Certified Tester Advanced Level

Test Analyst Syllabus

CTAL-TA

versão 3.1.2

International Software Testing Qualifications Board

BSTQB

Versão traduzida, adaptada e disponibilizada na Língua Portuguesa pelo *Grupo de Traduções do BSTQB*

Direitos autorais

Histórico da Revisão

Versão	Data	Observações
2012 v2.0	19.10.2012	First version as a separate AL-TA Syllabus
2019 v3.0	19.10.2019	Major update with overall revision and scope reduction
V3.1.0	03.03.2021	Minor update with section 3.2.3 rewritten and various wording improvements
V3.1.1	15.05.2021	Errata: The copyright notice has been adapted to the current ISTQB® standards.
V3.1.2	31.01.2022	Errata: Minor formatting, grammar and wording defects fixed

Histórico da versão BSTQB

Versão	Data	Observações
1	14/06/2023	Padronização do layout com o ISTQB

Índice

Agradecimentos

Este documento foi produzido por uma equipe do *International Software Testing Qualifications Board Advanced Level Working Group*: Mette Bruhn-Pedersen (Working Group Chair); Matthias Hamburg (Product Owner); Wim Decoutere, István Forgács, Adam Roman, Jan Sabak, Marc-Florian Wendland (Authors).

A equipe de trabalho agradece a Paul Weymouth e Richard Green pela edição técnica, Gary Mogyorodi pelas verificações de conformidade com o Glossário, e aos Membros de Comissões pelos comentários de revisão relacionados com a versão publicada de 2019 do Syllabus e as alterações propostas.

As seguintes pessoas participaram da revisão e comentário deste syllabus:

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

A versão 2019 deste documento foi produzida pela equipe central do *International Software Testing Qualifications Board Advanced Level Working Group:* Graham Bath, Judy McKay, Mike Smith.

As seguintes pessoas participaram da revisão, comentário e votação da versão 2019 deste syllabus:

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	

A versão 2012 deste documento foi produzida pela equipe central do *International Software Testing Qualifications Board Advanced Level Sub Working Group – Advanced Test Analyst*: udy McKay (Chair), Mike Smith, Erik van Veenendaal.

Quando o Advanced Level Syllabus foi concluído o grupo de trabalho tinha os seguintes membros: Graham Bath, Rex Black, Maria Clara Choucair, Debra Friedenberg, Bernard Homès (Vice Chair), Paul Jorgensen, Judy McKay, Jamie Mitchell, Thomas Mueller, Klaus Olsen, Kenji Onishi, Meile Posthuma, Eric Riou du Cosquer, Jan Sabak, Hans Schaefer, Mike Smith (Chair), Geoff Thompson, Erik van Veenendaal, Tsuyoshi Yumoto.

As seguintes pessoas participaram da revisão, comentário e votação da versão 2012 deste syllabus:

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 voluntário do GT Traduções pelo empenho e esforço na tradução e revisão deste material: Eduardo Medeiros, George Fialkovitz, Irene Nagase, Osmar Higashi, Rogério Athaide, Stênio Viveiros.

Este documento foi formalmente divulgado pela Assembleia Geral do *ISTQB®* em 23 de fevereiro de 2021, e sua versão na Língua Portuguesa em 27 de maio de 2021.

0 Introdução

0.1 Objetivo deste documento

Este documento forma a base de conhecimento para a Certificação *ISTQB® CTAL-TA Test Analyst*. O *ISTQB®* fornece este syllabus da seguinte forma:

- Aos Conselhos Nacionais, para traduzirem ao seu idioma local e credenciar os provedores de treinamento. Os Conselhos Nacionais podem adaptar o syllabus às suas necessidades linguísticas específicas e modificar as referências para se adaptarem às suas publicações locais.
- Aos Conselhos de Exame, para derivar questões de exame em sua língua local com base nos objetivos de aprendizagem para cada módulo.
- Aos Provedores de Treinamento, para que produzam o material didático e determinem métodos apropriados de ensino.
- Aos Candidatos à Certificação, como uma fonte para se prepararem para o exame.
- À Comunidade Internacional de Software e à Engenharia de Sistemas, para avançar a profissão de software e testes de sistema, e servir como base para livros e artigos.

O *ISTQB®* permite que outras entidades usem esse syllabus para outros fins, desde que obtenham autorização prévia por escrito.

0.2 A certificação de nível avançado no teste de software

A certificação de nível avançado é composta por três syllabus distintos relacionados com as seguintes funções:

- Gerente de Teste

Analista de Teste

- Analista Técnico de Teste

O *ISTQB® Advanced Level Overview 2019* é um documento separado [ISTQB_AL_OVIEW] que inclui as seguintes informações:

- Resultados de negócio;
- Matriz de rastreabilidade entre os resultados de negócio e os objetivos de aprendizagem;
- Sumário;
- Relacionamento entre os syllabi.

0.3 Objetivos de aprendizagem examináveis e níveis cognitivos

Os objetivos de aprendizagem suportam os resultados de negócios e são usados para criar o exame para obter a Certificação *ISTQB® CTAL-TA Test Analyst*.

Os níveis cognitivos dos objetivos de aprendizagem nos níveis K2, K3 e K4 são mostrados no início de cada capítulo e são classificados da seguinte forma:

- K2: Entender
- K3: Aplicar
- K4: Analisar

As definições de todos os termos listados como conceitos-chave logo abaixo dos títulos dos capítulos devem ser lembradas (K1), mesmo que não explicitamente mencionadas nos objetivos de aprendizagem.

0.4 O exame de certificação

O exame para Analista de Teste do Nível Avançado será baseado nesse syllabus. As respostas às perguntas do exame podem exigir o uso de materiais baseados em mais de um capítulo desse syllabus. Todos os capítulos são examináveis, exceto a introdução e os apêndices. Normas, livros e outros syllabi do ISTQB® são incluídos como referências, mas seu conteúdo não é examinável além do que está resumido nesse documento.

O formato do exame é de múltipla escolha com 40 questões. Para passar no exame, o candidato deve obter pelo menos 65% de acerto do total de pontos (100 pontos).

O exame pode ser feito como parte de um curso de treinamento credenciado ou realizado independentemente (p. ex., em um centro de exames ou em um exame público). A conclusão de um curso de treinamento credenciado não é um pré-requisito para um exame.

0.5 Pré-requisito para o exame

Estar aprovado na certificação *ISTQB® CTFL Foundation Level* é pré-requisito para que o candidato faça um exame de certificação *ISTQB® CTAL-TA Test Analyst*.

0.6 Expectativas de experiência

Nenhum dos objetivos de aprendizagem para o Analista de Teste necessita de uma experiência prévia.

0.7 Credenciamento de cursos

Um Conselho Nacional do *ISTQB®* (como o BSTQB) pode credenciar provedores de treinamento cujo material de curso siga esse syllabus. Os provedores de treinamento devem obter as diretrizes de credenciamento do Conselho Nacional ou do órgão que realiza o credenciamento. Um curso credenciado é reconhecido como estando em conformidade com esse syllabus e é permitido ter um exame *ISTQB®* como parte do curso.

0.8 Nível de detalhe do syllabus

O nível de detalhe nesse syllabus permite cursos e exames consistentes internacionalmente. Para alcançar este objetivo, o syllabus consiste em:

- Objetivos gerais de instrução descrevendo a intenção do Analista de Teste;
- Uma lista de termos que os alunos devem ser capazes de lembrar;
- Os objetivos de aprendizagem para cada área de conhecimento, descrevendo o resultado da aprendizagem cognitiva a alcançar;
- Uma descrição dos conceitos-chave, incluindo referências a fontes como normas e legislações.

O conteúdo do syllabus não é uma descrição de toda a área de conhecimento; ele reflete o nível de detalhe a ser coberto nos cursos de treinamento do Nível Avançado. Ele se concentra no material que pode ser aplicado a todos os projetos de software, incluindo os projetos ágeis. O conteúdo do syllabus não contém nenhum objetivo de aprendizagem específico relacionado a nenhum ciclo de vida de desenvolvimento de software em particular (SDLC), mas discute como esses conceitos se aplicam em projetos ágeis, tipos de ciclos de vida iterativos e incrementais, e em ciclos de vida sequenciais.

0.9 Como este syllabus é organizado

Há seis capítulos com conteúdo examinável. No título de cada capítulo é especificado entre colchetes o tempo mínimo de estudo para cada capítulo; o tempo não é fornecido para subcapítulos. Para cursos de treinamento certificados, o syllabus requer um mínimo de 20 horas e 30 minutos de instrução, distribuídos pelos seis capítulos da seguinte forma:

- Capítulo 1: As tarefas do Analista de Teste no processo de teste (150 minutos)
- Capítulo 2: As tarefas do Analista de Teste em testes baseados em risco (60 minutos)
- Capítulo 3: Técnicas de teste (630 minutos)
- Capítulo 4: Teste das características de qualidade do software (180 minutos)
- Capítulo 5: Revisões (120 minutos)
- Capítulo 6: Ferramentas de teste e automação (90 minutos)

1 Tarefas do Analista de Teste no processo de teste [150 min]

Palavras-chave

análise de teste, base de teste, caso de teste de alto nível, caso de teste de baixo nível, condição de teste, critérios de saída, dados de teste, execução de teste, implementação de teste, modelagem de teste, procedimento de teste, programação de execução de teste, suíte de teste, teste.

Objetivos de Aprendizagem

1.1 Introdução

Sem objetivos de aprendizagem

1.2 Teste no ciclo de vida de desenvolvimento de software

TA-1.2.1 (K2) Explicar como e por que o tempo e o nível de envolvimento do Analista de Teste variam ao trabalhar com diferentes modelos de ciclo de vida de desenvolvimento de software

1.3 Análise de teste

TA-1.3.1 (K2) Resumir as tarefas do Analista de Teste ao realizar as atividades de análise

1.4 Modelagem de teste

TA-1.4.1 (K2) Explicar por que as condições do teste devem ser entendidas pelos stakeholders

TA-1.4.2 (K4) Para um determinado cenário de projeto, selecionar o nível de modelagem apropriada para casos de teste (nível superior ou baixo)

TA-1.4.3 (K2) Explicar os problemas a serem considerados na modelagem do caso de teste

1.5 Implementação do teste

TA-1.5.1 (K2) Resumir as tarefas do Analista de Teste ao realizar as atividades de implementação de teste

1.6 Execução do teste

TA-1.6.1 (K2) Resumir as tarefas do Analista de Teste ao conduzir atividades de execução de teste

1.1 Introdução

No syllabus do *ISTQB® CTFL Foundation*, o processo de teste é descrito com as seguintes atividades:

- Planejamento de teste
- Monitoramento e controle de teste
- Análise de teste
- Modelagem de teste
- Implementação de teste
- Execução de teste
- Conclusão do teste

Nesse syllabus, as atividades que têm particular relevância para o Analista de Teste são consideradas com mais profundidade. Isso fornece um refinamento adicional ao processo de teste para melhor atender aos diferentes modelos de ciclo de vida de desenvolvimento de software (SDLC).

Determinar os testes apropriados, modelá-los, implementá-los e depois executá-los são as principais áreas de atuação do Analista de Teste. Embora seja importante entender as outras etapas desse processo, a maioria do trabalho do Analista de Teste geralmente se concentra nas seguintes atividades:

- Análise de teste
- Modelagem de teste
- Implementação de teste
- Execução de teste

As outras atividades do processo de teste são descritas adequadamente no nível da Fundamental e não necessitam de mais aprofundamento nesse nível.

1.2 Teste no ciclo de vida de desenvolvimento de software

O SDLC geral deve ser considerado ao definir uma estratégia de teste. O momento do envolvimento do Analista de Teste é diferente para os vários SDLCs. O grau de envolvimento, o tempo necessário, as informações disponíveis e as expectativas também podem variar bastante. O Analista de Teste deve estar ciente dos tipos de informações a serem fornecidas a outras funções organizacionais relacionadas, como:

- *Engenharia e gerenciamento de requisitos*: feedback das revisões de requisitos;
- *Gerência de projetos*: entrada do cronograma;
- *Gerência de configuração e mudança*: resultado de teste de verificação de compilação, informação de controle de versão;
- *Desenvolvimento de software*: notificações de defeitos encontrados;
- *Manutenção de software*: relatórios sobre defeitos, eficiência na remoção de defeitos, e testes de confirmação;
- *Suporte técnico*: documentação precisa para soluções alternativas e problemas conhecidos;
- *Produção de documentação técnica*: (p. ex., especificações de modelagem do banco de dados, documentação do ambiente de teste) entrada para esses documentos, bem como sua revisão técnica.

As atividades de teste devem estar alinhadas com o SDLC escolhido, cuja natureza pode ser sequencial, iterativa, incremental ou híbrida. Por exemplo, no modelo V sequencial, o processo de teste aplicado ao nível de teste do sistema pode alinhar-se da seguinte forma:

- O planejamento de teste do sistema ocorre simultaneamente com o planejamento do projeto, e o monitoramento e controle do teste continuam até a conclusão do teste. Isso influenciará as entradas do cronograma fornecidas pelo Analista de Teste para fins de gerenciamento de projetos;
- A análise e a modelagem dos testes do sistema estão alinhadas com documentos como a especificação de requisitos do sistema, especificação de modelagem do sistema, e da arquitetura (alto nível) e especificação de modelagem do componente (baixo nível);
- A implementação do ambiente de teste do sistema pode começar durante o desenho do sistema, embora a maior parte dele normalmente ocorra simultaneamente com o teste de codificação e componente, com o trabalho nas atividades de implementação de teste do sistema, estendendo-se frequentemente até poucos dias antes do início da execução do teste;
- A execução de testes do sistema começa quando todos os critérios de entrada são atendidos ou, se necessário, dispensados, o que significa normalmente que pelo menos os testes de componentes e, muitas vezes, também os testes de integração de componentes atendam aos critérios de saída. A execução do teste do sistema continua até que os critérios de saída do teste do sistema sejam atendidos;
- As atividades de conclusão do teste do sistema ocorrem depois que os critérios de saída do teste do sistema são atendidos.

Os modelos iterativos e incrementais podem não seguir a mesma ordem de atividades e podem excluir algumas atividades. Por exemplo, um modelo iterativo pode utilizar um conjunto reduzido de atividades de teste para cada iteração. Análise de teste, projeto, implementação e execução podem ser conduzidos para cada iteração, enquanto o planejamento de alto nível é feito no início do projeto, e as tarefas de conclusão são feitas no final.

No desenvolvimento de software Ágil, é comum usar um processo menos formalizado e uma relação de trabalho muito mais próxima com os parceiros do projeto que permite que as mudanças ocorram mais facilmente dentro do projeto. Pode não haver um papel bem definido de Analista de Testes. Há uma documentação de teste menos abrangente e a comunicação é mais curta e mais frequente.

O desenvolvimento de software Ágil envolve testes desde o início. Começa no início do desenvolvimento do produto, à medida que os desenvolvedores realizam seu trabalho inicial de arquitetura e desenho. As revisões podem não ser formalizadas, mas são contínuas à medida que o software evolui. Espera-se que haja envolvimento durante todo o projeto e que as tarefas de Analista de Testes sejam feitas pela equipe.

Os modelos iterativos e incrementais vão desde o desenvolvimento de software Ágil, onde há uma expectativa de mudança à medida que os requisitos do cliente evoluem, até modelos híbridos, por exemplo, desenvolvimento iterativo/incremental combinado com uma abordagem de modelo em V. Em tais modelos híbridos, os analistas de teste devem ser envolvidos nos aspectos de planejamento e projeto das atividades sequenciais, e então passar para um papel mais interativo durante as atividades iterativas/incrementais.

Seja qual for o SDLC utilizado, os analistas de teste precisam entender as expectativas de envolvimento, bem como o momento de tal envolvimento. Os analistas de teste fornecem uma contribuição efetiva à qualidade do software, ajustando suas atividades e seu momento de envolvimento ao SDLC específico, em vez de se ater a um modelo de papel pré-definido.

1.3 Análise de Teste

No planejamento do teste é definido o escopo do projeto de teste. Durante a análise do teste, os analistas de teste usam esta definição de escopo para:

- Analisar a base de teste;
- Identificar vários tipos de defeitos na base de teste;
- Identificar e priorizar as condições e recursos de teste a serem testados;
- Capturar a rastreabilidade bidirecional entre cada elemento da base de teste e as condições de teste associadas;
- Executar as tarefas associadas a testes baseado em risco (consulte o capítulo 2)

Para que os Analistas de Teste prossigam efetivamente com a análise de teste, os seguintes critérios de entrada devem ser atendidos:

- Ter um conjunto de informações (p. ex., requisitos, histórias de usuário) descrevendo o objeto de teste que pode servir como uma base para o teste (consulte [ISTQB_FL] capítulo 1.4.2 para obter uma lista de outras fontes possíveis).
- Revisar essa base para teste obtendo-se resultados razoáveis, atualizando-a conforme necessário após a revisão. Observe que, para definir casos de teste de alto nível (consulte o capítulo 1.4.1), talvez a base para o teste ainda não precise ser totalmente definida. Em um projeto Ágil, esse ciclo de revisão será iterativo, pois as histórias de usuário são refinadas no início de cada iteração.
- Ter um orçamento e cronograma aprovados disponíveis para realizar as tarefas de teste restantes para o objeto de teste.

As condições de teste são geralmente identificadas pela análise da base de teste em conjunto com os objetivos do teste (conforme definido no planejamento do teste). Em algumas situações, em que a documentação pode ser antiga ou inexistente, as condições de teste podem ser identificadas através do debate com os stakeholders relevantes (p. ex., em workshops ou durante o planejamento da iteração). Em um projeto Ágil, os critérios de aceite definidos como parte das histórias de usuário são frequentemente usados como base para a modelagem do teste.

Embora as condições de teste sejam geralmente específicas para o item que está sendo testado, há algumas considerações básicas para o Analista de Teste.

- Geralmente, é aconselhável definir as condições de teste em diferentes níveis de detalhes. Inicialmente, são identificadas as condições de alto nível para definir metas gerais para o teste, como "*funcionalidade da tela x*". Posteriormente, as condições mais detalhadas são identificadas como a base de casos de teste específicos, como "*a tela x rejeita um número de conta com um dígito a menos do comprimento correto*". O uso desse tipo de abordagem hierárquica para definir as condições de teste pode ajudar a garantir que a cobertura seja suficiente para os itens de alto nível. Essa abordagem também permite que o Analista de

Teste comece a trabalhar na definição de condições de teste de alto nível para as histórias de usuários que ainda não foram refinadas.
- Se os riscos do produto tiverem sido definidos, as condições de teste necessárias para lidar com cada risco do produto devem ser identificadas e rastreadas até esse item de risco.

A aplicação de técnicas de teste (conforme identificadas na estratégia de teste ou no plano de teste) pode ser útil no processo de análise de teste e pode ser usada para apoiar os seguintes objetivos:

- Identificar as condições de teste;
- Reduzir a probabilidade de omitir condições importantes de teste;
- Definir com mais precisão as condições de teste;
- Depois que as condições de teste forem identificadas e refinadas, a revisão dessas condições com os stakeholders poderá ser realizada para garantir que os requisitos foram claramente entendidos e que os testes estejam alinhados com os objetivos do projeto.

Na conclusão das atividades de análise de teste para uma determinada área (p. ex., uma função específica), o Analista de Teste deve saber quais testes específicos devem ser modelados para essa área.

1.4 Modelagem de Teste

Ainda aderindo ao escopo determinado durante o planejamento do teste, o processo de teste continua enquanto o Analista de Teste modela os testes que serão implementados e executados. A modelagem do teste inclui as seguintes atividades:

- Determinar em quais áreas de teste os casos de teste de alto ou baixo nível são adequados;
- Determinar a(s) técnica(s) de teste que permitirão a cobertura necessária. As técnicas que podem ser usadas são estabelecidas durante o planejamento do teste;
- Usar técnicas de teste para projetar casos de teste e conjuntos de casos de teste que cobrem as condições de teste identificadas;
- Identificar os dados de teste necessários para suportar as condições e casos de teste;
- Projetar o ambiente de teste e identificar a infraestrutura necessária, incluindo ferramentas;
- Capturar a rastreabilidade bidirecional (p. ex., entre a base de teste, condições e casos de teste)

Os critérios de priorização identificados durante a análise de risco e o planejamento do teste devem ser aplicados ao longo do processo, desde a análise e modelagem até a implementação e execução.

Dependendo dos tipos de testes que estão sendo projetados, um dos critérios de entrada para a modelagem de teste pode ser a disponibilidade de ferramentas que serão usadas durante o trabalho de modelagem.

Durante a modelagem do teste, o Analista de Teste deve considerar pelo menos:

- Alguns itens de teste são melhor atendidos definindo apenas as condições de teste, ao invés de avançar mais na definição dos scripts de teste, que fornecem a sequência de instruções necessárias para executar um teste. Nesse caso, as condições de teste devem ser definidas para serem usadas como um guia para testes sem script;
- Os critérios de aprovação/reprovação devem ser claramente identificados;

- Os testes devem ser modelados para serem compreensíveis por outros testadores, não apenas pelo autor. Se o autor não for a pessoa que executa o teste, outros testadores precisarão ler e entender os testes especificados anteriormente para entender os objetivos e a importância relativa do teste;
- Os testes também devem ser compreensíveis para outros stakeholders, como desenvolvedores que podem revisar os testes, e auditores que talvez precisem aprová-los;
- Os testes devem abranger todos os tipos de interação com o objeto de teste e não devem ser restritos às interações das pessoas por meio da interface visível ao usuário. Eles também podem incluir, por exemplo, interação com outros sistemas e eventos técnicos ou físicos. (consulte [IREB_CPRE] para mais detalhes).
- Os testes devem ser projetados para testar as interfaces entre os vários objetos de teste, bem como os comportamentos dos próprios objetos.
- O esforço do projeto de teste deve ser priorizado e equilibrado para se alinhar aos níveis de risco e valor do negócio.

1.4.1 Casos de teste de alto nível e baixo nível

Uma das tarefas do Analista de Teste é determinar o melhor nível de modelagem de casos de teste para uma determinada situação. Os casos de teste de alto nível e baixo nível são abordados no [ISTQB_FL]. Seguem algumas das vantagens e desvantagens em usá-las:

Casos de teste de baixo nível fornecem as seguintes vantagens:

- A equipe de teste inexperiente pode contar com informações detalhadas fornecidas no projeto. Os casos de teste de baixo nível fornecem todas as informações e procedimentos específicos necessários para o testador executar o caso de teste (incluindo quaisquer requisitos de dados) e verificar os resultados;
- Os testes podem ser executados novamente por diferentes testadores e devem alcançar os mesmos resultados;
- Podem ser revelados defeitos não óbvios na base de teste;
- Se necessário, o nível de detalhe permite uma verificação independente dos testes, como auditorias;
- O tempo gasto na implementação automatizada de caso de teste pode ser reduzido.

Os casos de teste de baixo nível têm as seguintes desvantagens:

- Podem exigir uma quantidade significativa de esforço, tanto para criação quanto na manutenção;
- Tendem a limitar a engenhosidade do testador durante a execução;
- Exigem que a base de teste seja bem definida;
- A rastreabilidade das condições de teste pode exigir mais esforço do que nos casos de teste de alto nível.

Casos de teste de alto nível fornecem as seguintes vantagens:

- Fornecem diretrizes para o que deve ser testado e permitem que o Analista de Teste varie os dados reais ou o procedimento seguido ao executar o teste;

- Podem fornecer uma melhor cobertura de risco do que os casos de teste de baixo nível, porque variam um pouco cada vez que são executados;
- Podem ser definidos no início do processo de requisitos;
- Fazem uso da experiência em teste do Analista de Teste e o objeto de teste quando o teste é executado;
- Podem ser definidos quando não é necessária documentação formal e detalhada;
- São mais adequados para reutilização em diferentes ciclos de teste quando diferentes dados de teste podem ser usados.

Os casos de teste de alto nível têm as seguintes desvantagens:

- São menos reprodutíveis, dificultando a verificação. Isso ocorre porque eles não têm uma descrição detalhada encontrada em casos de teste de baixo nível;
- Uma equipe de teste mais experiente pode ser necessária para executá-los.

Ao automatizar com base em casos de teste de alto nível, a falta de detalhes pode resultar na validação errada dos resultados ou na falta de itens que devem ser validados.

Os casos de teste de alto nível, podem ser usados para desenvolver casos de teste de baixo nível quando os requisitos se tornarem mais definidos e estáveis. Nesse caso, a criação do caso de teste é feita sequencialmente, passando de alto nível para baixo nível, com apenas os casos de teste de baixo nível sendo usados na execução.

1.4.2 Modelagem de casos de teste

Os casos de teste são modelados pela elaboração e aperfeiçoamento do passo a passo das condições de teste identificadas usando técnicas de teste (consulte o capítulo 3). Os casos de teste devem ser repetíveis, verificáveis e rastreáveis até a base de teste (p. ex., requisitos).

A modelagem do teste inclui a identificação do:

- Objetivo (ou seja, o objetivo observável e mensurável da execução do teste);
- Condições prévias, como requisitos do projeto ou do ambiente de teste e os planos de entrega, do estado do sistema antes da execução do teste etc.;
- Requisitos de dados de teste (dados de entrada para o caso de teste e dados que devem existir no sistema para que o caso de teste seja executado);
- Resultados esperados com critérios explícitos de aprovação/reprovação;
- Pós-condições, como dados afetados, estado do sistema após a execução do teste, disparadores (*triggers*) para processamento subsequente etc.

Um desafio específico pode ser a definição do resultado esperado de um teste. Computar isso manualmente geralmente é tedioso e propenso a erros, se possível, pode ser preferível encontrar ou criar um oráculo de teste automatizado. Na identificação do resultado esperado, os testadores estão preocupados não apenas com as saídas na tela, mas também com os dados e as pós-condições ambientais. Se a base de teste estiver claramente definida, identificar o resultado correto, teoricamente, deve ser simples. No entanto, a documentação da base de teste pode ser vaga, contraditória, sem cobertura de áreas-chave ou totalmente ausente. Nesses casos, o Analista de Teste deve ter experiência no assunto ou ter acesso a ela. Além disso, mesmo quando a base de teste é bem especificada, interações complexas de estímulos e respostas complexos podem

dificultar a definição dos resultados esperados; portanto, um oráculo de teste é essencial. Em um projeto Ágil, o oráculo de teste pode ser o dono do produto (product owner). A execução de casos de teste sem qualquer maneira de determinar a exatidão dos resultados pode ter um valor agregado ou benefício muito baixo, geralmente gerando relatórios de falhas inválidas ou da falsa confiança no sistema.

As atividades descritas acima podem ser aplicadas a todos os níveis de teste, embora a base de teste varie. Ao analisar e projetar testes, é importante lembrar o nível alvo para o teste, bem como o objetivo do teste. Isso ajuda a determinar o nível de detalhe necessário, bem como quaisquer ferramentas que possam ser necessárias (p. ex., drivers e simuladores no nível do teste do componente).

Durante o desenvolvimento de condições de teste e casos de teste, normalmente é criada uma certa quantidade de documentação, resultando em produtos de trabalho de teste. Na prática, a extensão em que os produtos de teste são documentados varia consideravelmente. Isso pode ser afetado por:

- Riscos do projeto (o que deve e não deve ser documentado);
- O valor agregado que a documentação traz para o projeto;
- Padrões a serem seguidos ou regulamentos a serem cumpridos;
- DLC ou abordagem usada (p. ex., uma abordagem Ágil visa a documentação "apenas o suficiente");
- O requisito de rastreabilidade a partir da base de teste por meio de análise e projeto de teste.

Dependendo do escopo do teste, a análise e a modelagem abordam as características de qualidade do(s) objeto(s) de teste. O padrão ISO-25010 [ISO25010] fornece uma referência útil. Podem ser aplicadas características adicionais ao se testar sistemas de hardware ou software.

Os processos de análise e modelagem podem ser aprimorados entrelaçando-os com revisões e análises estáticas. De fato, a realização da análise e da modelagem dos testes geralmente é uma forma de teste estático, pois podem ser encontrados problemas nos documentos da base de teste durante o processo. A análise e a modelagem, com base na especificação de requisitos, são uma excelente maneira de se preparar para uma reunião de revisão de requisitos. A leitura dos requisitos para usá-los na criação de testes requer a compreensão do requisito e a capacidade de determinar uma maneira de avaliar o seu cumprimento. Essa atividade geralmente descobre requisitos que não são claros, que não podem ser testados ou que não possuem critérios definidos de aceite. Da mesma forma, os produtos de trabalho de teste, como os casos de teste, as análises de risco e os planos de teste, podem ser submetidos a revisões.

Durante a modelagem de teste, os requisitos detalhados da infraestrutura de teste podem ser definidos, embora na prática eles não possam ser finalizados até a implementação do teste. Deve-se lembrar que a infraestrutura de teste inclui mais do que os objetos de teste e o *testware*. Por exemplo, os requisitos de infraestrutura podem incluir salas, equipamentos, pessoal, software, ferramentas, periféricos, equipamentos de comunicação, autorizações de usuário e todos os outros itens necessários para a execução dos testes.

Os critérios de saída para análise e modelagem variarão dependendo dos parâmetros do projeto, mas todos os itens discutidos nesses dois capítulos devem ser considerados para inclusão nos critérios definidos de saída. É importante que os critérios de saída sejam mensuráveis e que todas as informações e preparações necessárias para as etapas subsequentes tenham sido fornecidas.

1.5 Implementação do teste

A implementação do teste prepara o testware necessário para a execução do teste, com base na análise e na modelagem de teste. Inclui as seguintes atividades:

- Desenvolver procedimentos de teste e, potencialmente, criar scripts de teste automatizados.
- Organizar procedimentos de teste e scripts de teste automatizados (se houver) em conjuntos de teste a serem executados em uma execução de teste específica.
- Consultar o Gerente de Testes para priorizar os casos de teste e conjuntos de teste a serem executados.
- Criar um cronograma de execução de teste, incluindo alocação de recursos, para permitir que a execução de teste comece (ver [ISTQB_FL_SYL] capítulo 5.2.4).
- Finalizando a preparação dos dados de teste e ambientes de teste.
- Atualizando a rastreabilidade entre a base de teste e o testware, tais como condições de teste, casos de teste, procedimentos de teste, scripts de teste e conjuntos de teste.

Durante a implementação do teste, os Analistas de Teste identificam uma ordem de execução eficiente dos casos de teste e criam procedimentos de teste. A definição dos procedimentos de teste requer a identificação cuidadosa das restrições e dependências que podem influenciar a sequência de execução do teste. Os procedimentos de teste documentam quaisquer pré-condições iniciais (p. ex., carregamento de dados de teste de um repositório de dados) e quaisquer atividades após a execução (p. ex., reinicialização do status do sistema).

Analistas de teste identificam procedimentos de teste e scripts de teste automatizados que podem ser agrupados (p. ex., todos eles se relacionam com o teste de um determinado processo comercial de alto nível), e os organizam em conjuntos de teste. Isto permite que os casos de teste relacionados sejam executados em conjunto.

Os analistas de teste organizam as suítes de teste dentro de um cronograma de execução de teste de forma que resulte em uma execução eficiente do teste. Se uma estratégia de teste baseada em risco estiver sendo usada, o nível de risco será a principal consideração na determinação da ordem de execução dos casos de teste. Pode haver outros fatores que determinam a ordem de execução dos casos de teste, tais como a disponibilidade das pessoas certas, equipamentos, dados e a funcionalidade a ser testada.

Não é incomum que o código seja lançado nas seções e o esforço de teste ser coordenado com a sequência na qual o software se torna disponível para teste. Particularmente nos modelos de ciclo de vida iterativo e incremental, é importante que o Analista de Teste coordene com a equipe de desenvolvimento para garantir que o software seja lançado para teste em uma ordem testável.

O nível de detalhe e a complexidade associada ao trabalho realizado durante a implementação do teste podem ser influenciados pelos detalhes dos casos e condições de teste. Em alguns casos, aplicam-se regras regulatórias e os produtos de teste devem fornecer evidências de conformidade com os padrões aplicáveis, como o padrão dos Estados Unidos DO-178C (na Europa, ED 12C). [RTCA DO-178C / ED-12C].

Conforme especificado acima, os dados de teste são necessários para a maioria dos testes e, em alguns casos, esses conjuntos de dados podem ser bastante grandes. Durante a implementação, os

Analistas de Teste criam dados de entrada e do ambiente para carregar nos bancos de dados e em outros repositórios. Esses dados devem ser "adequados à finalidade" para permitir a detecção de defeitos. Os Analistas de Teste também podem criar dados para serem usados com testes de controlados por dados e por palavras-chave (consulte o capítulo 6.2), bem como para testes manuais.

A implementação do teste também se preocupa com o(s) ambiente(s) de teste. Durante esta atividade, o(s) ambiente(s) deve(m) ser totalmente montado(s) e verificado(s) antes da execução do teste. Um ambiente de teste "adequado à finalidade" é essencial, ou seja, o ambiente de teste deve ser capaz de permitir a exposição dos defeitos presentes durante os testes controlados, operar normalmente quando não houver falhas e replicar adequadamente, se necessário, o ambiente de produção ou de usuário final para níveis de teste mais altos. Mudanças no ambiente de teste podem ser necessárias durante a execução do teste, dependendo de mudanças imprevistas, resultados do teste ou outras considerações. Se ocorrerem mudanças no ambiente durante a execução, é importante avaliar o impacto das mudanças nos testes que já foram executados.

Durante a implementação do teste, o Analista de Teste deve verificar se os responsáveis pela criação e manutenção do ambiente de teste são conhecidos e disponíveis, e se todas as ferramentas de suporte ao teste e testware e processos associados estão prontos para uso. Isso inclui o gerenciamento de configuração, gerenciamento de defeitos e gestão e registro dos testes. Além disso, os Analistas de Teste devem verificar os procedimentos que coletam dados para avaliar o status atual com base nos critérios de saída e nos relatórios de resultados dos testes.

É aconselhável usar uma abordagem equilibrada para testar a implementação, conforme determinado durante o planejamento do teste. Por exemplo, as estratégias de teste analítico com base no risco geralmente são combinadas com estratégias de teste reativas. Nesse caso, uma porcentagem do esforço de implementação do teste é alocada para testes que não seguem scripts predeterminados (sem script).

O teste sem script não deve ser aleatório ou sem objetivo, pois pode ser imprevisível em termos de duração e cobertura, e terem baixo rendimento em encontrar defeitos. Em vez disso, deve ser conduzido em sessões controladas de tempo, cada uma dada direção inicial por um regulamento de teste, mas com a liberdade de se afastar dessas prescrições se forem descobertas oportunidades de teste potencialmente mais produtivas no decorrer da sessão. Ao longo dos anos, os Testadores desenvolveram uma variedade de técnicas de teste baseadas na experiência, como ataques [Whittaker03], suposição de erros [Myers11] e testes exploratórios [Whittaker09]. A análise, a modelagem e a implementação ainda ocorrem, mas principalmente durante a execução do teste.

Ao seguir essas estratégias de teste reativas, os resultados de cada teste influenciam a análise, a modelagem e a implementação dos testes subsequentes. Embora essas estratégias sejam leves e frequentemente eficazes para encontrar defeitos, existem algumas desvantagens, incluindo:

- É necessário que o Analista de Teste seja experiente;
- Pode ser difícil prever a duração do teste;
- A cobertura pode ser difícil de rastrear;
- A repetibilidade pode ser perdida sem uma boa documentação ou suporte de ferramenta.

1.6 Execução do Teste

A execução do teste é conduzida de acordo com o planejamento do teste e inclui (consultar [ISTQB_FL]):

- Execução de testes manuais, incluindo testes exploratórios;
- Execução de testes automatizados;
- Comparação dos resultados obtidos com os resultados esperados;
- Análise das anomalias para estabelecer suas prováveis causas;
- Documentação dos defeitos com base nas falhas observadas;
- Registro dos resultados reais da execução do teste;
- Atualização da rastreabilidade entre a base de teste e o testware para considerar os resultados do teste;
- Execução de testes de regressão.

As tarefas de execução de teste listadas acima podem ser conduzidas pelo Testador ou pelo Analista de Teste.

A seguir, são apresentadas as típicas tarefas adicionais que podem ser executadas pelo Analista de Teste:

- Reconhecimento de agrupamentos de defeitos que podem indicar a necessidade de mais testes de uma parte específica do objeto de teste;
- Dar sugestões para futuras sessões de testes exploratórios com base nos resultados obtidos dos testes exploratórios;
- Identificar novos riscos a partir de informações obtidas da execução do teste;
- Dar sugestões para melhorar qualquer um dos produtos de trabalho da atividade de implementação de teste (p. ex., melhorias nos procedimentos de teste).

2 Tarefas do Analista de Teste em testes baseados em risco [60 min]

Palavras-chave

identificação de risco, mitigação de risco, risco de produto, testes baseados em risco

Objetivos de aprendizagem

2.2 Tarefas do Analista de Teste em testes baseados em risco

TA-2.1.1 (K3) Para uma determinada situação, participar da identificação de riscos, avaliá-los e propor a mitigação apropriada.

2.1 Introdução

Os gerentes de teste geralmente têm a responsabilidade geral de estabelecer e gerenciar uma estratégia de teste baseada em risco. Geralmente, eles solicitam o envolvimento de o Analista de Teste para garantir que a abordagem baseada em risco seja implementada corretamente.

Os Analistas de Teste devem estar envolvidos ativamente nas tarefas de teste baseadas em risco:

- Identificação de riscos;
- Avaliação de risco;
- Mitigação de riscos.

Estas tarefas são executadas iterativamente ao longo do SDLC para lidar com riscos emergentes, avaliar e alterar as prioridades, e comunicar regularmente o status dos riscos (mais detalhes em [vanVeenendaal12] e [Black02]). No desenvolvimento de software Ágil, as três tarefas geralmente são combinadas em uma, chamada sessão de risco, com foco em uma iteração ou em uma liberação.

Os Analistas de Teste devem trabalhar dentro da estrutura de teste baseada em risco estabelecida para o projeto pelo Gerente de Teste. Eles devem contribuir com o conhecimento dos riscos do domínio do negócio inerentes ao projeto, como riscos relacionados à segurança funcional, preocupações comerciais e econômicas, fatores políticos, entre outros.

2.2 Identificação de Riscos

Ao recorrer à mais ampla amostra possível de stakeholders, é mais provável que o processo de identificação de riscos detecte o maior número possível de riscos significativos.

Os Analistas de Teste geralmente possuem conhecimento exclusivo sobre o domínio do negócio específico do sistema em teste (SUT). Isso significa que eles são particularmente adequados para as tarefas:

- Realizar entrevistas com especialistas e usuários do domínio;
- Realizar avaliações independentes;
- Usar os modelos de risco;
- Participar de workshops de risco;
- Participar de sessões com usuários atuais e potenciais para troca de ideias;
- Definir as listas de verificação de teste;
- Recorrer a experiências anteriores com sistemas ou projetos similares.

Em particular, os Analistas de Teste devem trabalhar em estreita colaboração com os usuários e outros especialistas no domínio (p. ex., Engenheiros de Requisitos, Analistas de Negócio) para determinar as áreas de risco de negócio que devem ser abordadas durante o teste. No desenvolvimento de software Ágil, esse relacionamento próximo com os stakeholders permite que a identificação de riscos seja realizada regularmente, como durante as reuniões de planejamento de iteração.

A amostra de riscos que podem ser identificadas em um projeto incluem:

- Problemas com correção funcional, por exemplo, cálculos incorretos;

- Problemas de usabilidade, por exemplo, atalhos insuficientes de teclado;
- Problemas de portabilidade, por exemplo, incapacidade de instalar um aplicativo em plataformas específicas.

2.3 Avaliação do Risco

Embora a identificação dos riscos tenha como objetivo identificar o maior número possível de riscos pertinentes, a avaliação é o estudo desses riscos identificados. Especificamente, categorizando cada risco e determinando seu nível de risco.

A determinação do nível de risco normalmente envolve a avaliação, para cada item de risco, da probabilidade de risco e do impacto do risco. A probabilidade de risco é geralmente interpretada como a probabilidade de que o problema potencial possa existir no sistema em teste e será observado quando o sistema estiver em produção. Os Analistas de Teste Técnico devem contribuir para encontrar e compreender a probabilidade potencial para cada item de risco, enquanto os Analistas de Teste contribuem para compreender o impacto comercial potencial do problema caso ele ocorra (no desenvolvimento de software Ágil esta distinção baseada no papel pode ser menos forte).

O impacto do risco é frequentemente interpretado como a gravidade do efeito sobre os usuários, clientes ou outras partes interessadas. Em outras palavras, surge do risco de negócio. Os Analistas de Teste devem contribuir para identificar e avaliar o potencial domínio do negócio ou o impacto do usuário para cada item de risco. Fatores que influenciam o risco de negócio incluem:

- Frequência de uso do recurso afetado;
- Perda de negócios;
- Danos financeiros;
- Perdas ou responsabilidades ecológicas ou sociais;
- Sanções legais civis ou criminais;
- Preocupações de segurança funcional;
- Multas, perda de licença;
- Falta de soluções razoáveis se as pessoas não puderem mais trabalhar;
- Visibilidade do recurso;
- Visibilidade de falha que leva a publicidade negativa e dano potencial à imagem.

Dadas as informações de risco disponíveis, os Analistas de Teste precisam estabelecer os níveis de risco do negócio de acordo com as diretrizes fornecidas por um Gerente de Teste. Estes poderiam ser classificados usando uma escala ordinal (real/numérica ou baixa/média/alta), ou cores de sinais de tráfego. Uma vez que a probabilidade de risco e o impacto do risco tenham sido atribuídos, os Gerentes de Teste usam esses valores para determinar o nível de risco para cada item de risco. Esse nível de risco é então usado para priorizar as atividades de mitigação de risco. [vanVeenendaal12].

2.4 Mitigação de riscos

Durante o projeto, os Analistas de Teste devem procurar:

- Reduzir o risco do produto projetando casos de teste eficazes que demonstrem inequivocamente se os testes são aprovados ou reprovados e participando de revisões de produtos de trabalho de software, como requisitos, projetos e documentação do usuário;
- Implementar as atividades apropriadas de mitigação de risco identificadas na estratégia e no plano de teste (p. ex., testar um processo de negócio com risco particularmente alto usando técnicas específicas de teste);
- Reavaliar os riscos conhecidos com base em informações adicionais coletadas à medida que o projeto se desenrola, ajustando a probabilidade de risco, o impacto do risco ou ambos, conforme apropriado;
- Identificar os novos riscos a partir de informações obtidas durante o teste.

Quando se fala em risco de produto, o teste faz uma contribuição essencial para atenuar esses riscos. Ao encontrar defeitos, os testadores reduzem o risco, fornecendo conhecimento dos defeitos e oportunidades para lidar com eles antes da liberação. Se os testadores não encontrarem defeitos, o teste reduzirá o risco, fornecendo evidências de que, sob certas condições (isto é, as condições testadas), o sistema opera corretamente. Os Analistas de Teste ajudam a determinar as opções de mitigação de risco, investigando oportunidades para reunir dados precisos de teste, criando e testando cenários realistas do usuário e conduzindo ou supervisionando estudos de usabilidade, entre outros.

2.4.1 Priorizando os testes

O nível de risco também é usado para priorizar testes. Um Analista de Teste pode determinar que há um alto risco na área de precisão transacional em um sistema contábil. Como resultado, para reduzir o risco, o testador pode trabalhar com outros especialistas no domínio do negócio para reunir uma amostragem forte de dados que podem ser processados e verificados quanto à precisão. Da mesma forma, o Analista de Teste pode determinar que os problemas de usabilidade representam um risco significativo para um novo objeto de teste. Em vez de esperar que um teste de aceite do usuário descubra quaisquer problemas, o Analista de Teste pode priorizar um teste de usabilidade inicial com base em um protótipo para ajudar a identificar e resolver problemas de desenho de usabilidade antes do teste de aceite do usuário. Essa priorização deve ser considerada o mais cedo possível nas etapas de planejamento, para que o cronograma possa acomodar os testes necessários no tempo necessário.

Em alguns casos, todos os testes de maior risco são executados antes de qualquer teste de menor risco, e os testes são executados em ordem estrita de risco (chamada “profundidade em primeiro lugar”); em outros casos, uma abordagem de amostragem é usada para selecionar uma amostra de testes em todas as áreas de riscos identificados, usando o nível de risco para ponderar a seleção e, ao mesmo tempo, garantir a cobertura de todos os riscos pelo menos uma vez (denominada “amplitude”).

Quer o teste com base no risco prossiga em profundidade ou amplitude, é possível que o tempo alocado para o teste possa ser consumido sem a execução de todos os testes. O teste baseado em risco permite que os testadores se reportem à gerência em termos do nível de risco restante neste momento, e permite que a gerência decida se deseja estender o teste ou transferir o risco restante para usuários, clientes, suporte técnico, *helpdesk*, ou para a equipe operacional.

2.4.2 Ajustando o teste para futuros ciclos de teste

A avaliação de riscos não é uma atividade única realizada antes do início da implementação do teste; é um processo contínuo. Cada ciclo de teste planejado para o futuro deve ser submetido a uma nova análise de risco para levar em consideração fatores como:

- Quaisquer riscos novos ou significativamente alterados do produto;
- Áreas instáveis ou propensas a falhas descobertas durante o teste;
- Riscos de defeitos corrigidos;
- Defeitos típicos encontrados durante o teste;
- Áreas subtestadas (baixa cobertura de requisitos);

3 Técnicas de teste [630 min]

Palavras-chave

análise de valor-limite, particionamento de equivalência, regulamento de teste, suposição de erros, taxonomia de defeitos, técnica de árvore de classificação, técnica de teste baseada em defeitos, técnica de teste baseado na experiência, técnica de teste caixa-preta, teste baseado em checklist, teste baseado na experiência, teste de caso de uso, teste de tabela de decisão, teste de transição de estado, teste em pares, teste exploratório.

Objetivos de aprendizagem

3.1 Introdução

Sem objetivos de aprendizagem

3.2 Técnicas de teste caixa-preta

TA-3.2.1 (K4) Analisar um dado item(s) de especificação e casos de teste de projeto, aplicando a divisão de equivalência.

TA-3.2.2 (K4) Analisar um item da especificação e modelar os casos de teste aplicando a análise de valor de limite.

TA-3.2.3 (K4) Analisar um item da especificação e modelar os casos de teste aplicando o teste de tabela de decisão.

TA-3.2.4 (K4) Analisar um item da especificação e modelar os casos de teste aplicando o teste de transição de estado.

TA-3.2.5 (K2) Explicar como os diagramas de árvore de classificação suportam as técnicas de teste.

TA-3.2.6 (K4) Analisar um item da especificação e modelar os casos de teste aplicando o teste em pares.

TA-3.2.7 (K4) Analisar um item da especificação e modelar os casos de teste aplicando o teste de caso de uso.

TA-3.2.8 (K4) Analisar um sistema ou sua especificação de requisitos para determinar os prováveis tipos de defeitos a serem encontrados e selecione as técnicas apropriadas de teste caixa-preta.

3.3 Técnicas de teste baseadas na experiência

TA-3.3.1 (K2) Explicar os princípios das técnicas de teste baseadas na experiência e os benefícios e desvantagens em comparação com as técnicas de teste baseadas na caixa preta e nos defeitos.

TA-3.3.2 (K3) Identificar testes exploratórios a partir de um determinado cenário.

TA-3.3.3 (K2) Descrever a aplicação de técnicas de teste baseadas em defeitos e diferenciar seu uso das técnicas de teste caixa-preta.

3.4 Aplicando as técnicas de teste mais apropriadas

TA-3.4.1 (K4) Para uma determinada situação do projeto, determinar quais técnicas de teste baseadas na experiência ou caixa-preta devem ser aplicadas para alcançar os objetivos especificados.

3.1 Introdução

As técnicas de teste consideradas neste capítulo estão divididas nas categorias:

- Caixa-preta
- Baseado na experiência

Essas técnicas são complementares e podem ser usadas conforme apropriado para qualquer atividade de teste, independentemente de qual nível de teste esteja sendo executado.

Observe que ambas as categorias de técnicas podem ser usadas para testar características de qualidade funcionais e não funcionais. O teste das características é discutido no próximo capítulo.

As técnicas de teste discutidas nesses capítulos podem se concentrar principalmente na determinação de dados ideais de teste (p. ex., de partições de equivalência) ou na derivação de procedimentos de teste (p. ex., de modelos de estado). É comum combinar técnicas para criar casos de teste completos.

3.2 Técnicas de teste caixa-preta

As técnicas de teste caixa-preta são introduzidas no syllabus ISTQB® CTFL Foundation Level [ISTQB_FL].

Os recursos comuns das técnicas de teste caixa-preta incluem:

- Criação de modelos durante o projeto do teste de acordo com a técnica de teste, por exemplo: diagramas de transição de estado e tabelas de decisão;
- Derivação sistemática das condições de teste desses modelos.

As técnicas de teste geralmente fornecem critérios de cobertura, que podem ser usados para medir as atividades de modelagem e execução do teste. O preenchimento completo dos critérios de cobertura não significa que todo o conjunto de testes esteja completo, mas o modelo não sugere mais testes adicionais para aumentar a cobertura com base nessa técnica.

O teste caixa-preta geralmente é baseado em alguma forma de documentação de especificação, como uma especificação de requisitos do sistema ou histórias de usuários. Como a documentação de especificação deve descrever o comportamento do sistema, particularmente na área de funcionalidade, os testes derivados dos requisitos muitas vezes fazem parte dos testes de comportamento do sistema. Em alguns casos, pode não haver documentação de especificação, mas existem requisitos implícitos, como a substituição da adequação funcional de um sistema antigo.

Existem várias técnicas de teste caixa-preta. Essas técnicas têm como alvo diferentes tipos de software e cenários. Os capítulos seguintes mostram a aplicabilidade de cada técnica, algumas limitações e dificuldades que o Analista de Teste pode enfrentar, o método pelo qual a cobertura é medida e os tipos de defeitos visados.

Mais detalhes em: [ISO29119-4], [Bath14], [Beizer95], [Black07], [Black09], [Copeland04], [Craig02], [Forgács19], [Koomen06] e [Myers11].

3.2.1 Particionamento de equivalência

O particionamento de equivalência é uma técnica usada para reduzir o número de casos de teste necessários para testar efetivamente o manuseio de entradas, saídas, valores internos e valores relacionados ao tempo. O particionamento é usado para criar partições de equivalência (geralmente chamadas classes de equivalência) a partir de conjuntos de valores que precisam ser processados da mesma maneira. Ao selecionar um valor representativo de uma partição, a cobertura para todos os itens na mesma partição é assumida.

Normalmente, vários parâmetros determinam o comportamento do objeto de teste. Ao combinar as partições de equivalência de diferentes parâmetros para os casos de teste, várias técnicas podem ser aplicadas

Aplicabilidade

Esta técnica é aplicável em qualquer nível de teste e é apropriada quando se espera que todos os membros de um conjunto de valores a serem testados sejam tratados da mesma maneira e quando os conjuntos de valores utilizados pela aplicação não interagem. Uma partição de equivalência pode ser qualquer conjunto de valores não vazio, por exemplo: ordenado, não ordenado, discreto, contínuo, infinito, finito, ou mesmo um único botão. A seleção de conjuntos de valores é aplicável a partições válidas e inválidas (ou seja, partições contendo valores que devem ser considerados inválidos para o software em teste).

O particionamento de equivalência é mais forte quando usado em combinação com a análise de valor limite, que expande os valores de teste para incluir aqueles nos limites das partições. Usando valores de partições válidas, essa é uma técnica comumente usada para *smoke teste*, uma nova compilação ou uma nova versão, pois determina rapidamente se a funcionalidade básica está funcionando.

Limitações e Dificuldades

Se a suposição estiver incorreta e os valores na partição não forem tratados exatamente da mesma maneira, essa técnica poderá apresentar defeitos. Também é importante selecionar as partições cuidadosamente. Por exemplo, um campo de entrada que aceita números positivos e negativos pode ser mais bem testado como duas partições válidas, uma para os números positivos e outra para os números negativos, devido à probabilidade de manipulação diferente. Dependendo se o zero é ou não permitido, isso também pode se tornar outra partição. É importante para o Analista de Teste entender o processamento subjacente para determinar a melhor partição dos valores. Isso pode exigir apoio na compreensão do desenho do código.

O Analista de Testes também deve levar em conta possíveis dependências entre partições de equivalência de diferentes parâmetros. Por exemplo, em um sistema de reserva de voo, o parâmetro "adulto acompanhante" só pode ser usado em combinação com a classe de idade "criança".

Cobertura

A cobertura é determinada considerando o número de partições para as quais um valor foi testado dividido pelo número de partições identificadas. A cobertura da partição de equivalência é então declarada como uma porcentagem. O uso de vários valores para uma única partição não aumenta a porcentagem de cobertura.

Se o comportamento do objeto de teste depender de um único parâmetro, cada partição de equivalência, seja válida ou inválida, deve ser coberta pelo menos uma vez.

No caso de mais de um parâmetro, o Analista de Teste deve selecionar um tipo de cobertura simples ou combinatória, dependendo do risco [Offutt16]. Portanto, é essencial diferenciar entre combinações contendo apenas partições válidas e combinações contendo uma ou mais partições inválidas. Quanto às combinações com apenas partições de equivalência válidas, o requisito mínimo é uma cobertura simples de todas as partições válidas sobre todos os parâmetros. O número mínimo de casos de teste necessários em tal conjunto de teste é igual ao maior número de partições válidas de um parâmetro, assumindo que os parâmetros sejam independentes uns dos outros. Tipos de cobertura mais completos relacionados às técnicas combinatórias incluem a cobertura em pares (ver capítulo 3.2.6), ou a cobertura completa de qualquer combinação de partições válidas. As partições de equivalência inválida devem ser testadas pelo menos individualmente, ou seja, em combinação com partições válidas para os outros parâmetros, a fim de evitar o mascaramento de defeitos. Assim, cada partição inválida contribui com um caso de teste para o conjunto de teste para uma cobertura simples. Em caso de alto risco, outras combinações podem ser adicionadas ao conjunto de teste, por exemplo, consistindo apenas de partições inválidas, ou de pares de partições inválidas.

Tipos de Defeitos

Um Analista de Teste usa essa técnica para encontrar defeitos no tratamento de vários valores de dados.

3.2.2 Análise de valor limite

A análise de valor limite (BVA) é usada para testar o manuseio adequado dos valores existentes nos limites das partições de equivalência ordenadas. Duas abordagens para BVA são de uso comum: teste de limite de dois valores ou teste de limite de três valores. Com o teste de limite de dois valores, são usados o valor limite (no limite) e um valor que está fora do limite (pela menor precisão possível, com base na precisão necessária). Por exemplo, para valores em uma moeda com duas casas decimais, se a partição incluísse os valores de 1 a 10, os valores de teste de dois valores para o limite superior seriam 10 e 10,01. Os valores de teste do limite inferior seriam 1 e 0,99. Os limites são definidos pelos valores máximo e mínimo na partição de equivalência definida.

Para testes de limite de três valores, são utilizados os valores antes, dentro e fora do limite. No exemplo anterior, os testes de limite superior incluiriam 9,99, 10 e 10,01. Os testes de limite inferior incluiriam 0,99, 1 e 1,01. A decisão quanto ao uso de testes de limite de dois ou três valores deve ser baseada no risco associado ao item a ser testado, com a abordagem de limite de três valores sendo usada para os itens de maior risco.

Aplicabilidade

Essa técnica é aplicável em qualquer nível de teste e é apropriada quando existem partições de equivalência ordenada. Por esse motivo, essa técnica é frequentemente conduzida em conjunto com a técnica de partição de equivalência. As partições de equivalência ordenadas são necessárias devido ao conceito de estar dentro e fora dos limites. Por exemplo, uma gama de números é uma partição ordenada. Uma partição que consiste em algumas cadeias de texto também pode ser ordenada, por exemplo, por sua ordem léxica, mas se o pedido não for relevante do ponto de vista comercial, então

os valores-limite não devem estar em foco. Além das faixas de números, as partições para as quais a análise de valores-limite pode ser aplicada incluem:

- Atributos numéricos de variáveis não numéricas (p. ex., comprimento);
- O número de ciclos de execução de loop, incluindo loops em diagramas de transição de estado;
- O número de elementos de iteração em estruturas de dados armazenados, tais como matrizes;
- O tamanho dos objetos físicos, por exemplo, memória;
- A duração das atividades.

Limitações e Dificuldades

Como a precisão dessa técnica depende da identificação precisa das partições de equivalência para identificar corretamente os limites, ela está sujeita às mesmas limitações e dificuldades que a partição de equivalência. O Analista de Teste também deve estar ciente da precisão nos valores válidos e inválidos para poder determinar com precisão os valores a serem testados. Somente as partições ordenadas podem ser usadas para análise de valor limite, mas isso não se limita a uma variedade de entradas válidas. Por exemplo, ao testar o número de células suportadas por uma planilha, há uma partição que contém o número de células até e incluindo o número máximo de células permitidas (o limite) e outra partição que começa com uma célula acima do valor máximo (acima de o limite).

Cobertura

A cobertura é determinada considerando o número de condições de limite testadas dividida pelo número de condições de limite identificadas (usando o método de dois valores ou três valores). A cobertura é declarada como uma porcentagem.

Tipos de Defeitos

A análise do valor do limite localiza de maneira confiável o deslocamento ou a omissão de limites, podendo encontrar casos extras de limites. Essa técnica encontra defeitos no manuseio dos valores-limite, particularmente erros com lógica “menor que” e “maior que” (ou seja, deslocamento). Também pode ser usada para encontrar defeitos não funcionais, por exemplo, um sistema suporta 10.000 usuários simultâneos, mas não 10.001.

3.2.3 Teste de tabela de decisão

Uma tabela de decisão é uma representação tabular de um conjunto de condições e ações relacionadas, regras expressas que indicam qual ação deve ocorrer para qual conjunto de valores de condição [OMG-DMN]. Os analistas de teste podem utilizar tabelas de decisão para analisar as regras que se aplicam ao software em teste e testes de projeto para cobrir essas regras.

As tabelas de decisão nas quais as condições são Booleanas com valores simples "Verdadeiro" e "Falso" são chamadas de tabelas de decisão de entrada limitada. Um exemplo para tal condição é "Renda do usuário < 1000". As tabelas de decisão de entrada estendida permitem condições com valores múltiplos que podem representar elementos discretos ou conjuntos de elementos. Por exemplo, uma condição "Renda do usuário" pode assumir um dos três valores possíveis: "inferior a 1000", "entre 1000 e 2000" e "superior a 2000".

Ações simples tomam os valores booleanos "Verdadeiro" e "Falso" (p. ex., a ação "Desconto admitido = 20%" toma os valores "Verdadeiro" indicados por "X" se a ação deve ocorrer e "Falso" indicado por "-" se não ocorrer). Assim como com as condições, as ações também podem tomar valores de outros domínios. Por exemplo, uma ação "Desconto admitido" pode tomar um dos cinco valores possíveis: 0%, 10%, 20%, 35% e 50%.

Os testes da tabela de decisão começam com o desenho de tabelas de decisão baseadas na especificação. Regras contendo combinações inviáveis de valores de condição são excluídas ou marcadas como "inviáveis". Em seguida, o Analista de Testes deve rever as tabelas de decisão com as outras partes interessadas. O Analista de Teste deve garantir que as regras dentro da tabela sejam consistentes (ou seja, as regras não se sobrepõem), completas (ou seja, elas contêm uma regra para cada combinação viável de valores de condição) e corretas (ou seja, elas modelam o comportamento pretendido).

O princípio básico no teste da tabela de decisão é que as regras formam as condições de teste

Ao projetar um caso de teste para cobrir uma determinada regra, o Analista de Teste deve estar ciente de que os valores de entrada do caso de teste podem ser diferentes dos valores das condições da tabela de decisão. Por exemplo, o valor `Verdadeiro` da condição "renda anual > 100.000?" pode não ser diretamente aplicável, mas pode exigir trabalho do testador para definir a conta de um cliente com créditos maiores que 100.000 em um determinado ano fiscal. Da mesma forma, os resultados esperados do caso de teste podem ser diferentes das ações da tabela de decisão.

Após a tabela de decisão estar pronta, as regras precisam ser implementadas como casos de teste, selecionando valores de entrada de teste (e resultados esperados) que satisfaçam as condições e ações.

Tabelas de decisão em colapso

Ao tentar testar todas as combinações de entradas possíveis de acordo com as condições, as tabelas de decisão podem se tornar muito grandes. Uma tabela de decisão de entrada limitada completa com n condições tem 2n regras. Uma técnica de redução sistemática do número de combinações é chamada de teste de tabela de decisão colapsada [Mosley93]. Quando esta técnica é utilizada, um grupo de regras com o mesmo conjunto de ações pode ser reduzido (colapsado) a uma regra se, dentro deste grupo, algumas condições não forem relevantes para a ação, e todas as outras condições permanecerem inalteradas. Nesta regra resultante, os valores das condições irrelevantes são indicados como "não importa", geralmente marcados com um traço "-". Para condições com valores "não importa", o Analista de Testes pode especificar valores válidos arbitrários para a implementação do teste.

Outro caso para regras em colapso é quando um valor de condição não é aplicável em combinação com alguns outros valores de condição ou quando duas ou mais condições têm valores conflitantes. Por exemplo, em uma tabela de decisão para pagamentos com cartão, se a condição "cartão é válido" for falsa, a condição "código PIN está correto" não é aplicável.

As tabelas de decisão em colapso podem ter muito menos regras do que as tabelas de decisão completas, o que resulta em menos casos de teste e menos esforço. Se uma determinada regra tiver entradas "não importa", e apenas um caso de teste cobrir essa regra, apenas um dos vários valores possíveis da condição será testado para essa regra, de modo que um defeito envolvendo outros

valores pode permanecer sem ser detectado. Assim, para níveis de alto risco, em alinhamento com o Gerente de Testes, o Analista de Testes deve definir regras separadas para cada combinação viável dos valores de uma única condição, em vez de colapsar a tabela de decisão.

Aplicabilidade

Os testes de tabela de decisão são comumente aplicados aos níveis de integração, sistema e teste de aceitação. Também podem ser aplicáveis aos testes de componentes quando um componente é responsável por um conjunto de lógica de decisão. Esta técnica é particularmente útil quando o objeto de teste é especificado na forma de fluxogramas ou tabelas de regras comerciais.

As tabelas de decisão também são uma técnica de definição de requisitos e às vezes as especificações de requisitos já podem ser definidas neste formato. O Analista de Teste ainda deve participar da revisão das tabelas de decisão e analisá-las antes de iniciar o projeto do teste.

Limitações e dificuldades

Ao considerar combinações de condições, encontrar todas as condições de interação pode ser um desafio, particularmente quando as exigências não estão bem definidas ou não existem. Deve-se tomar cuidado ao selecionar as condições consideradas em uma tabela de decisão para que o número de combinações dessas condições permaneça manejável. Na pior das hipóteses, o número de regras crescerá exponencialmente.

Cobertura

O padrão comum de cobertura para esta técnica é cobrir cada regra da tabela de decisão com um caso de teste. A cobertura é medida como uma porcentagem do número de regras cobertas pelo conjunto de testes e o número total de regras viáveis.

A análise do valor-limite e a partição de equivalência podem ser combinadas com a técnica da tabela de decisão, especialmente no caso das tabelas de decisão de entrada prolongada. Se as condições contiverem partições de equivalência totalmente ordenadas, os valores-limite podem ser usados como entradas adicionais levando a regras adicionais e casos de teste.

Tipos de Defeitos

Defeitos típicos incluem processamento incorreto relacionado à lógica com base em combinações específicas de condições, obtendo-se resultados inesperados. Durante a criação das tabelas de decisão, defeitos podem ser encontrados no documento de especificação. Não é incomum preparar um conjunto de condições e determinar que o resultado esperado não seja especificado para uma ou mais regras. Os tipos mais comuns de defeitos são omissões de ação (ou seja, não há informações sobre o que realmente deve acontecer em uma determinada situação) e contradições.

3.2.4 Teste de transição de estado

O teste de transição de estado é usado para testar a capacidade do objeto de teste de entrar e sair de estados definidos por meio de transições válidas, assim como para tentar entrar em estados inválidos ou cobrir transições inválidas. Eventos fazem com que o objeto de teste faça a transição de estado para estado e execute ações. Os eventos podem ser qualificados por condições (às vezes chamados de guardas de transição ou condições de guarda) que influenciam o caminho de transição a ser seguido. Por exemplo, um evento de *logon* com uma combinação válida de nome de usuário e

senha resultará em uma transição diferente de um evento de *logon* com uma senha inválida. Essas informações são representadas em um diagrama de transição de estado ou em uma tabela de transição de estado (que também pode incluir possíveis transições inválidas entre estados).

Aplicabilidade

O teste de transição de estado é aplicável a qualquer software que tenha definido estados e tenha eventos que causarão as transições entre eles (p. ex., alteração de telas). O teste de transição de estado pode ser usado em qualquer nível de teste. São bons candidatos para esse tipo de teste o software incorporado, software da web e qualquer tipo de software transacional. Os sistemas de controle, por exemplo, controladores de semáforo, também são candidatos para esse tipo de teste.

Limitações e dificuldades

Determinar os estados geralmente é a parte mais difícil de definir o diagrama de transição de estados ou a tabela de transição de estados. Quando o objeto de teste possui uma interface com o usuário, as várias telas exibidas para o usuário geralmente são representadas por estados. Para software incorporado, os estados podem depender dos estados do hardware

Além dos próprios estados, a unidade básica dos testes de transição estatal é a transição individual. O simples teste de todas as transições individuais encontrará alguns tipos de defeitos de transição de estado, mas mais podem ser encontrados testando sequências de transições. Uma única transição é chamada de um interruptor 0; uma sequência de duas transições sucessivas é chamada de um interruptor 1; uma sequência de três transições sucessivas é chamada de um interruptor 2, e assim por diante. Em geral, um N-switch representa N+1 transições sucessivas [Chow1978]. Com o N aumentando, o número de interruptores N cresce muito rapidamente, dificultando a cobertura do interruptor N com um número razoável e pequeno de testes.

Cobertura

Como em outros tipos de técnicas de teste, há uma hierarquia de níveis de cobertura. O grau mínimo aceitável de cobertura é ter visitado todos os estados e percorrido todas as transições pelo menos uma vez. 100% de cobertura de transição (também conhecida como cobertura 100% *0-switch*) garantirá que todos os estados sejam visitados e todas as transições sejam percorridas, a menos que o desenho do sistema ou o modelo de transição de estado (diagrama ou tabela) esteja com defeito. Dependendo das relações entre os estados e transições, pode ser necessário percorrer algumas transições mais de uma vez para executar outras transições uma única vez.

O termo "cobertura de *N-switch*" refere-se ao número de interruptores (*switchs*) cobertos de comprimento N+1, como uma percentagem do número total de interruptores desse comprimento. Por exemplo, atingir 100% de cobertura de *1-switch* requer que cada sequência válida de duas transições sucessivas tenha sido testada pelo menos uma vez. Este teste pode desencadear alguns tipos de falhas que uma cobertura de 100% de *0-switch* falharia

A "c*obertura de ida e volta*" aplica-se a situações nas quais as sequências de transição formam loops. Uma cobertura de 100% de "*ida e volta*" é alcançada quando todos os loops de qualquer estado anterior ao mesmo estado foram testados para todos os estados nos quais os loops começam e terminam. Esse loop não pode conter mais de uma ocorrência de qualquer estado específico (exceto o inicial e final) [Offutt16].

Para qualquer uma dessas abordagens, um grau ainda mais alto de cobertura tentará incluir todas as transições inválidas identificadas em uma tabela de transição de estados. Os requisitos de cobertura e os conjuntos de coberturas para testes de transição de estado devem identificar se as transições inválidas estão incluídas.

A modelagem de casos de teste, para obter a cobertura desejada, é suportada pelo diagrama de transição de estado ou pela tabela de transição de estado para o objeto específico de teste. Esta informação também pode ser representada em uma tabela que mostra as transições *N-switch* para um valor específico de "N" [Black09].

Um procedimento manual pode ser aplicado para identificar os itens a serem cobertos (p. ex., transições, estados ou *N-switches*). Um método sugerido é imprimir o diagrama de transição de estados e a tabela de transição de estados e marcar os itens cobertos até que a cobertura necessária seja exibida [Black09]. Essa abordagem seria muito demorada para diagramas de transição de estado e tabelas de transição de estado mais complexas. Uma ferramenta deve, portanto, ser usada para suportar os testes de transição de estado.

Tipos de defeitos

Defeitos típicos incluem (ver também [Beizer95]):

- Tipos ou valores de evento incorretos;
- Tipos ou valores de ação incorretos;
- Estado inicial incorreto;
- Incapacidade de alcançar alguns estados de saída;
- Incapacidade de entrar nos estados requeridos;
- Estados extras (desnecessários);
- Incapacidade de executar corretamente algumas transições válidas;
- Capacidade de executar transições inválidas;
- Condições de guarda erradas.

Durante a criação do modelo de transição de estado, defeitos podem ser encontrados no documento de especificação. Os tipos mais comuns de defeitos são omissões (ou seja, não há informações sobre o que realmente deve acontecer em uma determinada situação) e contradições.

3.2.5 Técnica da árvore de classificação

As árvores de classificação suportam certas técnicas de teste caixa-preta, permitindo que uma representação gráfica do espaço de dados a ser criado, que se aplica ao objeto de teste.

Os dados são organizados em classificações e classes da seguinte forma:

- **Classificações**: representam parâmetros no espaço de dados para o objeto de teste, tais como parâmetros de entrada (que podem conter ainda estados ambientais e pré-condições), e parâmetros de saída. Por exemplo, se um aplicativo puder ser configurado de várias maneiras diferentes, as classificações podem incluir cliente, navegador, idioma e sistema operacional.
- **Classes**: cada classificação pode ter qualquer número de classes e subclasses descrevendo a ocorrência do parâmetro. Cada classe ou partição de equivalência é um valor específico

dentro de uma classificação. No exemplo acima, a classificação do idioma pode incluir partição de equivalência para inglês, francês e espanhol.

As árvores de classificação permitem que os Analistas de Teste insiram combinações como entenderem. Isso inclui, por exemplo, combinações em pares (consulte o capítulo 3.2.6), combinações triplas e simples.

Informações adicionais sobre o uso da técnica de árvore de classificação são fornecidas em [Bath14] e [Black09].

Aplicabilidade

A criação de uma árvore de classificação ajuda o Analista de Teste a identificar parâmetros (classificações) e suas partições de equivalência (classes) que são de interesse.

Uma análise mais aprofundada do diagrama de árvore de classificação permite identificar possíveis valores-limite e certas combinações de entrada que são de particular interesse ou que podem ser descontados (p. ex., por serem incompatíveis). A árvore de classificação resultante pode então ser utilizada para suportar a partição de equivalência, análise de valor-limite ou testes em pares (ver capítulo 3.2.6)

Limitações e dificuldades

À medida que a quantidade de classificações ou classes aumenta, o diagrama se torna maior e menos fácil de usar. Além disso, a Técnica da Árvore de Classificação não cria casos de teste completos, apenas combinações de dados de teste. Os Analistas de Teste devem fornecer os resultados para cada combinação de teste para criar casos de teste completos.

Cobertura

Os casos de teste podem ser projetados para atingir, por exemplo, a cobertura mínima de classe (ou seja, todos os valores em uma classificação testada pelo menos uma vez). O Analista de Teste também pode decidir cobrir combinações em pares ou usar outros tipos de testes combinatórios, por exemplo, três partes.

Tipos de defeito

Os tipos de defeitos encontrados dependem da(s) técnica(s) suportada(s) pelas árvores de classificação (ou seja, particionamento de equivalência, análise de valor limite ou teste em pares).

3.2.6 Teste em pares

O teste em pares é usado ao testar o software no qual vários parâmetros de entrada, cada um com vários valores possíveis, devem ser testados em combinação, dando origem a mais combinações do que é possível testar no tempo permitido. Os parâmetros de entrada podem ser independentes no sentido de que qualquer opção para qualquer fator (ou seja, qualquer valor selecionado para qualquer parâmetro de entrada) pode ser combinada com qualquer opção para qualquer outro fator, porém nem sempre é o caso (veja uma nota sobre os modelos de características abaixo). A combinação de um parâmetro específico (variável ou fator) com um valor específico desse parâmetro é chamada de par parâmetro-valor (p. ex., se “cor” é um parâmetro com sete valores permitidos, incluindo “vermelho”, então “cor = vermelho” poderia ser um par parâmetro-valor).

Os testes em pares utilizam técnicas combinatórias para assegurar que cada par de parâmetro-valor seja testado uma vez contra cada par de parâmetro-valor de cada outro parâmetro (ou seja, 'todos os pares' de pares de parâmetro-valor para quaisquer dois parâmetros diferentes são testados), evitando ao mesmo tempo testar todas as combinações de pares de parâmetro-valor. Se o Analista de Teste utiliza uma abordagem manual, uma tabela é construída com casos de teste representados por linhas e uma coluna para cada parâmetro. O Analista de Teste então preenche a tabela com valores tais que todos os pares de valores possam ser identificados na tabela (ver [Black09]). Quaisquer entradas na tabela que sejam deixadas em branco podem ser preenchidas com valores pelo Analista de Testes usando seu próprio conhecimento de domínio.

Há várias ferramentas disponíveis para ajudar um Analista de Testes nesta tarefa (ver www.pairwise.org para amostras). Eles exigem, como entrada, uma lista dos parâmetros e seus valores e geram um conjunto adequado de combinações de valores de cada parâmetro que cobre todos os pares de pares de parâmetros-valor. A saída da ferramenta pode ser usada como entrada para casos de teste. Observe que o Analista de Teste deve fornecer os resultados esperados para cada combinação que é criada pelas ferramentas.

As árvores de classificação (consulte o capítulo 3.2.5) são frequentemente usadas em conjunto com o teste em pares [Bath14]. O formato da árvore de classificação é suportado por ferramentas e permite a visualização de combinações de parâmetros e seus valores (algumas ferramentas oferecem um aprimoramento em pares). Isto ajuda a identificar:

- As entradas a serem usadas pela técnica de teste em pares;
- As combinações particulares de interesse (p. ex., frequentemente usadas ou uma fonte comum de defeitos);
- As combinações particulares que são incompatíveis. Isto não pressupõe que os fatores combinados não se afetem mutuamente, mas devem afetar um ao outro de maneiras aceitáveis;
- As relações lógicas entre variáveis. Por exemplo, "*se a variável1 = x, então a variável 2 não pode ser y*". As árvores de classificação que capturam esses relacionamentos são chamadas “modelos de recursos”.

Aplicabilidade

O problema de ter muitas combinações de valores de parâmetros se manifesta em pelo menos duas situações diferentes relacionadas ao teste. Alguns itens de teste envolvem vários parâmetros, cada um com vários valores possíveis, por exemplo, uma tela com vários campos de entrada. Nesse caso, combinações de valores de parâmetros compõem os dados de entrada para os casos de teste. Além disso, alguns sistemas podem ser configuráveis em várias dimensões, resultando em um espaço de configuração potencialmente grande. Em ambas as situações, o teste em pares pode ser usado para identificar um subconjunto de combinações que seja manejável e viável.

Para parâmetros com muitos valores, o particionamento de equivalência ou algum outro mecanismo de seleção pode primeiro ser aplicado a cada parâmetro individualmente para reduzir o número de valores de cada parâmetro, antes que o teste em pares seja aplicado para reduzir o conjunto de combinações resultantes. A captura dos parâmetros e seus valores em uma árvore de classificação suporta esta atividade.

Essas técnicas são geralmente aplicadas nos níveis de integração de componentes, sistema integração de sistemas.

Limitações e dificuldades

A principal limitação dessas técnicas é a suposição de que os resultados de alguns testes são representativos em todos os testes e que esses poucos testes representam o uso esperado. Se houver uma interação inesperada entre determinadas variáveis, ele pode não ser detectado com esta técnica de teste se essa combinação específica não for testada. É difícil explicar essas técnicas para um público não técnico, pois eles podem não entender a redução lógica dos testes. Qualquer explicação desse tipo deve ser equilibrada mencionando os resultados de estudos empíricos [Kuhn16], que mostraram em estudo que na área de dispositivos médicos, 66% das falhas foram desencadeadas por uma única variável e 97% por uma ou duas variáveis interagindo. Há um risco residual de que o teste em pares possa não detectar falhas no sistema em que três ou mais variáveis interagem.

Às vezes, é difícil conseguir identificar os parâmetros e seus respectivos valores. Portanto, esta tarefa deve ser executada com o apoio de árvores de classificação sempre que possível (consulte o capítulo 3.2.5). É difícil encontrar um conjunto mínimo de combinações para satisfazer um certo nível de cobertura manualmente. As ferramentas podem ser usadas para encontrar o menor conjunto possível de combinações. Algumas ferramentas suportam a capacidade de forçar algumas combinações a serem incluídas ou excluídas da seleção final de combinações. Esse recurso pode ser usado pelo Analista de Teste para enfatizar ou não enfatizar fatores com base no conhecimento do domínio ou nas informações de uso do produto.

Cobertura

A cobertura de 100% em pares exige que cada o par de valores de qualquer par de parâmetros seja incluído em pelo menos uma combinação.

Tipos de defeito

O tipo mais comum de defeitos encontrados nessa técnica de teste são os relacionados aos valores combinados de dois parâmetros.

3.2.7 Teste de casos de uso

O teste de caso de uso fornece testes transacionais baseados em cenário que devem emular o uso pretendido do componente ou sistema especificado pelo caso de uso. Os casos de uso são definidos em termos de interações entre os atores e um componente ou sistema que atinge algum objetivo. Os atores podem ser usuários humanos, hardware externo ou outros componentes ou sistemas.

Um padrão comum para casos de uso é fornecido em [OMG-UML].

Aplicabilidade

Os testes de caso de uso geralmente são aplicados nos testes de sistema e de aceite. Também pode ser usado em testes de integração se o comportamento dos componentes ou sistemas é especificado por casos de uso. Os casos de uso também costumam ser a base para o teste de performance, porque retratam o uso realista do sistema. Os cenários descritos nos casos de uso

podem ser atribuídos a usuários virtuais para criar uma carga real no sistema (desde que os requisitos de carga e performance sejam especificados neles ou para eles).

Limitações e dificuldades

Para serem válidos, os casos de uso devem transmitir transações realistas do usuário. As especificações de caso de uso são uma forma de modelo do sistema. Os requisitos do que os usuários precisam cumprir devem vir de usuários ou representantes de usuários e devem ser verificados em relação aos requisitos organizacionais antes de projetar os casos de uso correspondentes. O valor de um caso de uso é reduzido se não refletir os requisitos reais do usuário e da organização ou dificultar, em vez de auxiliar na conclusão das tarefas do usuário.

Uma definição precisa da exceção, comportamentos alternativos e de tratamento de erros é importante para que a cobertura do teste seja completa. Os casos de uso devem ser tomados como uma diretriz, mas não como uma definição completa do que deve ser testado, pois podem não fornecer uma definição clara de todo o conjunto de requisitos. Também pode ser benéfico criar outros modelos, como fluxogramas ou tabelas de decisão, a partir da narrativa do caso de uso para melhorar a precisão do teste e verificar o próprio caso de uso. Como com outras formas de especificação, é provável que revele anomalias lógicas na especificação de casos de uso, se existirem.

Cobertura

O nível mínimo aceitável de cobertura de um caso de uso é ter um caso de teste para o comportamento básico e casos de teste adicionais suficientes para cobrir cada alternativa e comportamento de manuseio de erros. Se um conjunto mínimo de testes for necessário, vários comportamentos alternativos podem ser incorporados a um caso de teste, desde que sejam mutuamente compatíveis. Se for necessária uma melhor capacidade de diagnóstico (p. ex., para auxiliar no isolamento de defeitos), um caso de teste adicional por comportamento alternativo pode ser projetado, embora comportamentos alternativos aninhados ainda exijam que alguns desses comportamentos sejam reunidos em um único caso de teste (p. ex., o comportamento alternativo de encerramento versus não-encerramento dentro de um comportamento de exceção de “nova tentativa”).

Tipos de defeito

Os defeitos incluem o manuseio incorreto de comportamentos definidos, comportamentos alternativos perdidos, processamento incorreto das condições apresentadas e mensagens de erros mal implementados ou incorretos.

3.2.8 Combinando as técnicas

Às vezes, as técnicas são combinadas para criar casos de teste. Por exemplo, as condições identificadas em uma tabela de decisão podem ser submetidas a partição de equivalência para descobrir várias maneiras pelas quais uma condição pode ser satisfeita. Os casos de teste cobririam não somente cada combinações de condições, mas também, para aquelas condições que foram particionadas, deveriam ser gerados casos para cobrir as partições de equivalência. Ao selecionar a técnica específica a ser aplicada, o Analista de Teste deve considerar a aplicabilidade da técnica, as limitações e dificuldades e os objetivos do teste em termos de cobertura e defeitos a serem detectados. Esses aspectos são descritos para as técnicas individuais abordadas neste capítulo. Pode

não haver uma única técnica “melhor” para uma situação. As técnicas combinadas geralmente oferecem a cobertura mais completa, desde que haja tempo e habilidade suficientes para aplicá-las corretamente.

3.3 Técnicas de teste baseadas na experiência

Os testes baseados na experiência utilizam a habilidade e a intuição dos testadores, juntamente com sua experiência em aplicações ou tecnologias similares, para direcionar os testes de forma a aumentar a detecção dos defeitos. Estas técnicas de teste variam de "testes rápidos" nos quais o testador não tem atividades formalmente pré-planejadas para realizar, até por sessões de testes pré-planejadas usando o regulamento de teste para a sessão de teste com scripts. São quase sempre úteis, mas têm particular valor quando os aspectos incluídos na seguinte lista de vantagens podem ser alcançados.

Vantagens do teste baseado na experiência:

- Pode ser uma boa alternativa para abordagens mais estruturadas nos casos quando a documentação do sistema é inexistente;
- Pode ser aplicado quando o tempo de teste é severamente restrito;
- Permite que os conhecimentos disponíveis no domínio e na tecnologia sejam aplicados nos testes. Isso pode incluir aqueles não envolvidos em testes, por exemplo, de Analistas de Negócios, ou clientes;
- Pode fornecer feedback antecipado aos desenvolvedores;
- Ajuda a equipe a familiarizar-se com o software à medida que é produzido;
- É eficaz quando falhas operacionais são analisadas;
- Permite que diversas técnicas de teste sejam aplicadas.

Desvantagens do teste baseado na experiência:

- Pode ser inadequado em sistemas que exigem documentação detalhada de teste;
- São difíceis de alcançar níveis altos de repetibilidade;
- A capacidade de avaliar com precisão a cobertura é limitada;
- Os testes são menos adequados para automação futura.

Ao usar abordagens reativas e heurísticas, os testadores normalmente usam testes baseados na experiência, que são mais reativos a eventos do que abordagens de teste pré-planejadas. Além disso, a execução e a avaliação são tarefas simultâneas. Algumas abordagens estruturadas para testes baseados na experiência não são totalmente dinâmicas, ou seja, os testes não são criados inteiramente ao mesmo tempo em que o testador executa o teste. Pode ser o caso, por exemplo, em que a suposição de erros é usada para direcionar aspectos específicos do objeto de teste antes da execução do teste.

Observe que, embora algumas ideias sobre cobertura sejam apresentadas para as técnicas discutidas aqui, as técnicas de teste baseadas na experiência não possuem critérios formais de cobertura.

3.3.1 Suposição de erros

Ao usar a técnica de suposição de erros, o Analista de Teste usa a experiência para adivinhar os possíveis erros que poderiam ter sido cometidos quando o código estava sendo projetado e desenvolvido. Quando os erros esperados são identificados, o Analista de Teste determina os melhores métodos a serem usados para descobrir os defeitos resultantes. Por exemplo, se o Analista de Teste espera que o software apresente falhas quando uma senha inválida for inserida, serão executados testes para inserir uma variedade de valores diferentes no campo de senha para verificar se o erro foi realmente cometido e resultou em um defeito que pode ser visto como uma falha quando os testes são executados.

Além de ser usada como técnica de teste, a detecção de erros também é útil durante a análise de risco para identificar possíveis modos de falha. [Myers11]

Aplicabilidade

A detecção de erros é realizada principalmente durante a integração e o teste do sistema, mas pode ser usada em qualquer nível de teste. Essa técnica é frequentemente usada com outras técnicas e ajuda a ampliar o escopo dos casos de teste existentes. A detecção de erros também pode ser usada com eficácia ao testar uma nova versão do software para testar defeitos comuns antes de iniciar um teste mais rigoroso e com script.

Limitações e dificuldades

As seguintes limitações e dificuldades se aplicam à detecção de erros:

- A avaliação da cobertura é difícil e varia amplamente com a capacidade e a experiência do Analista de Teste;
- É mais bem utilizado por um testador experiente, familiarizado com os tipos de defeitos geralmente introduzidos no tipo de código que está sendo testado;
- É comumente utilizado, mas frequentemente não é documentado e, portanto, pode ser menos reprodutível do que outras formas de teste;
- Os casos de teste podem ser documentados, mas de uma maneira que apenas o autor entenda e possa reproduzir.

Cobertura

Quando uma taxonomia com defeito é usada, a cobertura é um percentual determinado pelo número de itens de taxonomia testado dividido pelo número total de itens de taxonomia. Sem uma taxonomia de defeito, a cobertura é limitada pela experiência e conhecimento do testador e pelo tempo disponível. A quantidade de defeitos encontrados nesta técnica variará com base em quão bem o testador pode atingir áreas problemáticas.

Tipos de defeitos

Defeitos típicos são geralmente aqueles definidos na taxonomia de defeitos particulares ou "adivinhados" pelo Analista de Teste, que podem não ter sido encontrados nos testes caixa-preta.

3.3.2 Teste baseado em checklist

Ao aplicar a técnica de teste baseada em *checklist*, o Analista de Teste experiente usa uma lista generalizada de alto nível de itens a serem anotados, verificados ou lembrados, ou um conjunto de regras ou critérios com base nos quais um objeto de teste deve ser verificado. Esses *checklists* são criados com base em um conjunto de padrões, experiência e outras considerações. Por exemplo, um *checklist* padrão da interface do usuário pode ser empregado como base para testar um aplicativo. No desenvolvimento de software Ágil, os *checklists* podem ser criados a partir dos critérios de aceite para uma história de usuário.

Aplicabilidade

O teste baseado em *checklist* é mais eficaz em projetos com uma equipe de teste experiente familiarizada com o software em teste ou com a área coberta pelo *checklist* (p. ex., para aplicar com êxito um *checklist* da interface do usuário, o Analista de Teste pode estar familiarizado com o teste de interface do usuário, mas não o sistema específico em teste). Como as listas de verificação são de alto nível e tendem a não ter as etapas comumente detalhadas encontradas em casos e procedimentos de teste, o conhecimento do testador é usado para preencher essas lacunas. Ao remover o detalhamento das etapas, as listas de verificação necessitarão de baixa manutenção e podem ser aplicadas a várias versões semelhantes.

As listas de verificação são adequadas para projetos em que o software é lançado e alterado rapidamente. Isso ajuda a reduzir o tempo de preparação e manutenção da documentação de teste. Eles podem ser usados para qualquer nível de teste e para os testes de regressão e *smoke teste*.

Limitações e dificuldades

A natureza de alto nível das listas de verificação pode afetar a reprodutibilidade dos resultados dos testes. É possível que vários testadores interpretem os *checklists* de maneira diferente e sigam abordagens diferentes para atender aos itens do *checklist*. Isso pode causar resultados de teste diferentes, mesmo que o mesmo *checklist* seja usado. Isso pode resultar em uma cobertura mais ampla, mas às vezes a reprodutibilidade é sacrificada. Os *checklist* também podem resultar em excesso de confiança em relação ao nível de cobertura alcançado, pois o teste real depende do julgamento do testador. Os *checklist* podem ser derivadas de casos teste ou *checklist* mais detalhadas e tendem a crescer com o tempo. É necessária a manutenção para garantir que os *checklist* cubram os aspectos importantes do software em teste.

Cobertura

A cobertura pode ser determinada pelo percentual obtido pelo número de itens testados do *checklist* dividido pelo número total de seus itens. A cobertura é tão boa quanto o *checklist*, mas, devido à natureza de alto nível do *checklist*, os resultados variam com base no Analista de Teste que o executa.

Tipos de defeito

Os defeitos típicos encontrados com esta técnica causam falhas resultantes da variação dos dados, da sequência de etapas ou do fluxo de trabalho geral durante o teste.

3.3.3 Teste exploratório

O teste exploratório é caracterizado pelo fato de o testador aprender simultaneamente sobre o objeto de teste e seus defeitos, planejar o trabalho de teste a ser realizado, projetar e executar os testes e relatar os resultados. O testador ajusta dinamicamente as metas de teste durante a execução e prepara apenas a documentação leve. [Whittaker09]

Aplicabilidade

Um bom teste exploratório é planejado, interativo e criativo. Requer pouca documentação sobre o sistema a ser testado e é frequentemente usado em situações em que a documentação não está disponível ou não é adequada para outras técnicas de teste. O teste exploratório é frequentemente usado para complementar outras técnicas de teste e servir como base para o desenvolvimento de casos de teste adicionais. O teste exploratório é frequentemente usado no desenvolvimento de software Ágil para que o teste da história do usuário seja feito de maneira flexível e rápida, com apenas uma documentação mínima. No entanto, a técnica também pode ser aplicada a projetos que utilizam um modelo de desenvolvimento sequencial.

Limitações e dificuldades

A cobertura dos testes exploratórios pode ser esporádica e a reprodutibilidade destes testes realizados pode ser difícil. O uso de um regulamento de teste para designar as áreas a serem cobertas em uma sessão de teste e caixas tempo para determinar o tempo permitido para o teste são técnicas usadas para gerenciar os testes exploratórios. No final de uma sessão de teste ou conjunto de sessões, o Gerente de Testes pode realizar uma reunião para recolher os resultados dos testes e determinar os regulamentos de teste para as próximas sessões de teste.

Outra dificuldade das sessões de testes exploratórios é rastreá-las com precisão em um sistema de gerenciamento de testes. Às vezes, isso é feito criando casos de teste que são realmente sessões exploratórias. Isso permite que o tempo alocado para o teste exploratório e a cobertura planejada sejam rastreados com os outros esforços de teste.

Como a reprodutibilidade pode ser difícil de alcançar com os testes exploratórios, isso também pode causar problemas quando for necessário recuperar as etapas para reproduzir uma falha. Algumas organizações usam o recurso de captura/reprodução de uma ferramenta de automação de teste para registrar as etapas executadas por um testador exploratório. Isso fornece um registro completo de todas as atividades durante a sessão de testes exploratórios (ou qualquer sessão de teste baseada na experiência). Analisar os detalhes para encontrar a causa real de uma falha pode ser entediante, mas pelo menos há um registro de todas as etapas envolvidas.

Outras ferramentas podem ser usadas para capturar as sessões de teste exploratório, mas elas não registram os resultados esperados porque não capturam a interação da interface gráfica do usuário. Nesse caso, os resultados esperados devem ser anotados para que uma análise adequada dos defeitos possa ser realizada, se necessário. Em geral, recomenda-se que também sejam feitas anotações durante a execução de testes exploratórios para oferecer suporte à reprodutibilidade quando necessário.

Cobertura

Os regulamentos de teste podem ser concebidos para tarefas, objetivos e resultados específicos. As sessões de testes exploratórias são planejadas para atingir esses critérios. A carta também pode identificar onde concentrar o esforço de teste, o que está dentro e fora do escopo da sessão de teste e quais recursos devem ser comprometidos para concluir os testes planejados. Uma sessão pode ser usada para se concentrar em tipos de defeitos específicos e em outras áreas potencialmente problemáticas que podem ser tratadas sem a formalidade de teste programados.

Tipos de defeito

Defeitos típicos encontrados no teste exploratório são problemas baseados em cenários que foram perdidos durante o teste de adequação funcional com script, problemas que caem entre os limites funcionais e problemas relacionados ao fluxo de trabalho. Às vezes, problemas de performance e segurança também são descobertos durante testes exploratórios.

3.3.4 Técnicas de teste baseadas em defeitos

Uma técnica de teste baseada em defeitos é aquela em que o tipo de defeito procurado é usado como base para a modelagem do teste, com os testes derivados sistematicamente do que é conhecido sobre o tipo de defeito. Diferentemente dos testes caixa-preta, que derivam seus testes da base de testes, os testes baseados em defeitos derivam os testes das listas focadas em defeitos. Em geral, as listas podem ser organizadas em tipos de defeitos, causas principais, sintomas de falha e outros dados relacionados a defeitos. As listas padrão se aplicam a vários tipos de software e não são específicas para um produto. O uso dessas listas ajuda a alavancar o conhecimento padrão do setor para derivar os testes específicos. Ao aderir a listas específicas do setor, as métricas relacionadas à ocorrência de defeitos podem ser rastreadas nos projetos e até nas organizações. As listas de defeitos mais comuns são aquelas que são específicas da organização ou do projeto e fazem uso de conhecimentos e experiências específicos.

Os testes baseados em defeitos também podem usar listas de riscos identificados e cenários de risco como base para direcionar os testes. Essa técnica de teste permite que o Analista de Teste direcione um tipo específico de defeito ou trabalhe sistematicamente através de uma lista de defeitos conhecidos e comuns de um tipo específico. A partir destas informações, o Analista de Testes cria as condições de teste e os casos de teste que farão com que o defeito se manifeste (se ele existir).

Aplicabilidade

O teste baseado em defeitos pode ser aplicado em qualquer nível de teste, mas é mais comumente aplicado durante o teste do sistema.

Limitações e dificuldades

Existem várias taxonomias de defeitos, e podem se concentrar em tipos específicos de teste, como usabilidade. É importante escolher uma taxonomia aplicável ao software em teste (se houver alguma disponível). Por exemplo, pode não haver nenhuma taxonomia disponível para um software inovador. Algumas organizações compilaram suas próprias taxonomias de defeitos prováveis ou frequentemente vistos. Qualquer que seja a taxonomia de defeito usada, é importante definir a cobertura esperada antes de iniciar o teste.

Cobertura

A técnica fornece critérios de cobertura que são usados para determinar quando todos os casos de teste úteis foram identificados. Os itens de cobertura podem ser elementos estruturais, elementos da especificação, cenários de uso ou qualquer combinação destes, dependendo da lista de defeitos. Como uma questão prática, os critérios de cobertura para técnicas de teste baseadas em defeitos tendem a ser menos sistemáticos do que para técnicas de teste de caixa preta, na medida em que apenas regras gerais de cobertura são dadas e a decisão específica sobre o que constitui o limite de cobertura útil é facultativa. Como em outras técnicas, os critérios de cobertura não significam que todo o conjunto de testes esteja completo, mas que os defeitos considerados não sugerem mais nenhum teste útil com base nessa técnica.

Tipos de defeitos

Os tipos de defeitos descobertos geralmente dependem da taxonomia de defeito em uso. Por exemplo, se uma lista de defeitos da interface do usuário for usada, a maioria dos defeitos descobertos provavelmente estará relacionada à interface do usuário, mas outros defeitos poderão ser descobertos como um subproduto específico do teste.

3.4 Aplicando as técnicas de teste mais apropriadas

As técnicas de teste caixa-preta e baseadas na experiência são mais eficazes quando usadas juntas. Técnicas de teste baseadas na experiência preenchem as lacunas na cobertura que resultam de deficiências sistemáticas nas técnicas de teste caixa-preta.

Não existe uma técnica perfeita para todas as situações. É importante que o Analista de Teste compreenda as vantagens e desvantagens de cada técnica e seja capaz de selecionar a melhor técnica ou conjunto de técnicas para a situação, considerando o tipo de projeto, cronograma, acesso a informações, habilidades do testador e outros fatores que podem influenciar a seleção.

Na discussão de cada técnica de teste caixa-preta e a baseada em experiência (consulte os capítulos 3.2 e 3.3, respectivamente), as informações fornecidas em “aplicabilidade”, “limitações ou dificuldades” e “cobertura” orientam o Analista de Teste na seleção da técnica de teste mais apropriada para aplicar.

4 Teste das características de qualidade do software [180 min]

Palavras-chave

acessibilidade, adequação funcional, adequação funcional, capacidade de aprender, compatibilidade, correção funcional, experiência do usuário, atratividade, integridade funcional, interoperabilidade, operacionalidade, proteção contra erros do usuário, Software Usability Measurement Inventory (SUMI), usabilidade, Website Analysis and Measurement Inventory (WAMMI).

Objetivos de aprendizagem

4.1 Introdução

Sem objetivos de aprendizagem

4.2 Características de qualidade para testes de domínio do negócio

TA-4.2.1 (K2) Explicar quais técnicas de teste são apropriadas para testar a integridade funcional, a correção funcional e a adequação funcional.

TA-4.2.2 (K2) Definir os típicos defeitos a serem direcionados para as características de integridade funcional, correção funcional e adequação funcional.

TA-4.2.3 (K2) Definir quando as características de integridade funcional, correção e adequação devem ser testadas no ciclo de vida de desenvolvimento de software.

TA-4.2.4 (K2) Explicar as abordagens que seriam adequadas para verificar e validar a implementação dos requisitos de usabilidade e o atendimento das expectativas do usuário.

TA-4.2.5 (K2) Explicar o papel do Analista de Teste nos testes de interoperabilidade, incluindo a identificação dos defeitos a serem direcionados

TA-4.2.6 (K2) Explicar o papel do Analista de Teste nos testes de portabilidade, incluindo a identificação dos defeitos a serem direcionados.

TA-4.2.7 (K4) Para um determinado conjunto de requisitos, determinar as condições de teste necessárias para verificar as características funcionais ou não-funcionais da qualidade no escopo do Analista de Teste

4.1 Introdução

Enquanto o capítulo anterior descreveu técnicas específicas disponíveis para o testador, este capítulo considera a aplicação dessas técnicas na avaliação das características usadas para descrever a qualidade de aplicativos ou sistemas de software.

Este syllabus discute as características de qualidade que podem ser avaliadas pelo Analista de Teste. Os atributos a serem avaliados pelo Analista Técnico de Teste são considerados no syllabus do *ISTQB® CTAL-TTA Technical Test Analyst* [ISTQB_AL_TTA].

A descrição das características de qualidade do produto fornecida no ISO-25010 [ISO25010] é usada como um guia para definir suas características. O modelo ISO de qualidade do software divide a qualidade do produto em diferentes características de qualidade, cada uma das quais pode conter subcaracterísticas. Estas são apresentadas na tabela abaixo, juntamente com a indicação de quais características e subcaracterísticas são cobertas pelo syllabus de Analista de Teste e Analista Técnico de Teste:

Característica	Subcaracterísticas	TA	TTA
Adequação funcional	Correção funcional, adequação funcional, integridade funcional	X	
Confiabilidade	Maturidade, tolerância a falhas, recuperação, disponibilidade		X
Usabilidade	Reconhecimento de adequação, capacidade de aprender, operacionalidade, atratividade, proteção contra erros do usuário, acessibilidade	X	
Eficiência de performance	Comportamento temporal, utilização de recursos, capacidade		X
Manutenção	Analisabilidade, modificabilidade, testabilidade, modularidade, reutilização		X
Portabilidade	Adaptabilidade, instalabilidade, substituibilidade	X	X
Segurança	Confidencialidade, integridade, não-rejeição, contabilização, autenticidade		X
Compatibilidade	Coexistência		X
	Interoperabilidade	X	

Embora essa alocação de trabalho possa variar em diferentes organizações, é a que é seguida nos syllabi do *ISTQB®*.

Para todas as características e subcaracterísticas de qualidade discutidas neste capítulo, os típicos riscos devem ser reconhecidos para que uma estratégia de teste apropriada possa ser formada e documentada. O teste das características de qualidade requer atenção especial ao tempo no SDLC, nas ferramentas necessárias, na disponibilidade de software e na documentação e conhecimento técnico. Sem uma estratégia para lidar com cada característica e suas necessidades únicas de teste, o testador pode não ter um planejamento adequado, incrementar e incorporar o tempo de execução de teste no cronograma [Bath14]. Alguns desses testes, por exemplo, testes de usabilidade, podem exigir alocação de recursos humanos especiais, planejamento extensivo, laboratórios dedicados, ferramentas específicas, habilidades de teste especializadas e, na maioria dos casos, uma quantidade significativa de tempo. Em alguns casos, o teste de usabilidade pode ser realizado por um grupo separado de especialistas em usabilidade ou na experiência do usuário.

Embora o Analista de Teste possa não ser responsável pelas características de qualidade que exigem uma abordagem mais técnica, é importante que o Analista de Teste esteja ciente das outras características e compreenda as áreas sobrepostas para o teste. Por exemplo, um objeto de teste que falha no teste de performance pode provavelmente falhar no teste de usabilidade se for muito lento para o usuário usar efetivamente. Da mesma forma, um objeto de teste com problemas de interoperabilidade com alguns componentes provavelmente não está pronto para o teste de portabilidade, pois isso tende a obscurecer os problemas mais básicos quando o ambiente é alterado.

4.2 Características de qualidade para testes de domínio do negócio

O teste de adequação funcional é o foco principal do Analista de Teste. O teste de adequação funcional é focado em "*o que*" o objeto de teste faz. A base de teste para testes de adequação funcional é geralmente um requisito, uma especificação, um conhecimento específico de domínio ou uma necessidade implícita. Os testes adequação funcional variam de acordo com o nível de teste em que são realizados e podem ser influenciados pelo SDLC. Por exemplo, um teste de adequação funcional conduzido durante o teste de integração testará a adequação funcional de componentes de interface que implementam uma única função definida. No nível de teste do sistema, os testes de adequação funcional incluem o teste da adequação funcional do sistema como um todo. Para sistemas de sistemas, os testes de adequação funcional se concentrarão principalmente nos testes de ponta a ponta em todos os sistemas integrados. Uma grande variedade de técnicas de teste é empregada durante os testes de adequação funcional (ver capítulo 3).

No desenvolvimento de software Ágil, os testes de adequação funcional geralmente incluem o seguinte:

- Testar uma funcionalidade específica (p. ex., histórias de usuário) planejada para implementação em uma iteração específica;
- Teste de regressão para todas as funcionalidades inalteradas.

Além dos testes de adequação funcional abordados nesse capítulo, também existem certas características de qualidade que fazem parte da área de responsabilidade do Analista de Teste que são consideradas áreas de teste não-funcionais (focadas em "*como*" o objeto de teste fornece a funcionalidade).

4.2.1 Teste de correção funcional

A correção funcional envolve verificar a aderência do aplicativo aos requisitos especificados ou implícitos e pode incluir precisão computacional. O teste de correção funcional emprega muitas das técnicas de teste explicadas no capítulo 3 e geralmente usa a especificação ou um sistema legado como oráculo de teste. O teste de correção funcional pode ser realizado em qualquer nível de teste e é direcionado ao tratamento incorreto de dados ou situações.

4.2.2 Teste de adequação funcional

O teste de adequação funcional envolve avaliar e validar a adequação de um conjunto de funções para as tarefas especificas pretendidas. Esse teste pode ser baseado no modelo funcional (p. ex., casos de uso ou histórias de usuários). O teste de adequação funcional geralmente é realizado durante o teste do sistema, mas também pode ser realizado durante os estágios posteriores do teste de integração. Os defeitos descobertos neste teste são indicações de que o sistema não poderá atender às necessidades do usuário de uma maneira que será considerada aceitável.

4.2.3 Teste de integridade funcional

O teste de integridade funcional é executado para determinar a cobertura de tarefas e objetivos do usuário especificados pela funcionalidade implementada. A rastreabilidade entre os itens especificados (p. ex., requisitos, histórias de usuário, casos de uso) e a funcionalidade implementada (p. ex., função, componente, fluxo de trabalho) é essencial para permitir que a integridade funcional necessária seja determinada. A medição da integridade funcional pode variar de acordo com o nível específico de teste ou o SDLC usado. Por exemplo, a completude funcional para o desenvolvimento de software Ágil pode ser baseada em histórias e recursos de usuários implementados. A integridade funcional para testes de integração de sistemas pode se concentrar na cobertura de processos de negócios de alto nível.

A determinação da integridade funcional geralmente é suportada pelas ferramentas de gerenciamento de teste se o Analista de Teste estiver mantendo a rastreabilidade entre os casos de teste e os itens de especificação funcional.

Os níveis inferiores ao esperado da integridade funcional são indicações de que o sistema não foi totalmente implementado.

4.2.4 Teste de interoperabilidade

O teste de interoperabilidade verifica a troca de informações entre dois ou mais sistemas ou componentes. Os testes se concentram na capacidade de trocar informações e subsequentemente usar as informações que foram trocadas. Os testes devem abranger todos os ambientes de destino pretendidos (incluindo variações no hardware, software, middleware, sistema operacional etc.) para garantir que a troca de dados funcione corretamente. Na realidade, isso pode ser viável apenas para um número relativamente pequeno de ambientes. Nesse caso, o teste de interoperabilidade pode ser limitado a um grupo representativo selecionado de ambientes. A especificação de testes para interoperabilidade requer que as combinações dos ambientes de destino pretendidos sejam identificadas, configuradas e disponibilizadas para a equipe de teste. Esses ambientes são testados usando uma seleção de casos de testes de adequação funcional que exercem os vários pontos de troca de dados presentes no ambiente.

A interoperabilidade está relacionada à maneira como diferentes componentes e sistemas de software interagem entre si. O software com boas características de interoperabilidade pode ser integrado a vários outros sistemas sem exigir grandes alterações ou impacto significativo no comportamento não-funcional. O número de alterações e o esforço necessário para implementar e testar essas alterações podem ser usados como uma medida de interoperabilidade.

Testar a interoperabilidade de software pode, por exemplo, focar nos seguintes recursos de construção:

- Uso de padrões de comunicação em todo o setor, como XML;
- Capacidade de detectar e ajustar automaticamente as necessidades de comunicação dos sistemas com os quais interage.

O teste de interoperabilidade pode ser particularmente significativo para:

- Produtos e ferramentas comerciais de software prontos para uso;
- Aplicativos baseados em um sistema de sistemas;
- Sistemas baseados na Internet das Coisas;
- Serviços da Web com conectividade com outros sistemas.

Esse tipo de teste é realizado durante a integração de componentes e o teste de integração de sistema. No nível da integração do sistema, esse tipo de teste é realizado para determinar o quão bem o sistema totalmente desenvolvido interage com outros sistemas. Como os sistemas podem interoperar em vários níveis, o Analista de Teste deve entender essas interações e ser capaz de criar as condições que exercerão as várias interações. Por exemplo, se dois sistemas trocarem dados, o Analista de Teste deverá criar os dados e as transações necessárias para realizar a troca de dados. É importante lembrar que todas as interações podem não estar claramente especificadas nos documentos de requisitos. Em vez disso, muitas dessas interações serão definidas apenas nos documentos de arquitetura e modelagem do sistema. O Analista de Teste deve estar apto e preparado para examinar esses documentos e determinar os pontos de troca de informações entre sistemas e entre o sistema e seu ambiente para garantir que todos sejam testados. As técnicas como partição de equivalência, análise de valor-limite, tabelas de decisão, diagramas de transição de estados, casos de uso e teste em pares são aplicáveis ao teste de interoperabilidade. Os típicos defeitos encontrados incluem troca incorreta de dados na interação de componentes.

4.2.5 Teste de usabilidade

Os Analistas de Teste geralmente estão em posição de coordenar e apoiar a avaliação da usabilidade. Isso pode incluir a especificação de testes de usabilidade ou atuar como um moderador trabalhando com os usuários para realizar os testes. Para fazer isso de forma eficaz, o Analista de Teste deve entender os principais aspectos, objetivos e abordagens envolvidas nesses tipos de teste. Consulte o Syllabus *ISTQB® CTFL Usability Testing* [ISTQB_UT_SYL] para obter detalhes além da descrição fornecida neste capítulo.

É importante entender por que os usuários podem ter dificuldade em usar o sistema ou não ter uma experiência positiva do usuário (UX) (p. ex., com o uso de software para entretenimento). Para obter esse entendimento, primeiro é necessário compreender que o termo "usuário" pode ser aplicado a uma ampla variedade de tipos de personas, que variam de Especialistas em TI a crianças e pessoas com deficiência.

Aspectos de usabilidade

A seguir estão os três aspectos considerados nesse capítulo:

- Usabilidade;

- Experiência do usuário (UX);
- Acessibilidade.

Usabilidade

O teste de usabilidade tem como alvo os defeitos de software que afetam a capacidade do usuário de executar tarefas por meio da interface do usuário. Tais defeitos podem afetar a capacidade do usuário de atingir seus objetivos de forma eficaz, eficiente ou com satisfação. Problemas de usabilidade podem levar a confusão, erro, atraso ou falha total para concluir alguma tarefa por parte do usuário.

A seguir, estão as subcaracterísticas da usabilidade [ISO 25010]; para suas definições, ver [ISTQB_GLOSSARY]:

- *Reconhecimento de adequação* (ou seja, compreensibilidade);
- *Aprendizagem;*
- *Operabilidade;*
- *Estética da interface do usuário* (ou seja, atratividade);
- *Proteção contra erros do usuário;*
- *Acessibilidade* (veja abaixo).

Experiência do Usuário (UX)

A avaliação da experiência do usuário aborda toda a experiência do usuário com o objeto de teste, não apenas a interação direta. Isso é de particular importância para objetos de teste em que fatores como diversão e satisfação do usuário são críticos para o sucesso dos negócios.

Fatores típicos que influenciam a experiência do usuário incluem o seguinte:

- Imagem da marca (ou seja, confiança do usuário no fabricante);
- Comportamento interativo;
- A utilidade do objeto de teste, incluindo o sistema de ajuda, suporte e treinamento.

Acessibilidade

É importante considerar a acessibilidade do software para pessoas com necessidades ou restrições específicas para seu uso. Isso inclui pessoas com deficiência. Os testes de acessibilidade devem considerar os padrões relevantes, como as *Web Content Accessibility Guidelines (WCAG)*, e a legislação, como a *Lei Brasileira de Inclusão da Pessoa com Deficiência* (Brasil), o *Disability Discrimination Acts* (Irlanda do Norte, Austrália), *Equality Act 2010* (Inglaterra, Escócia, País de Gales) e *Section 508* (EUA). A acessibilidade, semelhante à usabilidade, deve ser considerada ao realizar as atividades de modelagem. O teste geralmente ocorre durante os níveis de integração e continua nos testes do sistema e nos níveis de teste de aceite. Os defeitos geralmente são determinados quando o software falha em atender aos regulamentos ou padrões designados definidos para o software.

As típicas medidas para melhorar a acessibilidade concentram-se nas oportunidades oferecidas aos usuários com deficiência para interagir com o aplicativo. Isso inclui:

- Reconhecimento de voz para entradas;
- Garantir que o conteúdo não textual apresentado ao usuário tenha uma alternativa em texto equivalente;
- Permitir que o texto seja redimensionado sem perda de conteúdo ou funcionalidade.

As diretrizes de acessibilidade dão suporte ao Analista de Teste, fornecendo uma fonte de informações e listas de verificação que podem ser usadas para testes (exemplos de diretrizes de acessibilidade são fornecidos em [ISTQB_FL_UT]). Além disso, estão disponíveis ferramentas e *plugins* de navegador para ajudar os testadores a identificar os problemas de acessibilidade, como má escolha de cores em páginas da Web que violam as diretrizes para daltonismo.

Abordagens de avaliação de usabilidade

A usabilidade, a experiência do usuário e a acessibilidade podem ser testadas por uma ou mais das seguintes abordagens:

- Teste de usabilidade;
- Revisão de usabilidade;
- Pesquisas e questionários de usabilidade;

Teste de usabilidade

O teste de usabilidade avalia a facilidade com que os usuários podem usar ou aprender a usar o sistema para alcançar uma meta específica em um contexto específico. O teste de usabilidade é direcionado para medir:

- **Eficácia**: capacidade do objeto de teste de permitir aos utilizadores atingir os objetivos específicos com precisão e exaustividade num contexto específico de utilização;
- **Eficiência**: capacidade do objeto de teste de permitir que os usuários gastem quantidades apropriadas de recursos em relação à eficácia alcançada em um contexto específico de uso;
- **Satisfação**: capacidade do objeto de teste de satisfazer usuários em um contexto específico de uso.

É importante observar que a modelagem e a especificação do teste de usabilidade geralmente são conduzidas pelo Analista de Teste em cooperação com Testadores que possuem habilidades especiais em teste de usabilidade e Engenheiros de Modelos de usabilidade que entendem o processo de modelagem centrada em humano (para mais detalhes consulte [ISTQB_FL_UT]).

Revisão de usabilidade

Inspeções e revisões são um tipo de teste realizado a partir de uma perspectiva de usabilidade que ajuda a aumentar o nível de envolvimento do usuário. Isso pode ser econômico, encontrando problemas de usabilidade nas especificações e projetos de requisitos no início do SDLC. A avaliação heurística (inspeção sistemática de um projeto da interface do usuário para usabilidade) pode ser usada para encontrar os problemas de usabilidade no projeto, para que possam ser tratados como parte de um processo do projeto iterativo. Isso envolve ter um pequeno conjunto de avaliadores examinando a interface e julgando sua conformidade com os princípios de usabilidade reconhecidos (as "heurísticas"). As análises são mais eficazes quando a interface do usuário é mais visível. Por exemplo, exemplos de capturas de tela geralmente são mais fáceis de entender e interpretar do que apenas descrever a funcionalidade fornecida por uma tela específica. A visualização é importante para uma revisão adequada da usabilidade da documentação.

Pesquisas e questionários de usabilidade

Técnicas de pesquisa e questionário podem ser aplicadas para reunir observações e comentários sobre o comportamento do usuário com o sistema. Pesquisas padronizadas e publicamente

disponíveis, como SUMI (*Software Usability Measurement Inventory*) e WAMMI (*Website Analysis and MeasureMent Inventory*) permitem comparações com um banco de dados de medições anteriores de usabilidade. Além disso, como o SUMI fornece medidas tangíveis de usabilidade, isso pode fornecer um conjunto de critérios de conclusão ou aceite.

4.2.6 Teste de portabilidade

Os testes de portabilidade estão relacionados ao grau em que um componente ou sistema de software pode ser transferido para o ambiente pretendido, como uma nova instalação ou como um ambiente já existente.

A classificação ISO-25010 das características de qualidade do produto inclui as seguintes subcaracterísticas de portabilidade:

- Instabilidade;
- Adaptabilidade;
- Substituibilidade;

A tarefa de identificar riscos e projetar testes para as características de portabilidade é compartilhada entre o Analista de Teste e o Analista Técnico de Teste (ver [ISTQB_AL_TTA] capítulo 4.7).

Teste de instalabilidade

O teste de instalabilidade é realizado no software e nos procedimentos escritos são usados para instalar e desinstalar o software em seu ambiente de destino.

Os objetivos típicos dos testes que são o foco do Analista de Teste incluem:

- Validar que diferentes configurações do software podem ser instaladas com sucesso. Onde um número elevado de parâmetros pode ser configurado, o Analista de Teste pode modelar testes usando a técnica em pares para reduzir o número de combinações de parâmetros testadas e focar em configurações específicas de interesse (p. ex., aquelas usadas com frequência);
- Testar a correção funcional dos procedimentos de instalação e desinstalação;
- Executar testes de adequação funcional após uma instalação ou desinstalação para detectar quaisquer defeitos que possam ter sido introduzidos (p. ex., configurações incorretas, funções indisponíveis);
- Identificar problemas de usabilidade nos procedimentos de instalação e desinstalação (p. ex., para validar que os usuários recebam instruções compreensíveis e mensagens de feedback ou erro ao executar o procedimento).

Teste de adaptabilidade

O teste de adaptabilidade verifica se uma determinada aplicação pode ser adaptada de forma eficaz e eficiente para funcionar corretamente em todos os ambientes-alvo previstos (hardware, software, middleware, sistema operacional, nuvem etc.). O Analista de Teste suporta testes de adaptabilidade identificando os ambientes-alvo pretendidos (p. ex., versões de diferentes sistemas operacionais móveis suportados, diferentes versões de navegadores que podem ser usados), e projetando testes que cobrem combinações destes ambientes. Os ambientes-alvo são então testados usando uma

seleção de casos de teste de adequação funcional que exercem os vários componentes presentes no ambiente.

Teste de substituibilidade

O teste de substituibilidade concentra-se na capacidade dos componentes ou versões de software em um sistema serem trocados por outros. Isso pode ser particularmente relevante para arquiteturas de sistema baseadas na Internet das Coisas, onde a troca de diferentes dispositivos de hardware ou instalações de software é uma ocorrência comum. Por exemplo, um dispositivo de hardware usado em um armazém para registrar e controlar os níveis de estoque pode ser substituído por um dispositivo de hardware mais avançado (p. ex., com um scanner melhor) ou o software instalado pode ser atualizado com uma nova versão que permite a emissão automática de pedidos de reposição de estoque para o sistema de um fornecedor.

Os testes de substituibilidade podem ser executados pelo Analista de Teste em paralelo aos testes de integração funcional, onde mais de um componente alternativo está disponível para integração no sistema completo.

5 Revisões [120 min]

Palavras-chave

revisão baseada em checklist

Objetivos de aprendizagem

5.1 Introdução

Sem objetivos de aprendizagem

5.2 Usando checklist nas revisões

TA-5.2.1 (K3) Identificar os problemas em uma especificação de requisitos de acordo com as informações do *checklist* fornecido no syllabus.

TA-5.2.2 (K3) Identificar problemas em uma história de usuário de acordo com as informações do *checklist* fornecido no syllabus.

5.1 Introdução

O Analista de Teste deve ser participante ativo no processo de revisão, fornecendo sua opinião exclusiva. Quando feitas corretamente, as revisões podem ser o maior e mais econômico contribuinte para a qualidade geral da entrega.

5.2 Usando ckecklist nas revisões

A revisão baseada em *checklist* é a técnica mais comum usada pelo Analista de Teste ao revisar a base de teste. Os *checklists* são usados durante as revisões para lembrar os participantes de verificar pontos específicos durante a revisão. Eles também ajudam a remover a personalização da revisão (p. ex., "*Este é o mesmo checklist que usamos para todas as revisões. Não estamos segmentando apenas o seu produto de trabalho*").

A revisão baseada em *checklist* pode ser executada genericamente para todas as revisões ou pode se concentrar em características específicas da qualidade, áreas ou tipos de documentos. Por exemplo, um *checklist* genérico pode verificar as propriedades gerais do documento, como um identificador exclusivo, nenhuma referência marcada como "*a ser determinada*", a formatação adequada e os itens de semelhantes de conformidade. Um *checklist* específico para um documento de requisitos pode conter verificações para o uso adequado dos termos "*deve*" e "*deveria*", verificações para a testabilidade de cada requisito declarado e assim por diante.

O formato dos requisitos também pode indicar o tipo do *checklist* a ser usado. Um documento de requisito que está no formato de texto narrativo terá critérios de revisão diferentes dos que os baseados em diagramas.

Os *checklist* também podem ser orientados para um aspecto específico, como:

- Um conjunto de habilidades de arquiteto/programador ou um conjunto de habilidades de testador, no caso do Analista de Testes, um *checklist* do conjunto de habilidades de testador seria a mais apropriada.
- Um certo nível de risco (p. ex., em sistemas críticos de segurança): os *checklists* normalmente incluem as informações específicas necessárias para o nível de risco;
- Uma técnica de teste específica: o *checklist* se concentrará nas informações necessárias para uma técnica específica (p. ex., regras a serem representadas em uma tabela de decisão);
- Um item particular de especificação, como um requisito, caso de uso ou história do usuário: estes são discutidos nos capítulos seguintes e geralmente têm um foco diferente dos usados pelo Analista Técnico de Teste para a revisão de código ou arquitetura.

5.2.1 Revisões de requisitos

Os itens a seguir são um exemplo do que um *checklist* orientado para requisitos pode incluir:

- Fonte do requisito (p. ex., pessoa, departamento);
- Testabilidade de cada requisito;
- Prioridade de cada requisito;
- Critérios de aceite para cada requisito;

- Disponibilidade de uma estrutura de chamada de caso de uso (se aplicável);
- Identificação exclusiva de cada requisito, caso de uso ou história do usuário;
- Versão de cada requisito, caso de uso ou história do usuário;
- Rastreabilidade para cada requisito desde os de negócios ou marketing;
- Rastreabilidade entre requisitos ou casos de uso (se aplicável);
- Uso de terminologia consistente (p. ex., uso de um glossário).

É importante lembrar que, se um requisito não for testável, o que significa que é definido de tal forma que o Analista de Teste não pode determinar como testá-lo, então existe um defeito nesse requisito. Por exemplo, um requisito que declara "*O software deve ser muito amigável*" não pode ser testado. Como o Analista de Teste pode determinar se o software é amigável ou muito amigável? Se, em vez disso, o requisito diz "*O software deve estar em conformidade com os padrões de usabilidade estabelecidos no documento de padrões de usabilidade, versão xxx*", e se o documento de padrões de usabilidade existir, esse é um requisito testável. Também é um requisito abrangente, porque esse requisito se aplica a todos os itens na interface. Nesse caso, esse requisito pode gerar facilmente muitos casos de teste individuais em um aplicativo não trivial. A rastreabilidade deste requisito, ou talvez do documento dos padrões de usabilidade, para os casos de teste também é crítica, porque se a especificação de usabilidade referenciada mudar, todos os casos de teste precisarão ser revisados e atualizados conforme necessário.

Um requisito também não pode ser testado se o testador não conseguir determinar se o teste foi aprovado ou reprovado ou se é incapaz de construir um teste que possa ser aprovado ou reprovado. Por exemplo, "*O sistema deve estar disponível 100% do tempo, 24 horas por dia, 7 dias por semana, 365 (ou 366) dias por ano*" não é testável.

Um *checklist* simples para revisões de casos de uso pode incluir as perguntas:

- O comportamento principal (caminho) está claramente definido?
- Todos os comportamentos alternativos (caminhos) são identificados, completos com o tratamento de erros?
- As mensagens da interface do usuário estão definidas?
- Existe apenas um comportamento principal (deve haver, caso contrário há casos de uso múltiplos)?
- Cada comportamento é testável?

5.2.2 Revisões da história do usuário

No desenvolvimento do software Ágil, os requisitos geralmente assumem a forma de histórias de usuários. Essas histórias representam pequenas unidades de funcionalidade demonstrável. Enquanto um caso de uso é uma transação do usuário que percorre várias áreas de funcionalidade, uma história do usuário é um recurso mais isolado e geralmente tem escopo definido pelo tempo necessário para desenvolvê-lo.

Um *checklist*[1] para uma história de usuário pode incluir:

- A história é apropriada para a iteração ou sprint de destino?
- A história é escrita a partir da visão da pessoa que a está solicitando?
- Os critérios de aceite são definidos e testáveis?
- O recurso está claramente definido e distinto?
- A história é independente de outras?
- A história é priorizada?
- A história segue o formato mais usado: "*Como <tipo de usuário>, quero <algum objetivo> para que <algum motivo>*" [Cohn04]

Se a história definir uma nova interface, será apropriado usar um *checklist* genérico da história (como a acima) e um *checklist* detalhado da interface do usuário.

5.2.3 Personalizando checklists

Um *checklist* pode ser personalizado baseando-se em:

- Organização (p. ex., considerando políticas, padrões, convenções, restrições legais da empresa);
- Projeto ou esforço de desenvolvimento (p. ex., foco, padrões técnicos, riscos);
- Tipo de produto de trabalho que está sendo revisado (p. ex., as revisões de código podem ser personalizadas para linguagens específicas de programação);
- Nível de risco do produto de trabalho que está sendo revisado;
- Técnicas de teste a serem usadas.

Bons *checklists* encontrarão problemas e ajudarão a iniciar discussões sobre outros itens que podem não ter sido especificamente mencionados no *checklist*. Usar uma combinação de *checklists* é uma maneira forte de garantir que uma revisão atinja a mais alta qualidade do produto de trabalho. O uso da revisão baseada em *checklist* com um padrão de *checklists*, como as mencionadas no Syllabus *CTFL Foundation Level*, e o desenvolvimento de *checklists* específicos da organização, como os mostrados acima, ajudarão o Analista de Teste a ser eficaz nas revisões.

Para mais informações sobre revisões e inspeções, consulte [Gilb93] e [Wiegers03]. Outros exemplos de *checklists* podem ser obtidos nas referências no capítulo 7.4.

[1] A pergunta do exame fornecerá um subconjunto do checklist do caso de uso com o qual se pode responder à pergunta

6 Ferramentas de teste e automação [90 min]

Palavras-chave

execução de teste, modelagem de teste, preparação de dados de teste, teste orientado por palavras-chave, script de teste

Objetivos de aprendizagem

6.1 Introdução

Sem objetivo de aprendizado

6.2 Teste orientado por palavras-chave

TA-6.2.1 (K3) Para um determinado cenário, determinar as atividades apropriadas para o Analista de Teste em um projeto de testes com base em palavras-chave.

6.3 Tipos de ferramentas de teste

TA-6.3.1 (K2) Explicar o uso e os tipos de ferramentas de teste aplicadas na modelagem de teste, preparação dos dados e execução do teste

6.1 Introdução

As ferramentas de teste podem melhorar significativamente a eficiência e a precisão dos testes. As ferramentas de teste e as abordagens de automação usadas pelo Analista de Teste são descritas neste capítulo. Note-se que os Analistas de Teste trabalham em conjunto com Desenvolvedores, Engenheiros de Automação de Teste e Analistas Técnicos de Teste para criar as soluções de automação de teste. A automação orientada por palavras-chave, em particular, envolve o Analista de Teste e aproveita sua experiência com os negócios e a funcionalidade do sistema.

Informações adicionais sobre o assunto de automação de teste e o papel do Engenheiro de Automação de Teste são fornecidas no syllabus do *ISTQB® CTAL-TAE Test Automation Engineer* [ISTQB_TAE_SYL].

6.2 Teste orientado por palavra-chave

O teste orientado por palavra-chave é uma das principais abordagens de automação de teste e envolve o Analista de Teste no fornecimento das principais entradas: palavras-chave e dados.

As palavras-chave são usadas principalmente, mas não exclusivamente, para representar interações comerciais de alto nível com um sistema (p. ex., "cancelar pedido"). Cada palavra-chave é normalmente usada para representar várias interações detalhadas entre um ator e o sistema em teste. As sequências de palavras-chave (incluindo dados de teste relevantes) são usadas para especificar os casos de teste [Buwalda02].

Na automação de teste, uma palavra-chave é implementada como um ou mais scripts de teste executáveis. As ferramentas leem os casos de teste escritos como uma sequência de palavras-chave que chamam os scripts de teste apropriados, que implementam a funcionalidade da palavra-chave. Os scripts são implementados de uma maneira altamente modular para permitir o fácil mapeamento de palavras-chave específicas. São necessárias habilidades de programação para implementar esses módulos de scripts.

A seguir, estão as principais vantagens do teste orientada por palavras-chave:

- As palavras-chave relacionadas a um aplicativo ou domínio de negócios específico podem ser definidas por especialistas em domínio. Isso pode tornar a tarefa de especificação de caso de teste mais eficiente;
- Uma pessoa com experiência principalmente no domínio pode se beneficiar da execução automática dos casos de teste (depois que as palavras-chave forem implementadas como scripts) sem precisar entender o código de automação subjacente;
- O uso de uma técnica modular de escrita permite a manutenção eficiente dos casos de teste pelo Engenheiro de Automação de Teste quando ocorrem alterações na funcionalidade e na interface do software em teste [Bath14];
- As especificações dos casos de teste são independentes de sua implementação.

Os Analistas de Teste geralmente criam e mantêm os dados das palavras-chave. Eles devem perceber que a tarefa de desenvolvimento do script ainda é necessária para implementar as palavras-chave. Depois de definidas as palavras-chave e os dados a serem utilizados, o Automatizador de Teste (p.

ex., Analista Técnico de Teste ou Engenheiro de Automação de Teste) converte as palavras-chave do processo de negócios e as ações de nível inferior em scripts de teste automatizados.

Enquanto o teste orientado por palavras-chave geralmente é executado durante o teste do sistema, o desenvolvimento do código pode começar tão cedo quanto a modelagem de teste. Em um ambiente iterativo, particularmente quando a integração ou implantação contínua são usadas, o desenvolvimento da automação de teste é um processo contínuo.

Depois que as palavras-chave e os dados de entrada são criados, o Analista de Teste assume a responsabilidade de executar os casos de teste orientados por palavras-chave e analisar quaisquer falhas que possam ocorrer.

Quando uma anomalia é detectada, o Analista de Teste deve ajudar na investigação da causa da falha para determinar se o defeito está nas palavras-chave, nos dados de entrada, no próprio script de automação de teste ou no aplicativo que está sendo testado. Geralmente, o primeiro passo na solução de problemas é executar o mesmo teste com os mesmos dados manualmente para verificar se a falha está no próprio aplicativo. Se isso não mostrar uma falha, o Analista de Teste deve revisar a sequência de testes que levou à falha para determinar se o problema ocorreu em uma etapa anterior (talvez introduzindo dados de entrada incorretos), mas o defeito não apareceu até mais tarde no processamento. Se o Analista de Teste não conseguir determinar a causa da falha, as informações da solução dos problemas devem ser passadas ao Analista Técnico de Teste ou ao Desenvolvedor para análise posterior.

6.3 Tipos de ferramentas de teste

Grande parte do trabalho do Analista de Teste requer o uso eficaz de ferramentas. Aprimorada por:

- Saber quais ferramentas usar;
- Saber que as ferramentas podem aumentar a eficiência do esforço de teste (p. ex., ajudando a fornecer uma melhor cobertura de teste no tempo permitido).

6.3.1 Ferramentas de modelagem de teste

As ferramentas de modelagem de teste são usadas para ajudar a criar casos de teste e dados de teste a serem aplicados no teste. Essas ferramentas podem funcionar com formatos, modelos de documentos de requisitos específicos (p. ex., UML) ou entradas fornecidas pelo Analista de Teste. As ferramentas de modelagem de teste geralmente são projetadas e criadas para funcionar com formatos e ferramentas específicas, como ferramentas de gerenciamento de requisitos específicos.

As ferramentas de modelagem de teste podem fornecer informações para o Analista de Teste utilizar ao determinar os tipos de testes necessários para obter o nível específico de cobertura do teste, a confiança no sistema ou as ações de mitigação de risco do produto. Por exemplo, as ferramentas de árvore de classificação geram (e exibem) o conjunto de combinações necessárias para alcançar a cobertura total com base em um critério de cobertura selecionado. Essas informações podem ser usadas pelo Analista de Teste para determinar os casos de teste que devem ser executados.

6.3.2 Ferramentas de preparação de dados de teste

As ferramentas de preparação de dados de teste podem fornecem os benefícios:

- Analisar um documento como um documento de requisitos ou mesmo o código fonte para determinar os dados necessários durante o teste para atingir um nível de cobertura.
- Limpar ou descaracterizar (remover qualquer informação pessoal) um conjunto de dados de um sistema em produção, mantendo a integridade interna desses dados. Os dados limpos podem ser usados para testes sem o risco de vazamento de segurança ou uso indevido de informações pessoais. Isso é particularmente importante quando grandes volumes de dados reais são necessários e onde riscos de segurança e privacidade de dados se aplicam.
- Gerar dados de teste sintéticos a partir de determinados conjuntos de parâmetros de entrada (p. ex., para uso em testes aleatórios). Algumas dessas ferramentas analisarão a estrutura do banco de dados para determinar quais entradas serão necessárias para o Analista de Teste.

6.3.3 Ferramentas de execução de teste automatizadas

As ferramentas de execução de teste são usadas pelos Analistas de Teste em todos os níveis de teste para executar testes automatizados e verificar os resultados reais. O objetivo de usar uma ferramenta de execução de teste é geralmente:

- Reduzir os custos (em termos de esforço ou tempo);
- Executar mais testes;
- Executar o mesmo teste em muitos ambientes;
- Tornar a execução do teste mais repetível;
- Executar testes que seriam impossíveis de executar manualmente (ou seja, testes de validação de dados grandes).

Esses objetivos geralmente se sobrepõem aos principais de aumentar a cobertura e reduzir custos.

O retorno do investimento para ferramentas de execução de teste geralmente é mais alto ao automatizar-se testes de regressão devido ao baixo nível de manutenção esperado e à execução repetida. A automação de *smoke test* também pode ser um uso eficaz da automação devido à frequência dos testes, à necessidade de um resultado rápido do teste e, embora o custo de manutenção possa ser maior, à capacidade de ter uma maneira automatizada de avaliar um novo *build* em um ambiente de integração contínua.

As ferramentas de execução de teste são comumente usadas durante os testes de sistema e integração. Algumas ferramentas, particularmente as de teste de API, também podem ser usadas no teste de componentes. Potencializar as ferramentas onde elas são mais aplicáveis ajudará a melhorar o retorno sobre o investimento.

Referências

Normas e Padrões

[ISO25010] ISO/IEC 25010 (2011) Systems and software engineering, Systems and software Quality Requirements and Evaluation (SQuaRE), System and software quality models, chapter 4

[ISO29119-4] ISO/IEC/IEEE 29119-4 Software and Systems Engineering, Software Testing, Part 4, Test Techniques, 2015

[OMG-DMN] Object Management Group: OMG® Decision Model and Notation™, Version 1.3, December 2019; url: www.omg.org/spec/DMN/, Chapter 8

[OMG-UML] Object Management Group: OMG® Unified Modeling Language®, Version 2.5.1,

December 2017; url: www.omg.org/spec/UML/

[RTCA DO-178C/ED-12C] Software Considerations in Airborne Systems and Equipment Certification, RTCA/EUROCAE ED12C, 2013

Documentos IREB®

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

Documentos ISTQB®

[ISTQB_AL_OVIEW] ISTQB® Advanced Level Overview, Version 2.0

[**ISTQB_ALTTA_SYL]** ISTQB® Advanced Level Technical Test Analyst Syllabus, Version 2019

[ISTQB_FL_SYL] ISTQB® Foundation Level Syllabus, Version 2018

[ISTQB_GLOSSARY] Standard glossary of terms used in Software Testing, url: https://glossary.istqb.org/

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

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

Literatura

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

[Beizer95] Boris Beizer, Black-box Testing, John Wiley & Sons, 1995, ISBN 0-471-12094-4

[Black02] Rex Black, Managing the Testing Process (2nd edition), John Wiley & Sons, New York, 2002, ISBN 0-471-22398-0

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

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

[Buwalda02] Hans Buwalda, Integrated Test Design and Automation: Using the Test Frame Method, Addison-Wesley Longman, 2002, ISBN 0-201-73725-6

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

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

[Copeland04] Lee Copeland, *A Practitioner's Guide to Software Test Design*, Artech House, 2004, ISBN 1-58053-791-X

[Craig02] Rick David Craig, Stefan P. Jaskiel; *Systematic Software Testing*; Artech House, 2002, ISBN 1-580-53508-9

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

[Gilb93] Tom Gilb, Graham Dorothy; *Software Inspection*; Addison-Wesley, 1993, ISBN 0-201-63181-4

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

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

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

[Offutt16] Jeff Offutt, Paul Ammann, *Introduction to Software Testing* (2nd edition), Cambridge University Press, 2016, ISBN 13: 9781107172012,

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

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

[Whittaker03] James Whittaker, *How to Break Software*, Addison-Wesley, 2003, ISBN 0-201-796198

[Whittaker09] James Whittaker, Exploratory software testing: tips, tricks, tours, and techniques to guide test design, Addison-Wesley, 2009, ISBN 0-321-63641-4

Outras referências

As seguintes referências apontam para informações disponíveis na Internet e em outros lugares. Mesmo que estas referências tenham sido verificadas no momento da publicação deste syllabus de nível avançado, o *ISTQB®* não ser responsabilizado se as referências não estiverem mais disponíveis.

Capítulo 3

- Czerwonka, Jacek: www.pairwise.org
- Defect taxonomy: www.testingeducation.org/a/bsct2.pdf
- Sample defect taxonomy based on Boris Beizer's work: inet.uni2.dk/~vinter/bugtaxst.doc
- Good overview of various taxonomies: testingeducation.org/a/bugtax.pdf
- Heuristic Risk-Based Testing By James Bach
- Exploring Exploratory Testing, Cem Kaner and Andy Tinkham, www.kaner.com/pdfs/ExploringExploratoryTesting.pdf
- Pettichord, Bret, "An Exploratory Testing Workshop Report", www.testingcraft.com/exploratorypettichord

Capítulo 5

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

Apêndice A

A tabela a seguir é derivada da tabela completa fornecida na ISO-25010. Ela se concentra apenas nas características de qualidade cobertas por esse syllabus e compara os termos usados na ISO-9126 (como usado na versão 2012 do syllabus) com os da nova ISO-25010 (como usada nesta versão).

ISO/IEC 25010	**ISO/IEC 9126-1**	**Notas**
Adequação funcional	***Funcionalidade***	
Integridade funcional		
Correção funcional	Acurácia	
Adequação funcional	Adequação	
	Interoperabilidade	Movido para compatibilidade
Usabilidade		
Reconhecimento de adequação	Entendibilidade	Novo nome
Aprendizagem	Aprendizagem	
Operabilidade	Operabilidade	
Proteção contra erros do usuário		Nova subcaracterística
Atratividade	Atratividade	Novo nome
Acessibilidade		Nova subcaracterística
Compatibilidade		Nova definição
Interoperabilidade		
Coexistência		Coberto no CTAL-TTA