador **/=**.

Cada ponto OFF deve estar o mais próximo possível do ponto ON correspondente, de acordo com a precisão fornecida.

Cobertura de domínio confiável (Forgács et al., 2024), que requer os seguintes itens de cobertura:

- Um ponto ON, um OFF, um IN e um OUT para cada borda definida por um dos operadores **<**, **≤**, **>** ou **≥**
- Um ponto ON e dois pontos OFF localizados em lados diferentes da fronteira para o operador **=**.
- Um ponto OFF e dois pontos ON localizados em lados diferentes da fronteira para o operador **/=**.

Para os dois critérios de cobertura discutidos, o número de itens de cobertura depende linearmente do número de bordas. Ele pode ser otimizado para ambos os critérios de cobertura usando o mesmo item de cobertura para diferentes bordas. Por exemplo, todas as bordas de uma partição de equivalência podem usar um ponto IN comum ou um par de pontos ON e OFF de uma borda pode ser usado como pontos OFF e ON para a borda da partição de equivalência adjacente. Para domínios com muitas bordas, há um algoritmo de otimização avançado disponível (Forgács et al., 2024).

A cobertura de domínio confiável resulta em uma quantidade um pouco maior de itens de cobertura do que a cobertura de domínio simplificada, mas pode detectar consideravelmente mais defeitos de domínio (Site of Software Test Design, 2020).

O teste de domínio pode ser aplicado em qualquer nível de teste. Ele generaliza o BVA e o EP (consulte ISTQB-CTFL, v4.0.1, Seção 4.2) para domínios mais complexos. Várias abordagens para testes de domínio podem ser encontradas em livros-texto como (Beizer, 1990, Capítulo 6; Binder, 2000, Seção 10.2.4; Kaner; Padmanabhan, et al., 2013; Jorgensen, 2014, cap. 5).

3.1.2 Teste Combinatório

Algumas falhas de software decorrem de combinações específicas de valores de parâmetros, conhecidas como falhas de interação. Os testes combinatórios têm como objetivo revelar essas falhas explorando essas combinações de valores de parâmetros.

Nos testes combinatórios, as condições de teste geralmente combinam parâmetros de configuração ou valores de dados de entrada. Portanto, há duas abordagens principais para o teste combinatório. A primeira abordagem usa combinações de parâmetros de configuração, e cada uma dessas combinações pode ser testada usando o mesmo caso de teste. A segunda abordagem usa combinações de valores de dados de entrada, que se tornam parte de casos de teste completos, criando um conjunto de testes para o sistema em teste (D. R. Kuhn et al., 2013).

Um par específico de parâmetro-valor consiste em um parâmetro e seu valor (p. ex., '(color, red)').

Os critérios de cobertura combinatória incluem os seguintes (Ammann et al., 2008), (Forgács et al., 2019):

- **Cobertura de Escolha Base** assume que alguns pares parâmetro-valor são mais importantes do que outros. Um par de valor base é escolhido para cada parâmetro, e um item de cobertura base é a combinação dos pares parâmetro-valor base. Os itens de cobertura subsequentes são criados a partir do item de cobertura base para cada parâmetro, substituindo seu par de valor base por cada valor não base.
- **Cobertura em Pares**, na qual os itens de cobertura são pares de pares de parâmetro-valor para quaisquer dois parâmetros. Há ferramentas disponíveis para gerar itens de cobertura. Entretanto, geralmente é difícil encontrar um conjunto mínimo de casos de teste que atinja a cobertura por pares.

Para parâmetros com muitos valores, o EP pode ser aplicado primeiro para reduzir esse número e o conjunto de combinações resultantes. A captura dos parâmetros e de seus valores em uma árvore de classificação ou em um modelo de recursos (consulte o IREB-Glossary, 2024) dá suporte a essa atividade. O número final de casos de teste pode ser afetado por restrições entre pares de parâmetro-valor, por combinações adicionadas manualmente que se sabe serem problemáticas ou por causa de combinações inválidas ou inviáveis.

©International Software Testing Qualifications Board

A percepção essencial para o teste combinatório é que nem todos os parâmetros contribuem para uma falha e que a maioria das falhas é desencadeada por um único valor de parâmetro ou por interações entre um número relativamente pequeno de parâmetros (Cohen et al., 1994). Isso se alinha com a hipótese do efeito de acoplamento, indicando que a detecção de defeitos simples em um programa muitas vezes também pode revelar defeitos complexos (Offutt, 1992). Em um estudo limitado (D. Kuhn et al., 2004), os resultados mostraram que cerca de 97% das falhas são causadas por apenas uma ou duas condições de interação, o que indica que o teste em pares é uma técnica de teste eficaz.

Mais informações sobre testes combinatórios, incluindo outros critérios de cobertura como diff-pair-t ou n-wise testing, podem ser encontradas em (Ammann et al., 2008), (Forgács et al., 2019). Ferramentas de suporte a testes combinatórios também estão disponíveis em (Czerwonka, 2004).

3.1.3 Testes Aleatórios

O teste aleatório envolve a seleção aleatória de dados de teste do domínio de entrada do item de teste com base em uma distribuição de probabilidade específica. Para fins de validação, recomenda-se uma distribuição baseada em perfis operacionais. Para fins de verificação, a distribuição deve ser agnóstica em relação ao uso para evitar vieses. Os resultados esperados podem ser adicionados aos casos de teste usando um oráculo de teste, o que, na maioria das vezes, requer um oráculo de teste automatizado.

Os testes aleatórios podem ser guiados ou não guiados. No teste aleatório não guiado, a distribuição de probabilidade permanece fixa durante todo o processo. O teste aleatório guiado, exemplificado por técnicas como o teste aleatório adaptativo (Huang et al., 2019), ajusta a distribuição com base em valores previamente selecionados, evoluindo ao longo do tempo. O teste aleatório guiado visa cobrir o domínio de entrada de forma eficaz, considerando que os defeitos geralmente se agrupam em regiões específicas do domínio.

Os testes aleatórios não possuem critérios de cobertura reconhecidos. Portanto, os critérios de saída só podem se basear no número de testes executados, no tempo de teste ou em medidas semelhantes de conclusão.

O teste aleatório é especialmente valioso quando o conhecimento do domínio é limitado ou quando há necessidade de um grande volume de dados de teste. Ele é econômico e fornece, em termos probabilísticos, percepções sobre a confiabilidade do objeto de teste. Os testes aleatórios ajudam a evitar vieses, como a negligência de defeitos em testes manuais devido à confiança equivocada em algum código ou funcionalidade. No entanto, os testes aleatórios também apresentam vários desafios e limitações, incluindo a negligência da semântica dos dados, a possível falta de defeitos relacionados ao significado dos dados, a negligência de determinados defeitos, a geração de testes redundantes, a dependência de um oráculo de teste automatizado e saídas aleatórias que levam a resultados de teste inconsistentes. Equilibrar as vantagens e as limitações dos testes aleatórios é essencial em cada contexto de teste.

Tradicionalmente, o teste aleatório é considerado menos eficaz do que outras técnicas de teste. Nos últimos anos, essa hipótese foi investigada em muitos estudos empíricos. A conclusão é que, nas circunstâncias mencionadas acima, o teste aleatório pode ser mais eficaz e eficiente do que outras técnicas de teste baseadas em dados (Arcuri et al., 2012), (Wu et al., 2020). O teste aleatório também é aplicado em testes de fuzz e engenharia do caos.

3.2 Técnicas de Teste Baseado no Comportamento

As técnicas de teste baseadas no comportamento derivam casos de teste das especificações do comportamento dinâmico (ou seja, dependente do estado) do item de teste. Este syllabus aborda três técnicas de teste baseado no comportamento:

- Testes CRUD
- Teste de transição de estado
- Teste baseado em cenário

Os testes CRUD e os teste baseado em cenário ampliam a gama de técnicas de teste caixa-preta conhecidas do (ISTQB-CTFL, v4.0.1, Seção 4.2). Este syllabus complementa o teste de transição de estado com critérios de cobertura adicionais.

©International Software Testing Qualifications Board

3.2.1 Teste CRUD

O teste CRUD verifica o ciclo de vida das entidades de dados processadas pelo item de teste. CRUD significa criar, ler, atualizar e excluir. Essas são as quatro operações básicas que as funções podem executar nas entidades.

A matriz CRUD oferece uma visão geral do ciclo de vida das entidades de dados. Suas colunas representam as entidades e suas linhas representam as funções. Suponha que uma função execute uma ou mais operações específicas de criação (Create), leitura (Read), atualização (Update) ou exclusão (Delete) em uma determinada entidade. Isso é mostrado na matriz usando as iniciais dessas operações: C, R, U ou D. Para criar uma matriz CRUD, o TA determina, para cada função, quais das quatro operações ela executa em quais entidades. Deve-se dar atenção especial à operação de leitura, muitas vezes implicitamente vinculada às operações C-U-D.

O teste de CRUD consiste em duas partes (Koomen et al., 2006):

- **Teste de Integridade** é um teste estático que verifica se todas as operações possíveis (ou seja, C, R, U e D) ocorrem com cada entidade (ou seja, se todo o ciclo de vida foi implementado para cada entidade). A ausência de uma operação é uma anomalia que exige investigação.
- **Teste de Consistência** é um teste dinâmico que visa a integrar as várias funções e verificar se a entidade é usada de forma consistente. Ele verifica se as funções interagem corretamente ao lidar com uma entidade. Os casos de teste devem abranger todas as operações na matriz CRUD. Além disso, os testes negativos também devem ser incluídos (p. ex., a leitura de uma entidade que ainda não foi criada). Os casos de teste são projetados por entidade, combinando funções para cobrir todo o seu ciclo de vida.

A **Cobertura CRUD** é medida pelo número de operações executadas pelo conjunto de testes dividido pelo número total de operações na matriz CRUD. Uma cobertura CRUD mais rigorosa pode considerar combinações específicas de operações como itens de cobertura (p. ex., após cada U, todos os R possíveis devem ser cobertos).

O teste CRUD é usado principalmente no nível do sistema. Ele se concentra em defeitos no comportamento do item de teste no tratamento de entidades, como violações de integridade de dados, defeitos de controle de acesso ou inconsistências de dados.

3.2.2 Teste de Transição de Estado

Muitos sistemas complexos têm estado (ou seja, a reação do sistema a um evento depende do seu estado atual). Exemplos de sistemas com estado são os sistemas incorporados, os sistemas baseados em diálogo, os sistemas de controle ou os sistemas que lidam com entidades e seus ciclos de vida.

Um estado simplifica detalhes internos complexos, o que facilita a compreensão do comportamento esperado pelos stakeholders. Durante a análise do teste, o TA deve garantir que o modelo baseado em estado represente o comportamento esperado do item de teste no nível de detalhe necessário para o teste. Se a base do teste já contiver esse modelo, o TA deverá verificar se ele contém as condições do teste e, se necessário, adaptá-lo ou criar um novo modelo.

Nos modelos baseados em estado, os nós representam estados e as bordas representam transições de estado. As transições de estado são acionadas por eventos e podem incluir condições e ações de proteção (consulte ISTQB-CTFL, v4.0.1, Seção 4.2.4). Diversas variantes desses modelos estão em uso, como máquinas de estado finito estendidas (Bochmann et al., 1994), gráficos de estado Harel (Harel, 1987) ou máquinas de estado UML (OMG® UML, 2017).

Além dos critérios de cobertura de transição de estado discutidos em (ISTQB-CTFL, v4.0.1), dois outros critérios são discutidos abaixo, para os quais uma alta eficácia de detecção de defeitos foi comprovada empiricamente:

- **Cobertura de N-switch** se aplica a sequências válidas de N+1 transições consecutivas, também chamadas de N-switches (Chow, 1978). A cobertura de 0-switch é igual à cobertura de transições válidas (consulte ISTQB-CTFL, v4.0.1, Seção 4.2.4). A de 1-switch é um par de transições de entrada e saída em um estado. A extensão de N-switches em sua extremidade por transições de estado subsequentes válidas resulta em N+1 switches. A cobertura de 0 e 1 switch é usada com frequência na prática. Uma cobertura de 100% de 2 switches ou mais só é indicada para um alto risco de falha devido a sequências inesperadas de eventos, pois o número de N switches pode crescer exponencialmente com N.

©International Software Testing Qualifications Board

- **Cobertura de round-trip** (Ammann et al., 2008) aplica-se a caminhos em um modelo baseado em estado que formam loops. Uma viagem de ida e volta é um loop no qual os estados inicial e final são os mesmos, e nenhum outro estado nesse loop ocorre duas vezes. Os itens de cobertura são as viagens de ida e volta. (Antoniol et al., 2002) consideraram esse critério altamente eficaz na detecção de defeitos.

O teste de transição de estado é adequado para automação usando ferramentas de teste baseadas em modelos. Assim como a maioria das técnicas de teste caixa-preta, o teste de transição de estado contribui para a prevenção de defeitos ao detectar defeitos na especificação durante a modelagem. O teste de transição de estado pode ser aplicado em qualquer nível de teste.

Outros critérios de cobertura de transição de estado são discutidos em (Rechtberger et al., 2022). Um modelo recente baseado em estado é o modelo de estado de ação discutido em (Forgács et al., 2024).

3.2.3 Teste Baseado em Cenário

O teste baseado em cenário avalia o comportamento do item de teste em cenários realistas. Técnicas como pesquisa de usuários, histórias de usuários, personas e mapas de jornada do usuário (consulte a Seção 11) podem ajudar a identificar cenários valiosos. Nos teste baseado em cenário, o TA cria um modelo de cenário com base em sequências de ações que constituem fluxos de trabalho por meio do item de teste (cf. ISO/IEC/IEEE 29119-4, 2021). Este syllabus aborda os dois modelos a seguir: diagramas de atividade e casos de uso. Outros modelos incluem fluxogramas, modelo de processo de negócios e diagramas de notação (OMG® BPMN, 2013), diagramas de sequência ou diagramas de colaboração (OMG® UML, 2017). Esses modelos não são discutidos neste syllabus. Consulte (ISTQB- AcT, v1.0) para obter mais detalhes.

Um **diagrama de atividade** é uma representação gráfica do fluxo de trabalho em um sistema. Os diagramas de atividades são particularmente úteis para modelar processos de negócio, mas também podem modelar o fluxo de controle. Os diagramas de atividade ampliam a notação dos fluxogramas, permitindo a modelagem da simultaneidade. Os principais elementos dos diagramas de atividade são nós de início e fim, ações, transições, nós de decisão, nós de mesclagem, nós de bifurcação, nós de junção e swim lanes.

Um caso de uso é uma descrição textual ou gráfica de ações que representam as interações entre um usuário e um sistema ou entre sistemas. O modelo distingue três tipos de cenários:

- **Cenário principal** (o chamado "caminho feliz") - uma sequência típica e esperada de ações que levam à realização de uma meta específica da perspectiva do usuário. Um caso de uso só pode ter um cenário principal.
- **Extensão** (ou cenário alternativo) - uma sequência de ações, diferente do cenário principal, que eventualmente leva à realização da meta do cenário principal.
- **Exceção** - uma sequência de ações que não permite atingir a meta do cenário principal devido a uma ação inesperada (p. ex., uso anormal ou entrada inválida).

Nos teste baseado em cenário, o TA modela casos de teste para cobrir os cenários (ou seja, itens de cobertura), geralmente seguindo uma priorização baseada em riscos. Quando o modelo de cenário não contém loops, cada cenário pode ser testado com um caso de teste separado (ou seja, todos os cenários possíveis no modelo podem ser exercitados com um conjunto de testes). O número de cenários potenciais (caminhos) pode ser infinito quando há loops. Nesse caso, a cobertura de loop simples pode ser aplicada ao modelo de cenário. A cobertura de loop simples requer testes quando cada loop é executado zero vezes (ou seja, pulado), com exatamente uma iteração, com mais de uma iteração (ou seja, um número típico de iterações) e com o número máximo de iterações (se possível).

A **cobertura baseada em cenário** é medida pelo número de cenários executados dividido pelo número de todos os cenários identificados. Os cenários podem ser identificados pela aplicação de vários critérios de cobertura ao modelo de cenário (consulte Koomen et al., 2006). Os critérios de cobertura podem exigir o teste de cada cenário mais de uma vez. Por exemplo, um cenário pode exigir uma cobertura adicional de EP ou BVA das variáveis que ocorrem no cenário. Nesses casos, um cenário pode precisar ser testado por mais de um caso de teste.

O teste baseado em cenários é frequentemente realizado em testes de sistema ou testes de aceite. Ele assume a forma de teste de ponta a ponta com foco na adequação funcional de um sistema (consulte a Seção 4.1) da perspectiva do usuário. No entanto, o teste baseado em cenários também pode ser usado em outros níveis de teste (p. ex., teste de integração de componentes com cenários baseados nos protocolos de interação ou teste de componentes de classes com estado e orientadas a objetos com cenários que invocam vários métodos) e em

©International Software Testing Qualifications Board

testes não funcionais (p. ex., os cenários podem constituir os elementos dos perfis operacionais usados em testes de confiabilidade, flexibilidade ou compatibilidade).

3.3 Técnicas de Teste Baseado em Regras

As técnicas de teste baseadas em regras verificam a implementação do comportamento sem estado de um item de teste, especificado por regras que são válidas independentemente de seu estado (p. ex., regras de negócios). Neste syllabus, são discutidas duas técnicas de teste baseadas em regras, a saber:

- teste de tabela de decisão.
- testes metamórficos.

O syllabus do nível fundamental (ISTQB-CTFL, v4.0.1) abrange os fundamentos do teste de tabela de decisão. Este syllabus aborda tópicos mais avançados. A terminologia e a notação usadas seguem o padrão (OMG® DMN, 2024).

3.3.1 Teste de Tabela de Decisão

No teste de tabela de decisão, o TA geralmente começa criando uma tabela de decisão completa ou analisando uma tabela existente obtida da base de teste. O número de regras de uma tabela de decisão completa é o produto dos números de valores de suas condições. Esse número cresce exponencialmente com o número de condições e seus valores e motiva a minimização das tabelas de decisão.

As tabelas de decisão podem ser minimizadas com a fusão de regras menos importantes usando o operador "-". Recomenda-se desconsiderar regras inviáveis (ou seja, regras com uma combinação de valores de condição que nunca podem ocorrer) ao mesclar regras. Geralmente, as regras inviáveis são removidas da tabela de decisão antes da mesclagem. Um possível algoritmo de minimização sistemática procura regras equivalentes a ações que diferem apenas em uma condição e abrangem todos os seus valores possíveis. Essas regras são mescladas e o valor da condição diferente é substituído por "-". O resultado da minimização sistemática pode depender da ordem em que as colunas são minimizadas, nem sempre levando a uma solução ideal. O TA precisa verificar se é possível fazer mais alguma minimização.

A tarefa do TA é revisar a tabela de decisões, possivelmente com outros stakeholders, com os seguintes critérios:

- consistência (ou seja, se duas regras diferentes se aplicam à mesma combinação de valores de condição, então elas são equivalentes em termos de ação)
- viabilidade (ou seja, não contém regras inviáveis)
- completude (ou seja, nenhuma combinação viável de valores de condição está faltando)
- correção (ou seja, as regras modelam o comportamento pretendido do sistema)

Além disso, é aconselhável que as regras não se sobreponham (ou seja, para qualquer combinação de valores de condição, no máximo uma regra se aplica). A sobreposição de regras pode ocorrer quando a tabela de decisão original já estiver minimizada ou se as regras forem mescladas incorretamente.

O *procedimento de checksum* usa o número de regras em uma tabela de decisão minimizada para indicar sobreposições e lacunas. Para cada regra na tabela minimizada, é calculado o número de regras que ela representa na tabela de decisão original. Uma regra sem um '**-**' nas condições (ou seja, com valores individuais para todas as condições) tem pontuação 1. Cada valor '**-**' multiplica a pontuação da regra pelo número de valores individuais para a respectiva condição. A soma das pontuações das regras é a soma de verificação da tabela de decisão minimizada. Se a soma de verificação for menor que a soma de verificação da tabela de decisão original, a tabela de decisão minimizada estará incompleta. Se a soma de verificação for maior, algumas regras se sobrepõem ou existem regras adicionais (p. ex., combinações inviáveis de condições). Se a minimização foi realizada corretamente, os *checksums* são iguais. Os *checksums* iguais, por si só, não garantem que a tabela de decisão minimizada seja equivalente à original.

A **cobertura da tabela de decisão** é medida dividindo-se o número de colunas testadas pelo número total de colunas viáveis na tabela de decisão. Ao implementar casos de teste resultantes de uma regra da tabela de decisão, o TA deve implementar as condições e ações. O TA precisa decidir como implementar os valores de condição '**-**' para uma determinada regra porque '**-**' representa pelo menos dois valores. No entanto, se o nível de

©International Software Testing Qualifications Board

risco associado à tabela de decisão for alto, o TA deve abster-se de minimizar e deve medir a cobertura da tabela de decisão das colunas viáveis na tabela de decisão completa.

3.3.2 Teste de Metamórfico

O teste metamórfico (MT) é uma técnica que visa a gerar casos de teste baseados em um caso de teste de origem existente. Um ou mais casos de teste de acompanhamento são gerados pela alteração (metamorfização) do caso de teste de origem com base em uma relação metamórfica (MR). A MR define uma propriedade do item de teste e descreve como uma alteração nas entradas de um caso de teste é refletida nos resultados esperados do caso de teste.

O TA combina os casos de teste de origem e de acompanhamento de um MR a um procedimento de teste, com uma avaliação conjunta de resultados. Se eles atenderem à relação metamórfica, o teste será aprovado, caso contrário, será reprovado. Em caso de falha, a depuração subsequente deve determinar qual dos casos de teste individuais envolvidos falhou.

Por exemplo, considere uma função que determina a média de uma série de números. Um caso de teste de origem é criado com uma série de números e uma média esperada, e o caso de teste é executado para confirmar que foi aprovado. Um MR poderia afirmar que qualquer permutação da série de números resulta na mesma média.

Usando esse MR, o TA pode criar vários casos de teste de acompanhamento, cada um com o mesmo conjunto de números na entrada, mas em uma ordem diferente. O resultado esperado permanece o mesmo.

Para a mesma função média, o TA também pode usar outro MR, que afirma que, se cada número da série for multiplicado pelo mesmo número, **x**, o resultado esperado também será multiplicado por **x**. Com esse MR, o TA pode criar qualquer número de casos de teste de acompanhamento a partir de um caso de teste de origem, escolhendo valores diferentes de **x**. Isso pode ser particularmente útil quando o projeto e a execução do teste são automatizados. O TA também pode combinar dois ou mais MR para criar casos de teste de acompanhamento (p. ex., permutar e multiplicar por 2).

O MT também pode ser usado na presença de um problema de oráculo de teste (consulte a Seção 1.3.4). Em tal situação, os resultados esperados do caso de teste de origem e dos casos de teste de acompanhamento não estão disponíveis, portanto, seus resultados de teste não podem ser avaliados individualmente. Um exemplo pode ser um programa de atuária[1] baseado em IA que prevê a idade de morte com base em um grande conjunto de dados. O MR pode afirmar que, se o número de cigarros fumados for aumentado, a idade prevista para a morte deverá diminuir.

Atualmente, não há medidas de cobertura reconhecidas para MT que forneçam critérios de saída úteis. Cobrir cada MR uma vez é insuficiente porque só é possível obter uma verificação parcial dos resultados esperados. O TA pode combinar o MT com testes aleatórios para gerar muitos casos de teste de origem de baixo nível e casos de teste de acompanhamento para o mesmo MR. Por exemplo, no cálculo da média mencionado acima, um gerador de números aleatórios pode ser usado para gerar várias entradas para o caso de teste de origem, bem como permutações e multiplicadores aleatórios para os casos de teste de acompanhamento.

O MT pode ser usado para a maioria dos itens de teste e aplicado a testes funcionais e não funcionais (p. ex., teste de carga, geração de carga por casos de teste de acompanhamento usando MR ou teste de instalabilidade com vários parâmetros de instalação que podem ser selecionados em várias sequências). É uma técnica de teste preferida para sistemas baseados em IA (ISTQB-AI, v1.0).

Para obter mais detalhes, consulte também o padrão (ISO/IEC/IEEE 29119-4, 2021) ou os artigos de pesquisa (Segura; Towey, et al., 2020) e (Segura; Fraser, et al., 2016).

3.4 Teste Baseado na Experiência

Um TA emprega abordagens de teste baseado na experiência e técnicas de teste, aproveitando sua experiência e encontros anteriores para orientar os testes. Este capítulo detalha o uso de documentação pelo TA em testes

[1] *Nota do BSTQB*: A atuária é a ciência que aplica métodos matemáticos e estatísticos para analisar e gerenciar riscos e expectativas de diversas naturezas, como econômicas, financeiras etc.

©International Software Testing Qualifications Board

baseados em sessões e testes baseados em listas de verificação (consulte ISTQB-CTFL, v4.0.1, Seções 4.4.2 e 4.4.3). Além disso, é discutido o teste de multidão. Todas as técnicas de teste baseado na experiência podem ser aplicadas em teste baseado na colaboração, como o desenvolvimento orientado por testes de aceite. Para obter mais detalhes, consulte (ISTQB-CTFL, v4.0.1, Seção 4.5).

3.4.1 Cartas de Teste que apoiam o Teste Baseado em Sessão

Nos testes exploratórios, uma carta de teste fornece a missão de uma sessão de teste, que descreve seu escopo e objetivos e informações como limitações, cronogramas e riscos. Ele serve como um roteiro para os testes, fornecendo estrutura para as sessões de teste. As cartas de teste ajudam o TA a manter o foco em áreas ou recursos específicos a serem testados e, ao mesmo tempo, permitem a flexibilidade para explorar o sistema quando necessário. A carta de teste não especifica os conjuntos de testes que serão executados em cada sessão de teste.

Ao preparar uma carta de teste para testes baseados em sessões, o TA deve considerar determinados fatores que influenciam a modelagem da carta de teste, em particular:

- fatores de clientes e requisitos (p. ex., requisitos obtidos de clientes, casos de uso de negócio do sistema, requisitos de qualidade) e mapas de jornada do usuário (ou seja, interação do usuário com o sistema ao longo do tempo)
- fatores do produto (p. ex., fluxos funcionais, principais objetivos do produto, recursos do produto, desenho do software e interfaces)
- fatores de gerenciamento de projetos (p. ex., restrições de tempo, objetivo do projeto, esforço estimado e valor de negócio)

Uma carta de teste consiste em uma missão que descreve o objetivo do teste e várias informações adicionais. Um formato leve e popular de uma missão é "Explore [**alvo**] com [**recursos**] para descobrir [**informações**]" Hendrickson, 2013), em que [**alvo**] descreve o que deve ser explorado (p. ex., área, recurso, risco, componente, e requisito), [**recursos**] descrevem o que o TA usará (p. ex., dados de teste, configurações, ferramentas, restrições, heurística e dependências) e [**informações**] explicam que tipo de informação o TA pretende encontrar (p. ex., avaliações de características de qualidade, tipo esperado de defeitos e violações de padrões).

As cartas de teste podem conter, mas não estão limitadas às seguintes informações adicionais (Ghazi et al., 2017):

- informações organizacionais (p. ex., duração da sessão de teste, data e hora de início e nome do testador)
- objetivos do teste (p. ex., motivação do teste e missão da carta de teste)
- escopo do teste (p. ex., áreas específicas de interesse dentro do sistema em teste, nível de teste, técnicas de teste a serem usadas, ideias de teste, critérios de saída, prioridades, o que a carta de teste deve cobrir e uma descrição do que não será testado)
- critérios de entrada (ou seja, condições prévias que devem ser atendidas para que seja possível iniciar a sessão de teste)
- informações relacionadas ao produto (p. ex., definição, dados e fluxo de trabalho entre componentes e arquitetura do sistema)
- limitações (ou seja, o que o produto nunca deve fazer)
- descrição do ambiente de teste
- fontes de dados existentes, informações sobre produtos e ferramentas de teste que ajudariam nos testes
- informações históricas (p. ex., defeitos encontrados anteriormente, como defeitos de compatibilidade e interoperabilidade, questões abertas atuais que se referem a anomalias existentes e padrões de falha relacionados a testes do passado)
- restrições e riscos (p. ex., regulamentos, regras e padrões usados)

©International Software Testing Qualifications Board

Durante uma sessão de teste, o TA grava os registros de teste que incluem perguntas, observações ou ideias para testes futuros, juntamente com os resultados do teste. Esses dados são documentados em uma planilha de sessão.

O escopo e os detalhes das informações incluídas em cartas de teste específicas podem variar e afetar o grau de flexibilidade do TA. Por exemplo, a definição apenas dos objetivos gerais do teste oferece amplo espaço para exploração, enquanto a inclusão de informações sobre as técnicas de teste a serem usadas pode restringir o TA. Por outro lado, acrescentar informações (p. ex., o que o sistema definitivamente não pode fazer) pode ajudar a reduzir a probabilidade de relatar resultados falso-positivos.

3.4.2 Listas de Verificação que apoiam Técnicas de Teste Baseado na Experiência

O teste baseado em listas de verificação é uma técnica de teste amplamente utilizada devido à sua adaptabilidade, simplicidade e eficácia para garantir a qualidade do software. Com o uso de listas de verificação, o TA garante a cobertura de todos os aspectos essenciais conhecidos de um item de teste, o que evita a negligência de áreas críticas. Elas também introduzem consistência entre os ciclos de teste e entre vários TA. Ao usar uma lista de verificação, o TA se concentra nos aspectos importantes. As listas de verificação são uma forma de registrar as experiências anteriores do TA com falhas e defeitos. Elas servem como um lembrete ou uma fonte de inspiração se o TA ficar sem ideias (p. ex., durante o teste exploratório). O TA economiza tempo ao reutilizar uma lista de verificação padrão ou uma lista de verificação criada por um de seus colegas, mas somente se essas listas de verificação forem relevantes para o item de teste. As listas de verificação ajudam a reduzir o trabalho necessário para documentar os casos de teste, o que pode ser um grande recurso quando os requisitos e o software estão em constante mudança.

A preparação das listas de verificação geralmente é de responsabilidade do assistente técnico. As listas de verificação dão suporte aos testes baseados em experiência, pois ajudam a organizar, estruturar e orientar os testes. A criação de uma lista de verificação reutilizável, passível de manutenção, clara e eficiente exige esforço.

As etapas a seguir podem ser uma diretriz para a configuração de uma lista de verificação adequada para teste baseado na experiência:

Primeiro, o TA determina o escopo, os objetivos e o formato da lista de verificação, pois eles influenciam a profundidade do teste e o nível de detalhe necessário. Uma lista de verificação para *ler-fazer* contém os principais elementos a serem considerados em um determinado processo (p. ex., os itens da lista de verificação contêm entradas inválidas concretas para um campo de entrada a ser verificado). Uma lista de verificação do tipo *fazer-confirmar* serve como auxílio para orientar o processo de pensamento, fornecendo ideias de teste baseado na experiência para explorar um aplicativo mais a fundo (p. ex., um item da lista de verificação pode exigir que se verifique se os resultados da pesquisa são relevantes).

Em seguida, o TA coleta as informações necessárias para definir os itens da lista de verificação. Isso pode incluir a coleta de percepções de profissionais experientes, a pesquisa em bibliotecas de defeitos e taxonomias de defeitos (Beizer, 1990), (Kaner; Falk, et al., 1999), a revisão da documentação relevante e a análise de riscos, casos de teste e cenários potenciais. Os itens da lista de verificação devem ser claros, específicos, não ambíguos, consistentes, relevantes, passíveis de manutenção, acionáveis e mensuráveis. Eles devem ser formulados como perguntas que possam ser respondidas com "sim", "não" ou "não aplicável". Elas exigem prioridades atribuídas com base em sua importância, impacto potencial e nível de risco. As listas de verificação não são guias abrangentes de como fazer. Em vez disso, são ferramentas rápidas e simples para profissionais especializados.

Por fim, o TA estrutura e organiza a lista de verificação categorizando os itens em grupos lógicos com base em áreas funcionais, funções de usuário, níveis de teste ou outros critérios relevantes. A criação de categorias separadas pode ser especialmente útil para uma lista de verificação longa.

Sempre que possível, devem ser usados modelos e padrões. O TA pode economizar esforços utilizando modelos existentes ou listas de verificação predefinidas alinhadas com os padrões e as práticas recomendadas do setor.

Uma lista de verificação nunca é finalizada. O TA a revisa e refina continuamente, adaptando-a para refletir novas descobertas, prioridades alteradas, feedback de outros testadores ou lições aprendidas em ciclos de teste anteriores. Ao compartilhar a lista de verificação com outros testadores, o TA promove a consistência e a colaboração e os ajuda a entender melhor os itens de teste e as áreas críticas nas quais devem se concentrar durante o teste.

©International Software Testing Qualifications Board

3.4.3 Teste de Multidão

O teste de multidão distribui os testes entre um grupo de testadores internos ou externos de diversas origens e locais. Pode ser uma maneira econômica de validar a usabilidade e abrange características de qualidade funcionais e não funcionais (Alyahya, 2020), (Leicht et al., 2017).

Alguns dos benefícios do teste de multidão inclui:

- **Diversos ambientes de teste**. Os testadores podem estar em vários locais geográficos e usar várias configurações de ambiente com uma ampla variedade de dispositivos, navegadores e condições de rede.
- **Maior flexibilidade**. Facilmente dimensionável para lidar com muitos testes em um curto espaço de tempo.
- **Custo-benefício**. Normalmente, os testes em grupo são mais baratos do que manter uma equipe de testes interna grande e diversificada ou complementar com serviços de testes externos.
- **Feedback rápido**. Os testadores podem fornecer feedback rápido, ajudando a identificar e corrigir falhas com antecedência.
- **Perspectiva real do usuário**. Os testadores podem ser usuários reais do aplicativo e podem fornecer perspectivas melhores sobre a experiência e a usabilidade do usuário. Isso pode ser especialmente valioso nos testes de aceite do usuário.
- **Variabilidade**. Como os testes são sempre executados por uma grande variedade de testadores, eles não são tão repetíveis. Embora isso possa ser uma limitação, também resulta em uma cobertura mais ampla, o que aumenta as chances de encontrar defeitos.

Algumas das limitações do teste de multidão inclui:

- **Qualidade não confiável dos testes**. A qualidade dos testes pode variar significativamente, dependendo das habilidades de cada testador, embora isso possa não ser relevante, por exemplo, quando o objetivo é o feedback sobre a experiência do usuário.
- **Desafios de comunicação**. A coordenação com muitos testadores de vários locais com fusos horários diferentes, diferenças culturais e barreiras linguísticas pode ser um desafio.
- **Segurança**. O compartilhamento de software com testadores externos apresenta riscos à segurança e à confidencialidade dos dados. Medidas adequadas podem atenuar esses riscos, permitindo o uso responsável do teste de multidão sem revelar detalhes confidenciais ou facilitar o plágio.
- **Documentação e relatórios**. Garantir a documentação abrangente dos testes e gerenciar muitos descobertas, inclusive duplicatas e falsos positivos, pode ser um desafio quando se lida com um grupo grande e diversificado de testadores.

O teste de multidão é uma abordagem que não substitui os TA que aplicam técnicas de teste, mas pode aumentar a cobertura de diversos ambientes de teste.

3.5 Aplicação das Técnicas de Teste mais apropriadas

Os testes devem ser tão eficazes e eficientes quanto possível em um determinado contexto. Para isso, o TA ajuda o gerente de testes a selecionar a(s) técnica(s) de teste mais adequada(s). Além disso, o TA pode usar automação para apoiar as atividades de teste. Isso inclui a automação do projeto do teste, descrita nesta seção, e a automação da execução do teste de suporte, descrita na Seção 1.3.6.

3.5.1 Seleção de Técnicas de Teste para atenuar os Riscos do Produto

A seleção da(s) técnica(s) de teste mais adequada(s) é fundamental para a mitigação eficaz e eficiente do risco do produto e é influenciada por muitos fatores, incluindo:

Objetivos do teste (consulte ISTQB-CTFL, v4.0.1, Seção 1.1.1) especificam quais aspectos do objeto de teste devem ser avaliados. Eles orientam a escolha das técnicas de teste e dependem do tipo de sistema (p. ex., teste

©International Software Testing Qualifications Board

baseado em domínio para cálculos numéricos em engenharia versus teste de tabela de decisão em gerenciamento de risco de crédito).

Riscos do produto estão associados a possíveis defeitos. A melhor maneira de detectar esses defeitos é usar técnicas de teste específicas, já que a maioria das técnicas de teste se concentra na detecção de tipos específicos de defeitos (consulte as Seções 3.1, 3.2, 3.3 e 3.4 deste syllabus). Por exemplo:

- As técnicas de teste baseado em dados podem detectar defeitos no manuseio de dados, na implementação do domínio, nas interfaces de usuário, nos cálculos e nas combinações de parâmetros.
- As técnicas de teste baseado no comportamento podem detectar defeitos nos requisitos do usuário, como recursos ausentes, defeitos de comunicação e defeitos de processamento.
- As técnicas de teste baseado em regras podem detectar defeitos na lógica e nos fluxos de controle.

A análise de risco também ajuda a determinar a abordagem de teste adequada, por exemplo:

- Critérios de saída baseado na cobertura. Quanto mais alto for o nível de risco, mais rigorosa poderá ser a cobertura (p. ex., todas as combinações na cobertura combinatória em vez da cobertura por pares). No entanto, o TA deve sempre levar em conta o equilíbrio entre a força da cobertura e o esforço de teste necessário.
- O teste baseado na experiência pode ser usado quando é difícil definir a cobertura, quando o nível de risco é baixo ou quando o cronograma do projeto é apertado.

Base do teste. Se a especificação do objeto de teste estiver usando modelos, o TA poderá usar técnicas de teste com base nesses modelos. Se for impossível ou difícil derivar um oráculo de teste a partir da base de teste, poderão ser aplicadas técnicas de teste como teste metamórfico ou técnicas de teste baseado na experiência.

Conhecimento dos tipos de defeitos recorrentes pode indicar a seleção das técnicas de teste que se concentram na detecção desses defeitos (p. ex., teste baseado em listas de verificação). Se a expectativa for descobrir defeitos semelhantes aos de iterações ou projetos anteriores, pode ser razoável usar as mesmas técnicas de teste bem-sucedidas.

Conhecimento e experiência do testador. Se o TA não estiver familiarizado com uma determinada técnica de teste, não é recomendável usá-la em tarefas de teste críticas. O conhecimento do domínio também pode afetar a seleção das técnicas de teste. Por exemplo, pouco ou nenhum conhecimento do domínio indica que as técnicas de teste, como o teste exploratório, podem ser ineficazes.

Ciclo de vida de desenvolvimento de software utilizado. Um modelo de desenvolvimento sequencial é propício para o emprego de técnicas mais formais. Por outro lado, um modelo de desenvolvimento iterativo pode ser mais apropriado para a adoção de técnicas de teste leves (p. ex., técnica de teste baseado na experiência) ou quando o projeto do teste pode ser automatizado.

Requisitos do cliente e do contrato. Os contratos podem exigir explicitamente a realização de testes específicos (p. ex., em termos de determinados níveis ou tipos de teste), o que influencia a seleção de técnicas de teste (p. ex., critérios de aceite com um conjunto de cenários fornecidos pelo cliente sugerem o uso de uma técnica de teste baseado em cenário).

Requisitos regulatórios. Quando um projeto segue um padrão, ele pode exigir o uso de técnicas de teste específicas. Por exemplo, a norma (ISO 26262, 2018) exige o uso de técnicas de teste, como particionamento de equivalência, análise de valor limite ou suposição de erro, dependendo do ASIL (Automotive Safety Integrity Level) atribuído a um objeto de teste.

Restrições do projeto, como tempo e orçamento, podem afetar o uso de técnicas demoradas ou que exijam recursos caros.

As técnicas de teste são frequentemente combinadas para aumentar a eficiência e a eficácia da detecção de defeitos. Por exemplo:

- O BVA pode ser usado para condições de proteção em testes de transição de estado.
- O teste de domínio pode ser usado em teste baseado em cenário para determinar o valor de uma condição de uma tabela de decisão ou de uma variável que ocorre em um sistema em teste.

©International Software Testing Qualifications Board

- O teste baseado em cenário pode ser combinado com a cobertura de decisão, uma técnica de teste caixa-branca discutida em (ISTQB-TTA, v4.0), para cobrir rigorosamente as decisões no processo de negócios. Para obter um exemplo, consulte o teste de ciclo de processo em (Koomen et al., 2006).
- O teste baseado em cenário pode ser combinado com a cobertura de *round-trip* para abordar os riscos específicos das atividades cíclicas dos processos de negócios.

3.5.2 Benefícios e riscos da Automação da Modelagem de Teste

O TA pode usar ferramentas para aplicar técnicas de teste, especialmente técnicas de teste caixa-preta. Ao automatizar a modelagem de teste, o TA cria um modelo de teste e gera automaticamente o material de teste a partir desse modelo. Por exemplo, o TA projeta um modelo de transição de estado e permite que uma ferramenta de teste baseado em modelo gere casos de teste para cobertura de *round-trip*.

A automação da modelagem de teste geralmente melhora a eficiência e a eficácia do teste. Seus benefícios incluem:

- **Prevenção de defeitos**. A modelagem para testes é uma forma eficaz de avaliar a qualidade da base de testes (consulte a Seção 5.2.1).
- **Capacidade ampliada**. A automação permite a aplicação de técnicas de teste mais complexas e critérios de cobertura, como teste combinatório, teste aleatório ou cobertura N-switch, reduzindo o risco de código não testado.
- **Compreensibilidade aprimorada**. Os critérios de seleção de teste especificados em uma ferramenta referem-se de forma mais clara e visível às condições de teste e justificam a cobertura gerada de forma mais compreensível.
- **Menos trabalho repetitivo**. O testware pode ser gerado a partir do modelo de teste, reduzindo o trabalho manual e repetitivo, como a especificação de testes.
- **Menos esforços de manutenção**. O modelo de teste é a única fonte de verdade usada para derivar o testware. Como resultado, somente o modelo de teste precisa ser mantido.
- **Menos material de teste com defeito**. O trabalho manual é propenso a erros. O material de teste gerado por ferramentas tem maior qualidade e consistência.
- **Colaboração aprimorada da equipe**. Os stakeholders podem revisar o modelo de teste para encontrar defeitos ou para entender melhor as condições do teste.
- **Rastreabilidade aprimorada**. É mais fácil vincular os elementos de um modelo de teste às condições de teste do que os próprios casos de teste. Se a ferramenta oferecer suporte, os casos de teste gerados herdarão esses links, melhorando a rastreabilidade geral dos testes.
- **Vários formatos de saída**. O material de teste pode ser gerado em vários formatos de saída, conforme necessário para outras ferramentas e atividades subsequentes.

O TA também deve considerar os riscos de automatizar a modelagem de teste. Eles incluem ignorar as condições de teste que não são mostradas no modelo, subestimar o esforço de manutenção do modelo de teste, os stakeholders acharem o modelo difícil de entender e os riscos genéricos da automação de teste (consulte ISTQB-CTFL, v4.0.1, Seção 6.2).

©International Software Testing Qualifications Board

4 Teste de Características de Qualidade - 60 min

Palavras-chave

adaptabilidade, compatibilidade, flexibilidade, adequação funcional, integridade funcional, correção funcional, adequação funcional, teste funcional, instalabilidade, capacidade de interação, interoperabilidade, usabilidade, experiência do usuário

Objetivos de aprendizagem para o Capítulo 4:

4.1 Teste funcional

TA-4.1.1 (K2) Diferenciar entre testes de correção funcional, adequação funcional e integridade funcional

4.2 Teste de usabilidade

TA-4.2.1 (K2) Explicar como o Analista de Teste contribui para os testes de usabilidade

4.3 Teste de flexibilidade

TA-4.3.1 (K2) Explicar como o Analista de Teste contribui para os testes de adaptabilidade e instalabilidade

4.4 Teste de compatibilidade

TA-4.4.1 (K2) Explicar como o Analista de Teste contribui para os testes de interoperabilidade

©International Software Testing Qualifications Board

Introdução

Este syllabus usa o modelo de qualidade de software fornecido na ISO 25010 (ISO/IEC 25010, 2023) como um guia e discute as características de qualidade em foco para um TA. Entretanto, a terminologia de usabilidade difere desse padrão para ser consistente com o Usability Testing Syllabus (ISTQB-UT, v1.0).

O Apêndice F fornece uma visão geral do modelo de qualidade da norma ISO 25010 atual, as alterações em comparação com a versão anterior e os syllabi relacionados do ISTQB® que se concentram em testar características de qualidade específicas.

4.1 Teste Funcional

O teste funcional é uma das principais tarefas do TA. Enquanto o (ISTQB-CTFL, v4.0.1) resume brevemente o teste funcional como um tipo de teste, este syllabus entra em mais detalhes, discutindo as subcaracterísticas da adequação funcional.

4.1.1 Subcaracterísticas da Adequação Funcional

O modelo de qualidade do produto da ISO 25010 (ISO/IEC 25010, 2023) diferencia três subcaracterísticas da adequação funcional. Embora todas elas possam ser avaliadas por meio de testes funcionais, nem toda atividade de teste funcional as aborda igualmente bem. O TA deve ser capaz de selecionar os níveis de teste e as técnicas de teste apropriados para abordar os riscos do produto relacionados a uma subcaracterísticas específica da adequação funcional.

O **Teste de Integridade Funcional** deve abranger todas as tarefas especificadas do software e os objetivos dos usuários pretendidos. A principal questão é se tudo o que foi solicitado foi implementado.

A integridade funcional deve ser abordada o mais cedo possível, revisando a especificação de requisitos em modelos de desenvolvimento sequencial e discutindo histórias de usuários, incluindo critérios de aceite, durante a redação colaborativa de histórias de usuários no desenvolvimento ágil de software. No teste do sistema, no teste de integração do sistema e no teste de aceite, a integridade funcional pode ser testada dinamicamente.

As técnicas de teste baseado no comportamento, como o teste baseado em cenário, é uma boa opção, embora outras técnicas de teste caixa-preta também sejam adequadas. A rastreabilidade entre a base de teste, as condições de teste e os casos de teste é essencial para determinar o nível alcançado de integridade funcional.

O **Teste de Correção Funcional** responde à questão de saber se os resultados reais estão corretos (p. ex., exatos, precisos e consistentes) para entradas válidas e inválidas. É fundamental encontrar um oráculo de teste eficaz que forneça os resultados esperados em detalhes.

A correção funcional pode ser testada em qualquer nível de teste. Em termos de deslocamento para a esquerda, a maioria dos testes de correção funcional deve ocorrer em testes de componentes e testes de integração de componentes. Mesmo que o TA não seja responsável por esses níveis de teste, ele deve contribuir para que eles atinjam os objetivos do teste da melhor forma possível.

Todas as técnicas de teste caixa-preta, técnicas de teste baseado na experiência e teste baseado em colaboração são adequadas.

O **Teste de Adequação Funcional** verifica se as funções facilitam a realização das tarefas e dos objetivos especificados. O foco é verificar se tudo o que foi implementado atende às necessidades dos usuários.

O teste de adequação funcional pode incluir revisões do desenho da interface do usuário, especialmente para aplicativos interativos. O teste dinâmico começa com o teste do sistema e o teste de aceite no desenvolvimento sequencial bem como sessões de demonstração sobre desenvolvimento ágil de software.

O teste exploratório e o teste baseado na colaboração são os mais adequados. Além disso, as técnicas de teste caixa-preta baseadas no comportamento também são adequadas.

©International Software Testing Qualifications Board

4.2 Teste de Usabilidade

A usabilidade refere-se a um conceito amplo de características de qualidade relacionadas ao usuário, abrangendo a capacidade de interação do modelo de qualidade do produto (ISO/IEC 25010, 2023) e a utilidade do modelo de qualidade em uso da ISO 25019 (ISO/IEC 25019, 2023). Mais informações sobre testes de usabilidade podem ser encontradas em (ISTQB-UT, v1.0), (ISO 9241-210, 2019) e (UXQB-FL, v4.01).

4.2.1 Contribuição do Analista de Teste para o Teste de Usabilidade

Os testes de usabilidade geralmente se concentram na avaliação dos seguintes aspectos:

- Capacidade de interação - permitir que os usuários concluam tarefas em contextos específicos de uso (ISO/IEC 25010, 2023) de forma eficaz, eficiente e satisfatória.
- Experiência do usuário - abordando as percepções dos usuários antes, durante e depois da interação com o objeto de teste.
- Acessibilidade - garantir que os usuários com deficiências, origens culturais diversas ou barreiras linguísticas possam usar o sistema de forma eficaz e eficiente.

O TA pode colaborar com os testes de usabilidade desde a fase inicial, usando seu conhecimento sobre os grupos de usuários-alvo, suas metas, seu contexto de uso, possíveis dificuldades no uso do sistema ou experiências negativas do usuário.

O TA pode contribuir para as principais técnicas de avaliação de usabilidade da seguinte forma:

- As **Revisões de Usabilidade** são realizadas por especialistas em usabilidade para identificar possíveis problemas de usabilidade e desvios dos critérios estabelecidos. As revisões de usabilidade podem variar de revisões informais a inspeções. O TA pode adaptar os critérios de revisão às necessidades específicas dos grupos de usuários, aos objetivos e prioridades de negócio específicos e ao contexto de uso (p. ex., adaptando uma lista de verificação genérica de usabilidade).
- As **Sessões de Teste de Usabilidade** envolvem futuros usuários ou seus representantes tentando resolver tarefas predefinidas para avaliar se as tarefas podem ser concluídas de forma eficaz, eficiente e satisfatória. O TA pode contribuir para a criação de cenários para as sessões de teste de usabilidade de acordo com personas, grupos de usuários ou perfis operacionais.
- Os **Questionários** ou **Pesquisas com Usuários**, incluindo classificação e feedback, medem a satisfação do usuário. Exemplos de questionários de usuários são o SUMI (Software Usability Measurement Inventory SUMI, 1991) e o WAMMI (Website Analysis and Measurement Inventory WAMMI, 1999). O TA pode ajudar a elaborar um questionário e avaliar as respostas para atender às metas específicas dos usuários-alvo dentro do contexto de uso.

Nos testes de acessibilidade, um objetivo comum de teste é verificar a conformidade com os padrões. O padrão internacional Web Content Accessibility Guidelines (WCAG, 2023) define três níveis de conformidade (A, AA e AAA) que representam graus crescentes de acessibilidade de conteúdo da Web. Os padrões nacionais incluem a Lei de Igualdade do Reino Unido (UK Government, 2010) e a Lei dos Americanos Portadores de Deficiência dos Estados Unidos (U.S. Department of Justice, 2010), e a Lei Brasileira de Inclusão da Pessoa com Deficiência, (LBI, 2015). A TA pode identificar o nível de conformidade exigido e as necessidades específicas do grupo-alvo pretendido analisando o contexto de uso.

4.3 Teste de Flexibilidade

Os testes de flexibilidade (também conhecidos como testes de portabilidade) verificam se o objeto de teste pode ser adaptado às mudanças em seus contextos de uso ou no ambiente do sistema. O modelo de qualidade de produto ISO 25010 (ISO/IEC 25010, 2023) diferencia as seguintes subcaracterísticas de flexibilidade:

- adaptabilidade
- escalabilidade
- capacidade de instalação

©International Software Testing Qualifications Board

- possibilidade de substituição

A flexibilidade tem aspectos técnicos e não técnicos. Esta seção se concentra em como a TA pode contribuir para os testes de adaptabilidade e instalabilidade. A escalabilidade e a substituibilidade estão relacionadas a aspectos técnicos e são discutidas em (ISTQB-PT, v1.0) e (ISTQB-TTA, v4.0), respectivamente.

4.3.1 Contribuição do Analista de Teste para o Teste de Adaptabilidade e o Teste de Instalabilidade

O **Teste de Adaptabilidade** verifica se o objeto de teste pode ser adaptado ou transferido para o hardware, software ou outros ambientes operacionais ou de uso pretendidos.

O TA dá suporte ao teste de adaptabilidade identificando os ambientes-alvo pretendidos (p. ex., versões de sistemas operacionais móveis suportados e versões de navegadores que podem ser usados) e projetando testes que abrangem combinações desses ambientes. Como isso exige dados de teste que representam várias configurações de parâmetros de ambiente, técnicas de teste como testes combinatórios são frequentemente aplicadas (consulte a Seção 3.1.2). Outro exemplo é a integração de vários componentes de plataforma em projetos de clientes para garantir a compatibilidade entre os ambientes de destino. Dependendo do risco do produto, o TA projeta e executa um *smoke-test* ou um conjunto de testes mais abrangente para verificar se o objeto de teste está adaptado corretamente ao ambiente de destino.

Ao seguir as boas práticas de teste de adaptabilidade (p. ex., definir os ambientes de destino com antecedência, usar testes combinatórios, fazer *smoke-test* em novos ambientes e monitorar defeitos específicos do ambiente), o TA pode descobrir defeitos que podem limitar a vida útil e a usabilidade do software (p. ex., facilidade de uso em diferentes tamanhos de tela). O trabalho do TA em testes de adaptabilidade também deve ser apoiado por testes automatizados entre plataformas implementados por Engenheiros de Automação de Testes. Testes rigorosos de adaptabilidade podem detectar antecipadamente problemas específicos do ambiente, ajudando a evitar atualizações ou reformulações importantes e frequentes após a implementação e reduzindo os custos de manutenção.

O **Teste de Instalabilidade** verifica se o objeto de teste pode ser instalado, desinstalado, atualizado e reconfigurado corretamente em ambientes especificados. O teste de instalabilidade vai além da simples verificação se o procedimento de instalação é executado até o fim.

Os objetivos típicos do teste de instalabilidade em foco para o TA são:

- Para verificar se os procedimentos de instalação são executados corretamente em várias configurações de parâmetros do ambiente. Isso torna úteis as técnicas de teste, como o teste combinatório, semelhante ao teste de adaptabilidade.
- Projetar e executar testes para determinar se o objeto de teste funciona corretamente após a instalação ou atualização
- Verificar se é fácil para os usuários instalarem, desinstalar ou atualizar o software. Isso inclui a revisão da documentação de instalação.
- Para testar o comportamento relacionado a permissões, especialmente para aplicativos móveis (consulte ISTQB-MAT, v1.0).

4.4 Teste de Compatibilidade

O teste de compatibilidade verifica se um objeto de teste é compatível com outros componentes ou sistemas quando é usado. O modelo de qualidade de produto ISO 25010 (ISO/IEC 25010, 2023) diferencia duas subcaracterísticas de compatibilidade: interoperabilidade e coexistência. Portanto, o teste de compatibilidade pode ser subdividido nos seguintes tipos de teste:

- **Testes de Interoperabilidade**, que verificam a compatibilidade com componentes ou sistemas com os quais o objeto de teste deve interagir. Esses testes geralmente são testes funcionais caixa-preta, portanto, o TA geralmente é responsável por eles.

©International Software Testing Qualifications Board

- **Teste de Coexistência**, que verifica se o objeto de teste pode compartilhar seu ambiente de destino com outros componentes ou sistemas sem interferência. Esse tipo de teste técnico é discutido no Technical Test Analyst Syllabus (ISTQB-TTA, v4.0).

4.4.1 Contribuição do Analista de Teste para o Teste de Interoperabilidade

O objetivo do **Teste de Interoperabilidade** é verificar se dois ou mais componentes ou sistemas podem trocar informações e usá-las mutuamente.. Quando a troca de informações envolve uma transformação de dados, o teste de interoperabilidade deve incluir a verificação dessa transformação.

O teste de interoperabilidade é particularmente importante quando vários sistemas precisam colaborar, compartilhar dados ou executar tarefas em conjunto. Esse é especialmente o caso das arquiteturas de software modernas, como soluções em nuvem, serviços da Web, microsserviços, conteinerização e a Internet das Coisas.

A interoperabilidade geralmente ocorre em vários níveis de arquitetura. O TA deve compreender as possíveis interações para definir as condições de teste adequadas para cobri-las. Nem todas as interações podem ser documentadas. O TA pode obter indiretamente informações sobre as interações na documentação de arquitetura e modelagem. Portanto, é fundamental entender essa documentação para garantir que todos os aspectos importantes das interações sejam testados.

Os testes de interoperabilidade podem detectar defeitos de:

- transformações de dados para intercâmbio;
- interpretação ou uso de dados trocados;
- fluxos e protocolos de comunicação;
- conformidade com os padrões;
- funcionalidade de ponta a ponta;
- documentação do projeto.

O teste de interoperabilidade geralmente é aplicado durante o teste de integração. Técnicas de teste caixa-preta, como técnicas de teste baseadas em dados que se concentram nos dados trocados, técnicas de teste baseado no comportamento para interpretar ou usar os dados e funcionalidade de ponta a ponta ou técnicas de teste baseadas em regras para dados transformação, são bem adequados para testes de interoperabilidade. As técnicas de teste baseadas na experiência podem ser úteis complementam o teste caixa-preta. Os casos de teste de testes funcionais ou de adaptabilidade podem ser reutilizados para testes de interoperabilidade.

Exemplos de padrões gerais de interoperabilidade são (ISO 15745, 2003) e (ISO 16100, 2009). O ETSI (ETSI EG 202 237 v1.2.1, 2010) fornece um exemplo de uma metodologia concreta de teste de interoperabilidade no domínio das telecomunicações.

©International Software Testing Qualifications Board

5 Prevenção de Defeitos de Software - 225 min

Palavras-chave

revisão ad hoc, revisão baseada em lista de verificação, prevenção de defeitos, teste baseado em modelo, leitura baseada em perspectiva, técnica de revisão, revisão baseada em função, análise de causa raiz, revisão baseada em cenário, resultado de teste

Objetivos de aprendizagem do Capítulo 5:

5.1 Práticas de prevenção de defeitos

TA-5.1.1 (K2) Explicar como o Analista de Teste pode contribuir para a prevenção de defeitos

5.2 Suporte à contenção de fases

TA-5.2.1 (K3) Usar um modelo do objeto de teste para detectar defeitos em uma especificação

TA-5.2.2 (K3) Aplicar uma técnica de revisão a uma base de teste para encontrar defeitos

5.3 Mitigando a recorrência de defeitos

TA-5.3.1 (K4) Analisar os resultados dos testes para identificar possíveis melhorias na detecção de defeitos

TA-5.3.2 (K2) Explicar como a classificação de defeitos apóia a análise da causa raiz

©International Software Testing Qualifications Board

Introdução

O objetivo da prevenção de defeitos é implementar ações que reduzam a probabilidade de (re)ocorrência de defeitos nos produtos de trabalho e mitiguem a propagação de defeitos para as fases subsequentes do SDLC. Esses esforços resultam em uma variedade de benefícios importantes, incluindo redução de custos e mão de obra, aumento da produtividade e melhoria da qualidade do produto. A prevenção de defeitos é responsabilidade de toda a equipe. O TA pode utilizar seu conhecimento e experiência específicos para contribuir para isso.

As práticas de prevenção de defeitos incluem:

- evitar a introdução de defeitos, que faz parte das atividades de garantia de qualidade;
- evitar que os defeitos escapem para as fases subsequentes do SDLC (consulte a Seção 5.2);
- evitar a recorrência de defeitos (consulte a Seção 5.3).

5.1 Práticas de Prevenção de Defeitos

5.1.1 Contribuição do Analista de Teste para a Prevenção de Defeitos

O TA pode contribuir para a prevenção de defeitos de várias maneiras, usando seu conhecimento de domínio, experiência em testes e habilidades analíticas. Os exemplos incluem:

- Participar da análise de risco: garantindo que os riscos identificados sejam devidamente mitigados (p. ex., selecionando as técnicas de teste mais adequadas).
- Revisão de requisitos, modelos e especificações: permitindo a detecção antecipada de defeitos na base de testes, evitando que eles escapem para o código e, assim, diminuindo significativamente o custo de corrigi-los.
- Participar de retrospectivas: permitindo a identificação de possíveis melhorias na análise de testes, no projeto de testes, na implementação de testes e na execução de testes (p. ex., usar técnicas de teste mais eficazes, direcionar os testes para áreas específicas de risco ou melhorar os dados de teste e os ambientes de teste para reduzir tanto os resultados falso-positivos quanto os falso-negativos) para evitar que os defeitos escapem.
- Coleta e avaliação de dados de defeitos: apoio à análise de causa raiz e melhoria de processos por meio da coleta de dados detalhados sobre defeitos para permitir sua classificação e análise estatística e, assim, facilitar a análise de causa raiz.
- Participar da análise de causa raiz: evitando a recorrência de defeitos ao propor ações corretivas para tratar das causas raiz identificadas.

Além de participar da prevenção de defeitos, o TA avalia (geralmente em consulta com o gerente de testes) se as medidas propostas surtiram o efeito desejado. Exemplos de métricas que ajudam a avaliar a eficácia dessas medidas incluem:

- Eficiência na remoção de defeitos (DRE) - mede a proporção de defeitos removidos antes do lançamento e o número total de defeitos. O denominador (desconhecido) pode ser estimado ou substituído pelo número total de defeitos encontrados até um determinado ponto no tempo (p. ex., até 6 meses após o lançamento). Um DRE alto indica que menos defeitos escapam para a produção, o que pode implicar melhores práticas de prevenção de defeitos. No entanto, o DRE não faz distinção entre defeitos detectados por meio de prevenção e detecção.
- Eficácia de contenção de fase (PCE) - mede o número de defeitos introduzidos e removidos na mesma fase em relação ao número total de defeitos introduzidos nessa fase. Um PCE alto indica que menos defeitos escapam para fases posteriores.
- Custo da qualidade - ilustra a relação entre prevenção de defeitos, detecção de defeitos e custos de remoção (consulte ISTQB-TM, v3.0, Seção 3.2.1).

©International Software Testing Qualifications Board

5.2 Suporte à Contenção de Fases

O objetivo da contenção de fases é detectar e remover defeitos na mesma fase do SDLC em que eles foram introduzidos. No desenvolvimento ágil de software, a abordagem *shift-left* pode ser usada de forma semelhante.

Essa política reduz o custo da qualidade. Como a base de teste é uma entrada importante para a análise e o projeto de teste, o TA pode contribuir melhor para a contenção de fases avaliando a qualidade da base de teste.

A atenção antecipada à qualidade da base de testes minimizará o esforço posterior e evitará a propagação de defeitos para as fases subsequentes. Este syllabus aborda duas opções comuns para o TA encontrar defeitos na base de teste: modelagem para fins de teste e revisão da base de teste usando várias técnicas (ISO/IEC 20246, 2017).

5.2.1 Uso de modelos para detectar defeitos

A modelagem fornece abstrações de um sistema em um determinado nível de precisão e detalhe. Ela ajuda os stakeholders a entender melhor o sistema que está sendo desenvolvido. Conforme discutido abaixo, a modelagem pode apoiar a contenção de fases de pelo menos três maneiras:

- detecção de defeitos nas especificações
- detecção de defeitos em modelos
- detecção de defeitos em objetos de teste usando testes baseados em modelos

Detecção de Defeitos em Especificações. As especificações geralmente são fornecidas como texto escrito informalmente. O TA pode representar formalmente essas especificações usando modelos. Ao criar um modelo, as condições de teste (p. ex., requisitos) são mapeadas para os elementos do modelo e vinculadas a eles para rastreabilidade. A formalização e a visualização das condições de teste em um modelo revelam com eficiência defeitos como incompleta, inconsistente ou ambígua. O ponto forte da modelagem é que a especificação é transformada enquanto as revisões a verificam em sua forma original. Como resultado, o TA também contribui para encontrar soluções adequadas para os defeitos detectados.

Os modelos baseados em dados incluem modelos de domínio e modelos de teste combinatório. Eles permitem que o TA detecte defeitos no domínio, como partições sobrepostas, lacunas na cobertura do domínio, partições vazias ou combinações de parâmetros ausentes ou incorretas.

Os modelos baseados em comportamento incluem matrizes CRUD, modelos baseados em estado ou modelos de cenário. Eles permitem que o TA detecte ciclos de vida de entidades incompletos ou inconsistentes, transições de estado ausentes ou defeituosas, deadlocks, loops infinitos, comportamento ambíguo ou inconsistente do sistema ou falta de tratamento de exceções.

Modelos baseados em regras, como tabelas de decisão ou relações metamórficas, permitem que o TA detecte defeitos nas regras de negócios, como omissões, inconsistências, ambiguidades, redundâncias, cenários propensos a erros ou lógica de negócios complexa.

A modelagem também pode detectar defeitos, como inconsistências na nomenclatura, valores de dados (p. ex., limites) e entradas ou saídas. Ela também pode identificar informações ausentes, incompletas, ambíguas ou desnecessárias.

Detecção de Defeitos em Modelos. Se a base de teste contiver modelos, o TA poderá analisar esses modelos para detectar defeitos. Este syllabus discute a detecção de defeitos em três modelos típicos usados em testes de software: diagramas de transição de estado (consulte a Seção 3.2.2), diagramas de atividade (consulte a Seção 3.2.3) e tabelas de decisão (consulte a Seção 3.3.1). Exemplos de defeitos nos modelos mencionados acima incluem:

- diagramas de transição de estado: estados ausentes/errados, transições impróprias, condições ou ações de proteção incorretas, estados redundantes ou inalcançáveis e comportamento não determinístico
- diagramas de atividade: ações ausentes, inalcançáveis ou sem saída, ordem incorreta de ações, condições de proteção erradas, não exclusivas ou incompletas em nós de decisão, pontos de sincronização ausentes ou fluxos paralelos sincronizados incorretamente que podem levar a um comportamento não intencional

©International Software Testing Qualifications Board

- tabelas de decisão: regras sobrepostas, inconsistentes ou inviáveis, incompletude (p. ex., combinações de condições ausentes ou ações ausentes para uma determinada combinação de condições)

Todos os modelos também podem conter defeitos, como erros de sintaxe, erros de digitação, duplicatas e nomeação inconsistente de elementos do modelo.

Detecção de Defeitos usando o Teste Baseado em Modelo (MBT). Nessa abordagem de teste, uma ferramenta de MBT projeta e gera casos de teste e outros softwares de teste com base em um modelo de MBT e em critérios de seleção de teste definidos pelo TA. O MBT é eficaz para encontrar anomalias na especificação porque permite uma cobertura abrangente e a exploração sistemática do comportamento esperado do objeto de teste de acordo com o modelo MBT. Os modelos de MBT, especialmente os gráficos, promovem a comunicação com os stakeholders, criando uma percepção e um entendimento comuns da base de teste. Mais detalhes sobre MBT podem ser encontrados no Model-Based Tester Syllabus (ISTQB-MBT, v1.1).

5.2.2 Aplicação de Técnicas de Revisão

Uma base de testes bem definida é a pedra angular para garantir a qualidade dos produtos de trabalho e o sucesso dos projetos. A revisão da base de testes ajuda a identificar e tratar os defeitos com antecedência, evitando que eles passem para as fases subsequentes.

O TA pode empregar várias técnicas de revisão durante a revisão individual para identificar defeitos na base de testes. A seleção da técnica de revisão mais adequada pode aumentar a eficiência e a eficácia da revisão. Essa seleção deve considerar fatores como metas de revisão, objetivos do projeto, recursos disponíveis, tipo de base de teste, riscos associados, domínio do negócio e cultura da empresa. A seguir, este syllabus discute cinco técnicas de revisão comumente usadas pelo TA.

A **Revisão ad hoc** é realizada pelos revisores de maneira informal, sem um processo estruturado. Os revisores recebem pouca ou nenhuma orientação sobre como a tarefa deve ser realizada. A revisão ad hoc precisa de pouca preparação e é altamente dependente das habilidades dos revisores. Durante a revisão, os revisores leem a base do teste e documentam as anomalias à medida que as encontram. Essa técnica, se não for gerenciada, pode levar a um grande volume de relatórios de anomalias duplicados de vários revisores.

A **Revisão Baseada em Lista de Verificação** envolve a avaliação da base do teste em relação a uma lista de verificação predefinida. As listas de verificação lembram os revisores de verificar pontos específicos e podem despersonalizar a revisão. As listas de verificação podem ser genéricas ou específicas para características de qualidade, objetivos de teste ou tipo de base de teste. O TA adapta a lista de verificação ao tipo de base de teste, ao nível de risco ou à condição do teste. Isso garante que a revisão se concentre nos aspectos mais relevantes da base de teste. As listas de verificação devem ser atualizadas regularmente com os defeitos não detectados anteriormente. Manter as listas de verificação atualizadas evita que anomalias recém-identificadas passem despercebidas. As listas de verificação não são completas, portanto, o TA não se limita a verificar os itens listados. Isso maximiza a detecção de defeitos e permite que o TA capture anomalias que as listas de verificação podem não abranger explicitamente.

A **Revisão Baseada em Cenários** envolve a simulação de um processo ou atividade para identificar anomalias e refinar a base de teste. Essa técnica de revisão é mais eficaz quando a base de teste tem um formato baseado em cenários, como um caso de uso ou um diagrama de atividades. Nesses casos, os revisores podem realizar "testes secos" com base no uso esperado do produto de trabalho. Os cenários fornecem diretrizes valiosas, mas os revisores não estão limitados aos cenários documentados e devem pensar além dos cenários para identificar mais anomalias.

A **Revisão Baseada em Funções** envolve a atribuição de funções ou responsabilidades específicas aos revisores. As funções típicas são baseadas em tipos específicos de usuários finais (p. ex., usuário experiente, inexperiente, sênior ou infantil) ou em funções específicas na organização (p. ex., administrador ou usuário comum). Cada função pode ser descrita por uma persona (ou seja, o personagem concreto, mas fictício, criado para representar as características, as necessidades, as metas e as preferências de um determinado grupo de usuários). Ao distribuir as responsabilidades entre os revisores com base em suas funções, as revisões baseadas em funções permitem que os indivíduos se concentrem em aspectos específicos da base de testes, garantindo uma cobertura abrangente e, ao mesmo tempo, evitando a duplicação de anomalias.

A **Leitura Baseada em Perspectiva** envolve a revisão da base de teste a partir de várias perspectivas ou pontos de vista (p. ex., designer, testador, administrador e usuário final). Isso leva a uma revisão individual mais aprofundada com menos duplicação de anomalias entre os revisores. Além disso, a leitura baseada em

©International Software Testing Qualifications Board

perspectiva exige que os revisores tentem usar a base de teste em análise para gerar o produto de trabalho que eles obteriam a partir dela. Por exemplo, um testador tentaria gerar rascunhos de testes de aceite com base nas especificações de requisitos para verificar se todas as informações necessárias estão incluídas.

5.3 Mitigando a recorrência de Defeitos

Um TA pode ajudar proativamente a minimizar a recorrência de defeitos no software. Este syllabus discute duas abordagens relacionadas à mitigação da recorrência de defeitos: analisar os resultados dos testes para melhorar a análise e o projeto dos testes e apoiar a análise da causa raiz com a classificação dos defeitos.

5.3.1 Análise dos resultados dos testes para melhorar a Detecção de Defeitos

Os resultados dos testes permitem que o TA identifique as falhas, mas também fornecem feedback ao TA para ajudar a melhorar a eficácia da detecção de defeitos. Algumas técnicas comumente usadas para analisar os resultados dos testes são descritas a seguir.

Análise de agrupamento de defeitos previstos versus reais. Alguns componentes geralmente contêm a maioria dos defeitos (consulte ISTQB-CTFL, v4.0.1, Seção 1.3). Após o teste, o TA pode prever áreas propensas a defeitos e comparar os agrupamentos de defeitos previstos com os reais. No caso de discrepâncias, testes mais rigorosos podem ser aplicados às áreas em que foram encontrados mais defeitos do que o esperado. Critérios mensuráveis devem garantir clareza e consistência ao se determinar os agrupamentos (p. ex., densidade e gravidade dos defeitos). Entre esses critérios, a gravidade dos defeitos deve desempenhar um papel significativo. Pequenos grupos de defeitos críticos ou principais geralmente são mais importantes (ou seja, precisam de testes mais rigorosos) do que grupos maiores de defeitos menores ou cosméticos.

Análise da porcentagem de detecção de defeitos (DDP). A DDP é uma das medidas de eficácia de teste mais importantes para um nível de teste. Ao calcular a DDP, o número de defeitos escapados deve ser limitado àqueles que o nível de teste em consideração poderia ter detectado. Limites claros para a contagem de defeitos, como limites temporais (p. ex., defeitos encontrados dentro de um período definido após o lançamento) e critérios de exclusão (p. ex., defeitos em componentes de terceiros ou ambientes específicos do cliente), devem ser estabelecidos para garantir a consistência. Um DDP baixo para um determinado nível de teste indica uma alta porcentagem de defeitos que escaparam, o que significa que a detecção de defeitos é ineficaz. Nesse caso, o TA deve analisar os motivos e propor medidas para melhorá-lo, tornando o nível de teste mais focado e rigoroso. A melhor maneira de dividir o DDP é por níveis de gravidade, pois a prioridade de reduzir os defeitos que escaparam geralmente depende de sua gravidade.

Análise de cobertura estrutural. Avalia a extensão em que os testes exercitaram áreas específicas do objeto de teste. A identificação de áreas com baixa cobertura permite que o TA direcione os esforços de teste para essas áreas. O aumento da cobertura ajuda a descobrir novos defeitos, que antes não eram detectados. A cobertura estrutural, como a cobertura de instruções, a cobertura de ramificações ou a cobertura de neurônios, geralmente é medida com ferramentas de teste. Ao selecionar as áreas adicionais a serem cobertas, seus níveis de risco devem ser considerados.

Análise de lacunas de teste. Avalia até que ponto os testes exercitaram alterações recentes no código. Isso permite que o TA concentre o esforço de teste adicional em áreas especialmente propensas a erros (ou seja, novas alterações que não foram testadas), em vez de visar todas as áreas com baixa cobertura (p. ex., código que não foi alterado há muito tempo e que foi testado em versões anteriores).

A**nálise do padrão de chegada de defeitos**. O número ou a densidade de defeitos encontrados em fases sucessivas de um projeto (p. ex., iterações) pode ser comparado com padrões que descrevem a distribuição teórica desses valores ao longo do tempo. Um exemplo clássico de um padrão de chegada de defeitos é o modelo Rayleigh (Elsayed, 2021). Ele tem um único pico e é inclinado para a direita, mostrando que o número esperado de defeitos encontrados primeiro aumenta com o tempo e, depois de atingir seu valor máximo, cai lentamente em direção a zero. Com base na análise desse padrão, é possível inferir a força dos casos de teste existentes e o potencial para sua melhoria. Por exemplo, suponha que a detecção de defeitos permaneça em um nível baixo e constante quando o padrão sugere que ela deveria estar aumentando. Nesse caso, isso pode significar que os testes existentes são muito fracos e incapazes de detectar defeitos adicionais.

Os métodos analíticos descritos acima usam várias métricas relacionadas a defeitos, como o número de defeitos ou DDP. Essas métricas são calculadas com base nos resultados dos testes (p. ex., número de testes

©International Software Testing Qualifications Board

aprovados/reprovados), relatórios de defeitos e métricas estruturais (p. ex., cobertura de código ou densidade de defeitos). Observe, no entanto, que essas métricas nem sempre são tão fáceis de ler nos resultados dos testes quanto parecem. Por exemplo, o número de testes com falha não é necessariamente o mesmo que o número de defeitos detectados por esses testes. Para calcular o número real de defeitos detectados, os resultados do processo de depuração devem ser analisados cuidadosamente, pois a relação entre os resultados dos testes e os defeitos pode ser de muitos para muitos. Vários testes podem detectar o mesmo defeito ou um teste pode detectar vários defeitos. Além disso, a gravidade dos defeitos pode ser diferente da criticidade dos testes, pois um caso de teste crítico pode falhar devido a um defeito cosmético.

5.3.2 Apoio à Análise de Causa Raiz com Classificação de Defeitos

A análise de causa raiz (RCA) é uma técnica para identificar e abordar as causas subjacentes ou fundamentais de um defeito, e não apenas seus sintomas. A RCA apoia uma abordagem estruturada para a melhoria da qualidade e seu principal objetivo é evitar a recorrência de defeitos. A TA usa várias técnicas para identificar as causas básicas de defeitos e falhas (p. ex., taxonomias de defeitos, a técnica dos cinco porquês, diagramas de causa e efeito e análise de Pareto).

A RCA clássica envolve especialistas no assunto que estudam um defeito em detalhes consideráveis depois de resolvê-lo. No entanto, geralmente há muitos defeitos que precisam ser analisados. Portanto, seria muito ineficiente e demorado ter um planejamento de ação preventiva para cada defeito. Uma maneira de abordar esse problema é classificar os defeitos e, em seguida, executar a RCA para os tipos de defeitos que estão ocorrendo.

A classificação de defeitos baseia-se no reconhecimento de que os defeitos individuais capturam uma grande quantidade de informações sobre o processo de desenvolvimento e o sistema em teste. A classificação de defeitos permite que o TA extraia informações sobre vários aspectos do processo de desenvolvimento a partir do defeito e o transforme em uma medida de processo. Isso, por sua vez, fornece uma visão dos tipos de erros cometidos durante o desenvolvimento, o que é útil para o aprimoramento do processo. A classificação de defeitos preenche a lacuna entre as estatísticas quantitativas de defeitos e a RCA qualitativa. Para apoiar efetivamente a RCA, os defeitos devem ser classificados de maneira uniforme em todo o SDLC, desde os primeiros testes até a produção.

O TA deve apoiar sua organização na padronização da classificação de defeitos de software. Isso melhorará a comunicação e a troca de informações sobre defeitos entre desenvolvedores e organizações, facilitando a RCA.

Exemplos de métodos de classificação de defeitos são:

- Classificação Ortogonal de Defeitos (ODC) (Chillarege, 1992), que classifica cada defeito em atributos ortogonais (ou seja, mutuamente exclusivos), coletados quando um defeito é relatado e quando é corrigido
- IEEE 1044 (IEEE 1044, 2009), uma classificação padrão para anomalias de software que fornece um conjunto básico de atributos para a classificação de falhas e defeitos
- Classificação Baseada em Gravidade, que classifica os defeitos com base em sua gravidade (p. ex., crítico, maior, menor, trivial)
- Modelos de Taxonomia de Defeitos, como (Beizer, 1990) ou (Catolino et al., 2019)

Os defeitos também podem ser mapeados para atributos de qualidade usando modelos de qualidade de software, como (ISO/IEC 25010, 2023) ou o modelo FURPS (Grady et al., 1987).

Mais informações sobre RCA podem ser encontradas em (ISTQB-ITP, v1.0).

©International Software Testing Qualifications Board

6 Referências

Normas

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.

Documentos ISTQB®

ISTQB-ACT, ISTQB®, 2019. Certified Tester Specialist Mobile Acceptance Testing: Syllabus. Version 1.0. ISTQB-AI, ISTQB®, 2021. Certified Tester AI Testing: Syllabus. Version 1.0.

ISTQB-CTFL, ISTQB®, 2024. Certified Tester Foundation Level: Syllabus. Version 4.0.1.

ISTQB-ITP, ISTQB®, 2011. Certified Tester Expert Level Improving the Test Process: Syllabus. Version 1.0.

ISTQB-MAT, ISTQB®, 2019. Certified Tester Specialist Mobile Application Testing: Syllabus. Version 1.0. ISTQB-MBT, ISTQB®, 2024. Certified Tester Model-Based Tester: Syllabus. Version 1.1.

ISTQB-PT, ISTQB®, 2018. Certified Tester Performance Testing: Syllabus. Version 1.0.

ISTQB-TAE, ISTQB®, 2024. Certified Tester Test Automation Engineering: Syllabus. Version 2.0. ISTQB-TM, ISTQB®, 2024. Certified Tester Advanced Level Test Management: Syllabus. Version 3.0.

©International Software Testing Qualifications Board

ISTQB-TTA, ISTQB®, 2021. Certified Tester Advanced Level Technical Test Analyst: Syllabus. Version 4.0.
ISTQB-UT, ISTQB®, 2018. Certified Tester Usability Testing: Syllabus. Version 1.0.

Livros

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.

Artigos

ANPD, Brasília, 2023, https://www.gov.br/anpd/pt-br/centrais-de-conteudo/documentos-tecnicos-orientativos/estudo_tecnico_sobre_anonimizacao_de_dados_na_lgpd_uma_visao_de_processo_baseado_em_risco_e_tecnicas_computacionais.pdf

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.

©International Software Testing Qualifications Board

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.

LBI, 2013 – Lei nº 13.146, http://www.planalto.gov.br/ccivil_03/_Ato2015-2018/2015/Lei/L13146.htm

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.

Web Pages

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/.

©International Software Testing Qualifications Board

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.

©International Software Testing Qualifications Board

7 Apêndice A - Objetivos de Aprendizagem e Nível Cognitivo de Conhecimento

Os objetivos de Aprendizagem (LO) deste syllabus são mostrados no início de cada capítulo. Cada tópico do syllabus será examinado de acordo com seu LO.

Os Objetivos de Aprendizagem começam com um verbo de ação correspondente ao seu nível cognitivo de conhecimento, conforme listado abaixo.

Nível 1: Lembrar (K1)

O candidato se lembrará, reconhecerá e recordará um termo ou conceito.

Verbos de ação: Recordar, reconhecer.

Observação: Os syllabis de nível avançado não têm Objetivos de aprendizagem específicos de nível K1. Entretanto, o conteúdo do syllabus nos capítulos 1 a 5 e as definições de todos os termos listados como palavras-chave logo abaixo dos títulos dos capítulos devem ser lembrados (K1), mesmo que não sejam explicitamente mencionados nos objetivos de aprendizado.

Nível 2: Compreender (K2)

O candidato pode selecionar as razões ou explicações para declarações relacionadas ao tópico e pode resumir, comparar, classificar e dar exemplos para o conceito de teste.

Verbos de ação: Classificar, comparar, diferenciar, distinguir, explicar, dar exemplos, interpretar, resumir

Exemplos	Notas
Diferenciar entre testes de correção funcional, adequação funcional e integridade funcional	Procura diferenças entre os conceitos.
Explicar o teste CRUD	
Dê exemplos de requisitos de ambiente de teste	
Resumir o envolvimento do Analista de Teste em vários ciclos de vida de desenvolvimento de software	

Nível 3: Aplicar (K3)

O candidato pode executar um procedimento quando confrontado com uma tarefa familiar ou selecionar o procedimento correto e aplicá-lo a um determinado contexto.

Verbos de ação: Aplicar, implementar, preparar, usar

Exemplos	Notas
Aplicar testes de domínio	Deve se referir a um procedimento / técnica / processo etc.
Preparar cartas de teste para testes baseados em sessões	
Usar testes orientados por palavras-chave para desenvolver scripts de teste	Pode ser usado em um LO que deseja que o candidato seja capaz de usar uma técnica ou um procedimento. Semelhante a "aplicar"

Nível 4: Analisar (K4)

O candidato pode separar as informações relacionadas a um procedimento ou técnica em suas partes constituintes para melhor compreensão e pode distinguir entre fatos e inferências. Uma aplicação típica é analisar um documento, software ou situação de projeto e propor ações apropriadas para resolver um problema ou tarefa.

Verbos de ação: Analisar, desconstruir, delinear, priorizar, selecionar.

©International Software Testing Qualifications Board

Exemplos	Notas
Analisar o impacto das alterações para determinar o escopo dos testes de regressão	Examinável somente em combinação com um objetivo mensurável da análise. Deve ser do tipo "Analisar X para Y" (ou similar).
Selecionar técnicas de teste adequadas para reduzir os riscos do produto em uma determinada situação	Necessário quando a seleção requer análise.

Referência

(Para os níveis cognitivos dos objetivos de aprendizagem)

Anderson, L. W. e Krathwohl, D. R. (eds) (2001) A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives (Uma revisão da taxonomia de Bloom dos objetivos educacionais), Allyn & Bacon

©International Software Testing Qualifications Board

8 Apêndice B: Matriz de rastreabilidade entre resultados de negócio x objetivos de aprendizagem

Esta seção lista o número de Objetivos de Aprendizagem do Nível Fundamental relacionados aos Resultados de Negócios e a Rastreabilidade entre os Resultados de Negócios do Nível Fundamental, e os Objetivos de Aprendizagem de Nível Fundamental.

Resultados do negócio: Analista de Teste de nível avançado		TA-BO1	TA-BO2	TA-BO3	TA-BO4	TA-BO5	TA-BO6	TA-BO7	TA-BO8	TA-BO9
TA-BO1	Apoiar e realizar testes apropriados com base no ciclo de vida de desenvolvimento de software seguido	6								
TA-BO2	Aplicar os princípios de testes baseados em riscos		3							
TA-BO3	Selecionar e aplicar técnicas de teste adequadas para apoiar a realização dos objetivos do teste			13						
TA-BO4	Fornecer documentação com níveis adequados de detalhes e qualidade				5					
TA-BO5	Determinar os tipos apropriados de testes funcionais a serem realizados					1				
TA-BO6	Contribuir para testes não funcionais						3			
TA-BO7	Contribuir para a prevenção de defeitos							5		
TA-BO8	Melhorar a eficiência do processo de teste com o uso de ferramentas								4	
TA-BO9	Especificar os requisitos para ambientes de teste e dados de teste									2

©International Software Testing Qualifications Board

Resultados do negócio: Advanced Level Test Analyst		TA-BO1	TA-BO2	TA-BO3	TA-BO4	TA-BO5	TA-BO6	TA-BO7	TA-BO8	TA-BO9
Número LO	Objetivo de aprendizado (nível K)									
1	As tarefas do Analista de Teste no processo de teste - 225 minutos									
1.1	Testes no ciclo de vida do desenvolvimento de software									
TA-1.1.1	Resumir o envolvimento do Analista de Teste em vários ciclos de vida de desenvolvimento de software (K2)	X								
1.2	Envolvimento em atividades de teste									
TA-1.2.1	Resumir as tarefas realizadas pelo Analista de Teste como parte da análise de testes (K2)	X								
TA-1.2.2	Resumir as tarefas realizadas pelo Analista de Teste como parte do projeto de testes (K2)	X								
TA-1.2.3	Resumir as tarefas realizadas pelo Analista de Teste como parte da implementação do teste (K2)	X								
TA-1.2.4	Resumir as tarefas realizadas pelo Analista de Teste como parte da execução do teste (K2)	X								
1.3	Tarefas relacionadas a produtos de trabalho									
TA-1.3.1	Diferenciar entre casos de teste de alto nível e casos de teste de baixo nível (K2)				X					
TA-1.3.2	Explicar os critérios de qualidade para casos de teste (K2)				X					
TA-1.3.3	Dê exemplos de requisitos de ambiente de teste (K2)				X					X
TA-1.3.4	Explicar o problema do oráculo de teste e as possíveis soluções (K2)			X	X					
TA-1.3.5	Dê exemplos de requisitos de dados de teste (K2)				X					X
TA-1.3.6	Usar testes orientados por palavras-chave para desenvolver scripts de teste (K3)	X							X	
TA-1.3.7	Resumir os tipos de ferramentas para gerenciar o material de teste (K2)								X	
2	As tarefas do Analista de Teste em testes baseados em riscos - 90 minutos									
2.1	Análise de risco									
TA-2.1.1	Resumir a contribuição do Analista de Teste para a análise de risco do produto (K2)		X							
2.2	Controle de riscos									
TA-2.2.1	Analisar o impacto das alterações para determinar o escopo do teste de regressão (K4)		X						X	

©International Software Testing Qualifications Board

Resultados do negócio: Advanced Level Test Analyst		TA-BO1	TA-BO2	TA-BO3	TA-BO4	TA-BO5	TA-BO6	TA-BO7	TA-BO8	TA-BO9
3	Análise e projeto de testes - 615 minutos									
3.1	Técnicas de teste baseadas em dados									
TA-3.1.1	Aplicar testes de domínio (K3)			X						
TA-3.1.2	Aplicar testes combinatórios (K3)			X						
TA-3.1.3	Resumir os benefícios e as limitações dos testes aleatórios (K2)			X						
3.2	Técnicas de teste baseado no comportamento									
TA-3.2.1	Explicar o teste CRUD (K2)			X						
TA-3.2.2	Aplicar o teste de transição de estado (K3)			X						
TA-3.2.3	Aplicar teste baseado em cenário (K3)			X						
3.3	Técnicas de teste baseadas em regras									
TA-3.3.1	Aplicar o teste de tabela de decisão (K3)			X						
TA-3.3.2	Aplicar testes metamórficos (K3)			X						
3.4	Testes baseados em experiência									
TA-3.4.1	Preparar cartas de teste para testes baseados em sessões (K3)			X						
TA-3.4.2	Preparar listas de verificação que apoiem os testes baseados em experiência (K3)			X						
TA-3.4.3	Dê exemplos dos benefícios e das limitações dos testes de multidão (K2)			X						
3.5	Aplicação das técnicas de teste mais apropriadas									
TA-3.5.1	Selecionar técnicas de teste adequadas para reduzir os riscos do produto em uma determinada situação (K4)		X	X						
TA-3.5.2	Explicar os benefícios e os riscos de automatizar o projeto de teste (K2)								X	
4	Teste de características de qualidade - 60 minutos									
4.1	Teste funcional									
TA-4.1.1	Diferenciar entre testes de correção funcional, adequação funcional e integridade funcional (K2)					X				
4.2	Teste de usabilidade									

©International Software Testing Qualifications Board

Resultados do negócio: Advanced Level Test Analyst		TA-BO1	TA-BO2	TA-BO3	TA-BO4	TA-BO5	TA-BO6	TA-BO7	TA-BO8	TA-BO9
TA-4.2.1	Explicar como o Analista de Teste contribui para os testes de usabilidade (K2)						X			
4.3	Teste de flexibilidade									
TA-4.3.1	Explicar como o Analista de Teste contribui para os testes de adaptabilidade e instalabilidade (K2)						X			
4.4	Teste de compatibilidade									
TA-4.4.1	Explicar como o Analista de Teste contribui para os testes de interoperabilidade (K2)						X			
5	Prevenção de defeitos de software - 225 minutos									
5.1	Práticas de prevenção de defeitos									
TA-5.1.1	Explicar como o Analista de Teste pode contribuir para a prevenção de defeitos (K2)							X		
5.2	Suporte à contenção de fases									
TA-5.2.1	Usar um modelo do objeto de teste para detectar defeitos em uma especificação (K3)							X		
TA-5.2.2	Aplicar uma técnica de revisão a uma base de teste para encontrar defeitos (K3)							X		
5.3	Mitigando a recorrência de defeitos									
TA-5.3.1	Analisar os resultados dos testes para identificar possíveis melhorias na detecção de defeitos (K4)							X		
TA-5.3.2	Explicar como a classificação de defeitos apoia a análise da causa raiz (K2)							X		

©International Software Testing Qualifications Board

9 Apêndice C: Notas de versão

O ISTQB® Advanced Level Test Analyst Syllabus v4.0 é uma grande atualização. Por esse motivo, não há notas de versão detalhadas por capítulo e seção. Entretanto, um resumo das principais alterações é fornecido abaixo. Além disso, em um documento separado de Notas de Versão, o ISTQB® fornece rastreabilidade detalhada entre os objetivos de aprendizagem nas versões v3.1.2 e v4.0 do syllabus do Advanced Level Test Analyst.

Essa versão principal fez as seguintes alterações:

Estrutura do syllabus

Todos os objetivos de aprendizagem foram editados para torná-los atômicos e para criar uma rastreabilidade de um para um dos objetivos de aprendizagem para o conteúdo, a fim de evitar a existência de conteúdo sem um objetivo de aprendizagem correspondente. Cada objetivo de aprendizagem é descrito em uma seção com o mesmo número (p. ex., TA-1.1.1 é discutido na Seção 1.1.1). A meta era tornar esta versão mais fácil de ler, entender, aprender e traduzir, concentrando-se em aumentar a utilidade prática e o equilíbrio entre conhecimento e habilidades.

Há 36 Objetivos de aprendizagem na v4.0, em comparação com 31 na v3.1.2. Vários objetivos de aprendizagem K4 foram reduzidos para K2 ou K3. No caso dos Objetivos de aprendizagem relacionados às técnicas de teste, a motivação foi que o foco deveria estar na aplicação das técnicas de teste e não na análise da documentação para um projeto de teste adicional. Alguns Objetivos de aprendizagem foram fundidos em um único objetivo de aprendizado. A tabela abaixo mostra estatísticas detalhadas sobre o número de objetivos de aprendizagem e o tempo de treinamento.

Versão do syllabus	#K2 LO (tempo)	#K3 LO (tempo)	#K4 LO (tempo)	#Total LO (tempo)
atual (4.0)	22 (330 min)	11 (660 min)	3 (225 min)	36 (1215 min)
anterior (3.1.2)	16 (240 min)	5 (300 min)	10 (750 min)	31 (1290 min)

Classificação das técnicas de teste

A estrutura do Capítulo 3 reflete uma classificação mais detalhada das técnicas de teste em comparação com a versão 3.1.2. As técnicas de teste caixa-preta agora são categorizadas como baseadas em dados, em comportamento e em regras, de acordo com o tipo de modelo de objeto de teste subjacente.

Ampliação do escopo do Capítulo 5

O antigo Capítulo 5 (dedicado exclusivamente a revisões) foi ampliado para discutir outras formas essenciais de práticas de prevenção de defeitos usadas pela TA, como o uso de modelos para detectar defeitos em especificações, a análise de resultados de testes para melhorar a detecção de defeitos e o uso da classificação de defeitos para apoiar a análise da causa raiz.

Mudanças nos objetivos de aprendizado

- O Capítulo 1 foi reorganizado em uma seção sobre atividades de teste e outra sobre testware.
- O tópico sobre casos de teste de alto nível e casos de teste de baixo nível (antigo TA-1.4.2) foi rebaixado de K4 para K2 (TA-1.3.1).
- O K3 LO sobre testes baseados em riscos (antigo TA-2.1.1) foi dividido em dois, o K2 TA-2.1.1 relacionado à análise de riscos e o K4 TA-2.2.1 relacionado ao controle de riscos.
- O particionamento de equivalência (antigo K4 TA-3.2.1) e a análise de valor de limite (antigo K4 TA-3.2.2) foram substituídos pelo K3 TA-3.1.1, mais geral, sobre teste de domínio.
- Os testes de pares (antigo K4 TA-3.2.6) e os diagramas de árvore de classificação (antigo K2 TA-3.2.5) foram fundidos em um único K3 TA-3.1.2 mais geral sobre testes combinatórios.

©International Software Testing Qualifications Board

- O teste de transição de estado (antigo K4 TA-3.2.4) foi substituído pelo K3 TA-3.2.2, que se concentra na cobertura de N-switch e de ida e volta. As redundâncias com o CTFL foram removidas.
- O teste de casos de uso (antigo K4 TA-3.2.7) foi substituído pelo K3 TA-3.2.3, mais geral, sobre teste baseado em cenário.
- O teste da tabela de decisão (antigo K4 TA-3.2.3) foi rebaixado para K3 (TA-3.3.1).
- Os testes exploratórios (antigo K3 TA-3.3.2) foram substituídos por dois OAs separados: sobre a preparação de cartas de teste (K3 TA-3.4.1) e sobre a preparação de listas de verificação que apoiam os testes baseados em experiência (K3 TA-3.4.2). As redundâncias com o atual CTFL v4.0 foram removidas.
- Quatro OAs relacionados à comparação de técnicas de teste e à aplicação da técnica mais adequada (antigos: K4 TA-3.2.8, K2 TA-3.3.3, K2 TA-3.4.1, K2 TA-3.3.1) foram combinados em um único K4 TA-3.5.1.
- Quatro OAs sobre testes funcionais (antigos: K2 TA-4.2.1, K2 TA-4.2.2, K2 TA-4.2.3 e K4 TA-4.2.7) foram combinados em um único K2 TA-4.1.1. O tópico foi simplificado porque o teste funcional já está amplamente descrito no contexto das técnicas de teste no Capítulo 3.
- Os tópicos do Capítulo 6 (Ferramentas de teste) foram transferidos para a Seção 1.3, concentrando-se em questões mais práticas. O antigo K3 TA-6.2.1 "Para um determinado cenário, determine as atividades apropriadas para um Analista de Teste em um projeto de teste orientado por palavras-chave" foi ligeiramente alterado para K3 TA-1.3.6 "Use testes orientados por palavras-chave para desenvolver scripts de teste". O antigo K2 TA-6.3.1 "Explicar o uso e os tipos de ferramentas de teste aplicadas no projeto de teste, na preparação dos dados de teste e na execução do teste" foi ligeiramente alterado para K2 TA-1.3.7 "Resumir os tipos de ferramentas aplicadas no gerenciamento do material de teste".
- Dois OAs semelhantes sobre revisões (antigos K3 TA-5.2.1 e K3 TA-5.2.2) foram combinados em um único K3 TA-5.2.2.

Novos tópicos

- Critérios de qualidade para casos de teste (K2 TA-1.3.2)
- Requisitos do ambiente de teste (K2 TA-1.3.3)
- Determinação de oráculos de teste (K2 TA-1.3.4)
- Requisitos de dados de teste (K2 TA-1.3.5)
- Testes aleatórios (K2 TA-3.1.3)
- Teste CRUD (K2 TA-3.2.1)
- Testes metamórficos (K3 TA-3.3.2)
- Teste de multidão (K2 TA-3.4.3)
- Benefícios e riscos da automação do projeto de teste (K2 TA-3.5.2)
- Contribuições do Analista de Teste para a prevenção de defeitos (K2 TA-5.1.1)
- Uso de modelos para detectar defeitos em especificações (K3 TA-5.2.1)
- Análise dos resultados dos testes para melhorar a detecção de defeitos (K4 TA-5.3.1)
- Apoio à análise de causa raiz com classificação de defeitos (K2 TA-5.3.2)

Atualizações de Normas

As alterações nas versões mais recentes dos padrões internacionais de qualidade de software (ISO/IEC 25010, 2023) e técnicas de teste (ISO/IEC/IEEE 29119-4, 2021) levaram à adaptação do conteúdo correspondente do syllabus.

©International Software Testing Qualifications Board

10 Apêndice D - Lista de abreviações

Abreviação	Significado
IA	inteligência artificial
BPMN	Modelo e notação de processos de negócios
BVA	análise de valor de contorno
CRUD	criar, ler, atualizar e excluir
CSV	valor separado por vírgula
DDP	porcentagem de detecção de defeitos
DRE	eficiência na remoção de defeitos
EP	particionamento de equivalência
GDPR	Regulamento Geral de Proteção de Dados
JSON	Notação de objeto JavaScript
MBT	testes baseados em modelos
RM	relação metamórfica
MT	testes metamórficos
ODC	classificação ortogonal de defeitos
PCE	eficácia da contenção de fases
RCA	análise de causa raiz
SDLC	ciclo de vida de desenvolvimento de software
TA	analista de testes
TTA	analista de testes técnicos
UML	Linguagem de modelagem unificada
UX	experiência do usuário
XML	Linguagem de marcação extensível

©International Software Testing Qualifications Board

11 Apêndice E - Termos específicos do domínio

Prazo	Definição
Matriz CRUD	Uma matriz que indica os tipos de ação das funções nas entidades de um sistema.
semântica de dados	O significado e a interpretação dos dados.
técnica dos cinco porquês	Uma técnica interrogativa iterativa usada para determinar a causa raiz de um defeito ou problema, repetindo a pergunta "por quê?" cinco vezes, cada vez direcionando o "por quê" atual para a resposta do "por quê" anterior.
Análise de Pareto	Uma abordagem para identificar áreas problemáticas ou tarefas que terão o maior retorno.
mapa da jornada do usuário	Documentação de visualização da experiência do usuário que mostra as etapas que um usuário executa em um processo para atingir uma meta.
pesquisa com usuários	Disciplina que consiste em aprender sobre as necessidades e os processos de pensamento dos usuários estudando como eles realizam as tarefas, observando como interagem com um componente ou sistema ou por meio da análise e interpretação de dados.

©International Software Testing Qualifications Board

12 Apêndice F - Modelo de qualidade de software

A tabela abaixo mostra as características de qualidade do modelo de qualidade de produto ISO/IEC 25010 (ISO/IEC 25010, 2023). Ela indica quais características/subcaracterísticas são abordadas neste syllabus (TA) e quais são abordadas em outros syllabis do ISTQB® (Technical Test Analyst (TTA), Performance Testing (PT), Usability Testing (UT), Automotive Software Tester (AuT) e Security Tester (SEC)). Se vários syllabi abordarem uma característica de qualidade, aquele que a abordar com mais detalhes será listado primeiro. A tabela também compara o modelo atual da ISO/IEC 25010 com a versão de 2011 usada na versão anterior deste syllabus.

ISO/IEC 25010:2023 (atual)	ISO/IEC 25010:2011 (anterior)	Notas	ISTQB® syllabi
Adequação funcional	**Adequação funcional**		**TA**
Completude funcional	Completude funcional		
Correção funcional	Correção funcional		
Adequação funcional	Adequação funcional		
Eficiência de performance	**Eficiência de performance**		PT, TTA
Comportamento no tempo	Comportamento no tempo		
Utilização de recursos	Utilização de recursos		
Capacidade	Capacidade		
Compatibilidade	**Compatibilidade**		TA, TTA
Coexistência	Coexistência		TTA
Interoperabilidade	Interoperabilidade		**TA**
Capacidade de interação	**Usabilidade**	**Renomeado**	**UT, TA**
Adequação e reconhecimento	Adequação e reconhecimento		
Capacidade de aprendizado	Capacidade de aprendizado		
Operabilidade	Operabilidade		
Proteção contra erros do usuário	Proteção contra erros do usuário		
Envolvimento do usuário	Estética da interface do usuário	Renomeado	
Inclusão	Acessibilidade	Dividido e renomeado	
Assistência ao usuário			
Autodescrição		Novo	
Confiabilidade	**Confiabilidade**		TTA
Inexistência de falhas	Maturidade	Renomeado	
Disponibilidade	Disponibilidade		
Tolerância a falhas	Tolerância a falhas		
Recuperabilidade	Recuperabilidade		
Segurança	**Segurança**		SEC, TTA
Confidencialidade	Confidencialidade		
Integridade	Integridade		
Não repúdio	Não repúdio		
Responsabilidade	Responsabilidade		
Autenticidade	Autenticidade		
Resistência		Novo	
Manutenibilidade	**Manutenibilidade**		TTA
Modularidade	Modularidade		

©International Software Testing Qualifications Board

ISO/IEC 25010:2023 (atual)	ISO/IEC 25010:2011 (anterior)	Notas	ISTQB® syllabi
Reutilização	Reutilização		
Capacidade de análise	Capacidade de análise		
Modificabilidade	Modificabilidade		
Testabilidade	Testabilidade		
Flexibilidade	**Portabilidade**	**Renomeado**	TTA, **TA**, PT
Adaptabilidade	Adaptabilidade		**TA**, TTA
Escalabilidade		Novo	PT
Instalabilidade	Instalabilidade		**TA**, TTA
Substituibilidade	Substituibilidade		TTA
Segurança		**Novo**	AuT
Restrição operacional		Novo	
Identificação de riscos		Novo	
Segurança contra falhas		Novo	
Aviso de perigo		Novo	
Integração segura		Novo	

©International Software Testing Qualifications Board

13 Apêndice G – Marcas

ISTQB® é marca registrada do International Software Testing Qualifications Board.

BSTQB® -e marca registrada do Brazilian Software Testing Qualifications Board.

UML® é marca registrada do Object Management Group (OMG).

BPMN™ é marca registrada do Object Management Group (OMG).

©International Software Testing Qualifications Board