Certified Tester

Test Automation Strategy

Syllabus

v1.0

International Software Testing Qualifications Board

Direitos autorais

Histórico da Revisão

Versão	Data	Observações
CT-TAS v1.0	03/05/2024	CT-TAS v1.0 GA Release

Histórico de Revisão da versão na Língua Portuguesa

Versão	Data	Observações
RC	18/06/2024	Tradução para a língua portuguesa
2.0br	29/07/2024	Lançamento da versão na Língua Portuguesa

Índice

Agradecimentos

Este documento foi formalmente divulgado pela Assembleia Geral do ISTQB® em 03 de maio de 2024.

Ele foi produzido pela Test Automation Task Force do Specialist Working Group do International Software Testing Qualifications Board: Graham Bath (Presidente do Specialist Working Group), Andrew Pollner (Vice-Presidente do Specialist Working Group e Presidente da Test Automation Task Force), Péter Földházi, Patrick Quilter, Gergely Ágnecz, László Szikszai. Os revisores da Test Automation Task Force incluíram: Armin Beer, Armin Born, Geza Bujdoso, Renzo Cerquozzi, Jan Giesen, Arnika Hryszko, Kari Kakkonen, Gary Mogyorodi, Chris van Bael, Carsten Weise, Marc-Florian Wendland.

Revisor técnico: Gary Mogyorodi

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

Horváth Ágota, Laura Albert, Prasunkumar Banerjee, Jürgen Beniermann, Armin Born, Piet de Roo, Nicola De Rosa, Dingguofu Ding Guofu, Elizabeta Fourneret, Jan Giesen, Erik Haartmans, Matthias Hamburg, Tobias Horn, Mattijs Kemmink, Ilia Kulakov, Ashish Kumar, Vincenzo Marrazzo, Marton Matyas, Patricia McQuaid, Smitha Mohandas, Ingvar Nordström, Sreeja Padmakumari, Nishan Portoyan, Meile Posthuma, Swapnil Shah, Péter Sótér, Szilard Szell, Richard Taylor, Giancarlo Tomasig, Chris Van Bael, Daniel Van der Zwan, Carsten Weise, Marc-Florian Wendland, Claude Zhang.

Agradecimentos do BSTQB

O BSTQB® agradece à equipe do **BSTQB WGT** (Grupo de Trabalho de Traduções do BSTQB) pelo empenho em traduzir este material. Atuaram na tradução e revisão: Eduardo Medeiros Rodrigues, George Fialkovitz, Irene Nagase, Osmar Higashi, Paula Oliveira, Rogério Athaide de Almeida, Stênio Viveiros, Thiago Cesar Andrade.

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

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

0 Introdução

0.1 Objetivo deste syllabus

Esse syllabus forma a base para a International Software Testing Qualification para a certificação Test Automation Strategy. O ISTQB® fornece esse syllabus da seguinte forma:

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

0.2 Test Automation Strategy em Testes de Software.

A certificação de especialização Test Automation Strategy destina-se a qualquer pessoa envolvida em testes de software e automação de teste. Isso inclui pessoas em funções como testadores, analistas de teste, engenheiros de automação de teste, consultores de teste, arquitetos de teste, gerentes de teste e desenvolvedores de software. Essa qualificação de especialização também é adequada para qualquer pessoa que queira ter uma compreensão básica da automação de teste, como gerentes de projeto, gerentes de qualidade, gerentes de desenvolvimento de software, analistas de negócios, diretores de TI e consultores de gerenciamento.

O syllabus Test Automation Strategy (CT-TAS) apresenta vários fatores que entram em jogo ao planejar a automação de teste em uma organização. Os aspectos de implementação técnica da engenharia dos métodos de automação de teste e as melhores práticas não estão no escopo, pois são abordados em separado no syllabus CTAL-TAE do ISTQB®.

O Test Automation Strategy aborda as necessidades de automação de teste além daquelas que são desafios de implementação e integração de ferramentas técnicas. Uma visão estratégica da automação de teste oferece uma visão da implementação em projetos dentro de uma organização de forma sistemática e consistente que, em última análise, demonstra valor para a organização.

0.3 Carreira para Testadores e Engenheiros de Automação de Teste

O esquema ISTQB® oferece suporte para profissionais de teste em todos os estágios de suas carreiras, oferecendo amplitude e profundidade de conhecimento. As pessoas que obtiverem a certificação ISTQB® Test Automation Strategy também podem se interessar pela certificação Test Automation Engineering (CTAL-TAE).

As pessoas que obtiverem a certificação de especialização ISTQB® Test Automation Strategy também podem se interessar pelas certificações de nível avançado (Test Analyst, Technical Test Analyst, e Test Management) e, posteriormente, pelas certificações no nível expert (Test Management ou Improving the Test Process). Quem quiser desenvolver habilidades em práticas de teste em uma área de ambiente ágil pode considerar as certificações Agile Technical Tester ou Agile Test Leadership at Scale. O fluxo de especializações oferece um mergulho profundo em áreas que têm abordagens e atividades de teste específicas, por exemplo, em Performance Testing, Security Testing, AI Testing, e Mobile Application Testing ou onde o conhecimento específico do domínio é necessário (p. ex., Automotive Software Testing ou Game Testing). Acesse www.istqb.org para obter as informações mais recentes sobre o Esquema de Certificação em Testes do ISTQB®.

0.4 Resultados de Negócio

Esta seção lista os resultados de negócio esperados de um candidato que tenha obtido a certificação Test Automation Strategy.

Um testador certificado de especialista em Test Automation Strategy pode:

TAS-B01	Compreender os fatores do software e dos sistemas que influenciam o sucesso da automação de teste
TAS-B02	Identificar os custos e os riscos da implementação de uma solução de automação de teste
TAS-B03	Compreender as funções e responsabilidades das pessoas que estão contribuindo para automação de teste
TAS-B04	Planejar a integração da automação de teste em todos os níveis de teste
TAS-B05	Identificar considerações estratégicas para a implementação da automação de teste em diferentes modelos de ciclo de vida de desenvolvimento de software
TAS-B06	Compreender a aplicabilidade e a viabilidade da automação de teste
TAS-B07	Planejar soluções de automação de teste que atendam às necessidades organizacionais
TAS-B08	Compreender as estratégias de implantação da automação de teste
TAS-B09	Compreender as dependências da automação de teste no ambiente de testes
TAS-B10	Compreender os custos de configuração e manutenção da automação de teste
TAS-B11	Saiba quais métricas de automação de teste ajudam a orientar a tomada de decisões
TAS-B12	Identificar maneiras pelas quais a automação de teste agrega valor ao projeto e à organização
TAS-B13	Identificar os requisitos de relatórios de automação de teste para atender às necessidades dos stakeholders
TAS-B14	Definir atividades de transição de testes manuais para automação de teste
TAS-B15	Definir uma estratégia de automação de teste que garanta que os projetos compartilhem ativos e métodos para assegurar uma implementação consistente em toda a organização.

0.5 Objetivos de Aprendizagem e Nível Cognitivo de Conhecimento

Os Objetivos de Aprendizagem (LO) apoiam os resultados de negócio e são usados para criar os exames para a Certified Tester, Test Automation Strategy.

Em geral, todo o conteúdo deste syllabus pode ser examinado nos níveis K2 e K3, exceto a Introdução e os Apêndices. Ou seja, o candidato pode ser solicitado a reconhecer, lembrar ou recordar uma palavra-chave ou conceito mencionado em qualquer um dos seis capítulos. Os níveis específicos dos Objetivos de Aprendizagem são mostrados no início de cada capítulo e classificados da seguinte forma:

- K2: Compreender
- K3: Aplicar

Mais detalhes e exemplos de objetivos de Aprendizagem são fornecidos no Apêndice A.

Todos os termos listados como palavras-chave logo abaixo dos títulos dos capítulos devem ser lembrados, mesmo que não sejam explicitamente mencionados nos objetivos de Aprendizagem.

0.6 Exame de Certificação Test Automation Strategy

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

Consulte o documento *Exam Structures and Rules v1.1 Compatible with Syllabus Foundation and Advanced Levels and Specialist Modules* para obter mais detalhes sobre o syllabus para o exame de certificação de especialista em *Test Automation Strategy*.

O critério de entrada para fazer o exame de certificação Test Automation Strategy é que os candidatos tenham interesse em testes de software e automação de teste. Entretanto, é altamente recomendável que os candidatos também tenham:

- Formação mínima em desenvolvimento de software e sistemas e liderança para implementação de tecnologia na empresa e experiência como Engenheiro de Testes Sênior, Líder de Teste ou Desenvolvedor de Software.
- Fazer um curso que tenha sido credenciado de acordo com os padrões do ISTQB® (por um dos Conselhos Membros).

Observação sobre os pré-requisitos: o certificado ISTQB® Foundation Level deve ser obtido antes de fazer o exame de certificação ISTQB® Test Automation Strategy.

0.7 Credenciamento

Um Conselho Membro do ISTQB® pode credenciar Provedores de Treinamento cujo material do curso siga este syllabus. Os Provedores de Treinamento devem obter as diretrizes de credenciamento do Conselho de Membro ou do órgão que realiza o credenciamento. Um curso credenciado é reconhecido como estando em conformidade com este syllabus e pode ter um exame ISTQB® como parte do curso.

As diretrizes de credenciamento para este syllabus seguem as diretrizes *Accreditation Guidelines* publicadas pelo *Processes Management and Compliance Working Group*.

0.8 Manuseio de Normas

Há padrões referenciados no syllabus Test Automation Strategy (p. ex., IEEE e ISO). O objetivo dessas referências é fornecer uma estrutura (como nas referências à ISO 25010 com relação às características de qualidade) ou fornecer uma fonte de informações adicionais, se o leitor desejar. Observe que o syllabus usa os documentos normativos como referência. Os documentos normativos não se destinam a exame. Consulte o Capítulo 7 Referências para obter mais informações sobre as normas.

0.9 Mantendo-se atualizado

O setor de software muda rapidamente. Para lidar com essas mudanças e fornecer aos stakeholders acesso a informações relevantes e atuais, os grupos de trabalho do ISTQB® criaram links no site www.istqb.org, que se referem a documentos de apoio e a mudanças nos padrões. Essas informações não são passíveis de exame no syllabus Test Automation Strategy.

0.10 Nível de detalhe

O nível de detalhe desse syllabus permite cursos e exames consistentes no nível internacional. Para atingir esse objetivo, o syllabus consiste em:

- Objetivos gerais de instrução que descrevem a intenção do especialista em Test Automation Strategy;
- Uma lista de termos que os alunos devem ser capazes de lembrar;
- Objetivos de Aprendizagem para cada área de conhecimento, descrevendo o resultado cognitivo de aprendizagem a ser alcançado;
- Uma descrição dos principais conceitos, incluindo referências a fontes, como literatura ou padrões aceitos.

O conteúdo do syllabus não é uma descrição de toda a área de conhecimento de testes de software; ele reflete o nível de detalhes a ser abordado nos cursos de treinamento Test Automation Strategy. Ele se concentra nos conceitos e nas técnicas de teste que podem ser aplicados a todos os projetos de software, incluindo os que

seguem os métodos Ágeis. Este syllabus não contém nenhum objetivo de Aprendizagem específico relacionado a testes ágeis, mas discute como esses conceitos se aplicam a projetos ágeis e a outros tipos de projetos.

0.11 Como este syllabus está organizado

Há seis capítulos com conteúdo passível de exame. O título de nível superior de cada capítulo especifica o tempo para o capítulo; o tempo não é fornecido abaixo do nível do capítulo. Para cursos de treinamento credenciados, o syllabus exige um mínimo de 12,75 horas de instrução, distribuídas pelos seis capítulos da seguinte forma:

- Capítulo 1: 45min - Introdução e objetivos da Estratégia de Automação de Teste.
  - Compreender os conceitos de automação de teste e aprender os critérios de seleção para projetos candidatos;
  - Compreender os fatores que definem uma implementação bem-sucedida da automação de teste.
- Capítulo 2: 60min - Recursos de Automação de Teste.
  - Aprender as diferentes soluções disponíveis para automação de teste e o investimento relativo de cada uma delas;
  - Entender como o licenciamento de software para ferramentas de automação de teste é abordado;
  - Entender as habilidades necessárias para a automação de teste.
- Capítulo 3: 225min – Preparando-se para a Automação de Teste.
  - Aprender como a automação de teste é usada em todos os níveis de teste;
  - Entender as estratégias de automação de teste para distribuir adequadamente os testes e alcançar o Shift Left e o Shift Right;
  - Aprender como a automação de teste oferece suporte a projetos legados e ágeis;
  - Entender a automação de teste em DevOps e em práticas de testes contínuos;
  - Entender como definir critérios para o uso da automação de teste, incluindo os testes mais adequados para a automação de teste;
- Capítulo 4: 135min - Estratégias Organizacionais de Implantação e Liberação para Automação de Teste.
  - Aprender como a automação de teste pode melhorar o tempo de lançamento no mercado;
  - Entender como desenvolver testes automatizados relevantes para a operação e relatar defeitos;
  - Entender a abordagem da estratégia de implantação da automação de teste e a mitigação de riscos;
  - Aprender sobre o ambiente de automação de teste e suas dependências;
  - Entender como a integração da automação de teste e dos dados de teste a um sistema em teste é coberta.
- Capítulo 5: 150min - Análise de impacto da Automação de Teste.
  - Entender a abordagem de métricas e relatórios de automação de teste para ajudar a informar as decisões;
  - Aprender a realizar um retorno sobre o investimento para a automação de teste;
  - Entender como são abordados os objetivos de uma organização e de um projeto para usar a automação de teste;
  - Aprender a analisar os relatórios de teste e a informar os tomadores de decisão de forma clara e compreensível.
- Capítulo 6: 150min - Estratégias de Implementação e Melhoria para Automação de Teste.
  - Aprender a fazer a transição do teste manual para a automação de teste e para o teste contínuo;
  - Entender como a avaliação da automação de teste para melhoria contínua é abordada.

1 Introdução e Objetivos da Estratégia de Automação de Teste (45min)

Palavras-chave

arquitetura de automação de teste, estrutura de automação de teste, estratégia de automação de teste

Objetivos de Aprendizagem:

1.1 Fatores de sucesso de um projeto de Automação de Teste

CT-TAS-1.1.1 (K2) Explicar os objetivos e a relevância da automação de testes.

CT-TAS-1.1.2 (K2) Identificar fatores técnicos de sucesso de um projeto de automação de teste.

CT-TAS-1.1.3 (K2) Resumir os critérios de investimento apropriados na seleção de projetos candidatos para automação de teste.

1.1 Fatores de sucesso de um projeto de Automação de Teste

1.1.1 Definir metas e objetivos para a estratégia de automação de teste.

Ao definir a estratégia de um projeto de automação de teste é necessário abordar os seguintes aspectos:

- Definição do objetivo pretendido;
- Identificação de riscos;
- Definição do escopo;
- Identificação dos stakeholders envolvidos;
- Seleção de ferramentas para automação;
- Projetar a arquitetura de automação de teste (TAA);
- Identificação de ambientes.

Os objetivos da automação de teste podem incluir:

- Eficiência aprimorada do teste;
- Cobertura mais ampla e profunda;
- Melhoria da qualidade geral do sistema em teste (SUT);
- Redução do custo total e do tempo de lançamento no mercado;
- Realização de testes que os testadores manuais não podem fazer;
- Redução do tempo de execução do teste;
- Aumento da frequência dos testes.

Como o desenvolvimento ágil de software proporciona ciclos de implantação mais rápidos e os aplicativos em nuvem estão mais difundidos, o desenvolvimento exige uma resposta mais rápida e antecipada sobre a qualidade do SUT. Como resultado dessa mudança, a automação de teste tem mais foco e relevância nos projetos modernos, dada sua natureza e seus objetivos de teste.

1.1.2 Identificar os fatores técnicos de sucesso de um projeto de automação de teste.

Os fatores de sucesso a seguir se aplicam a projetos de automação de teste que se concentram em fatores que afetam o sucesso de longo prazo de um projeto. O uso de um projeto piloto ajuda a determinar as ferramentas e a viabilidade das tecnologias selecionadas.

Os fatores de sucesso da automação de teste incluem:

- Testabilidade do SUT.
  - Permite que a automação de teste acesse as interfaces do SUT.
- Estratégia de automação de teste definida.
  - A estratégia precisa ser aplicável e personalizável; suas metas precisam ser atingidas dentro das restrições de tempo e custo, e ela precisa ser mantida atualizada.
- TAA (Arquitetura de Automação de Teste)
  - Clareza sobre o que e como implementar.
  - Para obter informações adicionais, consulte o syllabus CTAL-TAE, seção 3.1.1.
- O Framework de Automação de Teste (TAF).
  - TAF que seja fácil de usar, bem documentado e passível de manutenção, que ofereça suporte a uma abordagem consistente para a automação de teste. Para obter mais informações, consulte o syllabus CTAL-TAE, seção 3.1.3.

- Definição e implementação de relatórios de testes.
- Fácil solução de problemas.
- Ambiente de teste adequado.
- Casos de teste automatizados documentados.
- Testes automatizados rastreáveis para definições e requisitos de casos de teste.
- Fácil manutenção.
- Testes automatizados atualizados.
- Um plano de implantação claro.
- Um plano claro de execução de testes.
- Testes retirados conforme necessário.
- Tratamento eficaz de exceções.

Antes de iniciar o projeto de automação de teste, é importante analisar a chance de sucesso do projeto, considerando os fatores existentes e os fatores ausentes, tendo em mente os riscos da abordagem de teste escolhida, bem como o contexto do projeto. Nem todos os fatores são necessários e, na prática, raramente todos os fatores são atendidos.

1.1.3 Resumir os critérios de investimento apropriados na seleção de projetos candidatos para automação de teste.

A configuração da automação de teste em um projeto significa investimento e, na maioria dos casos, um custo significativo. Isso deve ser levado em consideração, juntamente com a natureza do projeto, e se a automação de teste deve ser usada.

Considere os seguintes critérios de investimento antes de introduzir a automação de teste:

- Custo inicial: Além do trabalho necessário para configurar a automação de teste, serão necessários custos adicionais para o projeto, pois pode haver a necessidade de contratar novos Engenheiros de Automação de Teste (TAE), comprar novo hardware ou configurar o treinamento.
- Fase atual do Ciclo de Vida de Desenvolvimento de Software (SDLC) do projeto: É melhor começar o mais cedo possível para que a automação de teste possa agregar mais valor, o mais cedo possível.
- Duração prevista/planejada do projeto/desenvolvimento de software: Em um projeto mais curto, pode não haver recursos suficientes para iniciar ou tempo restante para que a automação de teste agregue valor.
- Custo de manutenção: no caso de começar do zero, a configuração de uma Solução de Automação de Teste (TAS) levará tempo, além de precisar de manutenção.

Se o investimento na introdução da automação de teste em um projeto for aceitável, a preparação para a automação de teste poderá começar. Isso inclui a seleção da abordagem de teste correta e das ferramentas de automação de teste. A principal responsabilidade por esse processo é uma função estratégica de um arquiteto ou gerente de testes, que tem o conhecimento necessário sobre automação de teste para tomar decisões relevantes.

2 Recursos de Automação de Teste (60min)

Palavras-chave

engenheiro de automação de teste, solução de automação de teste

Objetivos de Aprendizagem:

2.1 Custos e riscos da implementação de uma solução de Automação de Teste

CT-TAS-2.1.1 (K2) Comparar soluções técnicas alternativas com relação ao custo de propriedade.

CT-TAS-2.1.2 (K2) Explicar as considerações sobre o modelo de licenciamento para ferramentas de automação de teste.

CT-TAS-2.1.3 (K2) Exemplificar os fatores a serem considerados ao definir uma estratégia de automação de teste.

2.2 Funções e Responsabilidades na Automação de Teste

CT-TAS-2.2.1 (K2) Resumir as funções e habilidades necessárias para uma solução de automação de teste bem-sucedida.

2.1 Custos e Riscos da implementação de uma Solução de Automação de Teste.

2.1.1 Comparar soluções técnicas alternativas com relação ao custo de propriedade.

Em termos de propriedade, uma abordagem comum é uma solução interna personalizada com base em ferramentas comerciais ou de código aberto. Outras abordagens alternativas incluem soluções comerciais não personalizáveis ou a assinatura de um contrato com uma empresa terceirizada para que o trabalho seja feito pelo TAE.

A criação de toda a TAS internamente garante que todos os custos, recursos, riscos e governança estejam dentro da organização (p. ex., membros da equipe/desenvolvedores e recursos de hardware e software necessários para a solução real). Um fator importante é que, ao seguir essa abordagem e alocar tempo para essa atividade, a TAS pode ser desenvolvida sem a necessidade de um contrato com um fornecedor ou empresa terceirizada, pois todos os TAE e seus conhecimentos estão disponíveis na organização. No entanto, para realizá-lo com sucesso, a organização precisa ter os TAE certos, contratados ou treinados, que possam conduzir o desenvolvimento da TAS.

Ao trabalhar com uma solução baseada em fornecedor, a propriedade será compartilhada entre o cliente e o fornecedor, dependendo dos detalhes do contrato. Se já houver uma ferramenta de teste na organização que já tenha sido testada e atenda a todos os requisitos, e se não houver necessidade de recursos adicionais nesta ferramenta, será mais fácil adotar essa abordagem. Os testadores da organização poderão trabalhar de forma produtiva depois de receberem o treinamento necessário do fornecedor e, normalmente, haverá especialistas no assunto (SME) designados como Product Owners dentro da organização. O risco dessa abordagem é que, se algum trabalho adicional precisar ser feito (p. ex., corrigir defeitos da ferramenta e solicitações de recursos adicionais) na ferramenta de teste, isso levará tempo e exigirá negociações com o fornecedor para que seja feito no prazo e da maneira correta, o que pode afetar o trabalho dos testadores que estão utilizando essa ferramenta de teste.

Se a organização não quiser ter a responsabilidade de montar uma equipe, a terceirização é uma solução recomendada. No caso de trabalhar com empresas terceirizadas, o cliente não precisa contratar ou comprar nenhum hardware, software ou recrutar funcionários com o conjunto de habilidades necessárias. Todo o trabalho e os custos adicionais ficarão com a empresa terceirizada, bem como a propriedade das ferramentas. Expectativas e métricas mensuráveis devem ser definidas no contrato para tornar o progresso transparente e visível. Essa abordagem é sugerida caso a organização não planeje investir esforços na contratação de TAE devido a prazos curtos de projeto, custos ou outras considerações.

2.1.2 Explicar as considerações sobre o modelo de licenciamento para ferramentas de automação de teste.

O licenciamento é um aspecto importante a ser considerado quando a automação de teste é estabelecida. Cada modelo de licenciamento mencionado abaixo tem um impacto diferente nos fatores de usabilidade e custo, por exemplo, custos de execução de testes e dificuldade de configurar o ambiente de desenvolvimento.

Os modelos de licenciamento incluem:

- **Código aberto**: Em muitos casos, as organizações estão usando ferramentas de código aberto para atingir seus objetivos de teste na automação de teste. O principal motivo para isso é que não há custos de licença nem taxas de manutenção para o uso das ferramentas de teste. Muitos usuários e organizações contribuem para o desenvolvimento de ferramentas de código aberto, o que facilita a coleta de informações e o recebimento de suporte. A maioria das licenças de código aberto também permite que as pessoas modifiquem as ferramentas, se necessário, ou até mesmo as republiquem sem restrições significativas.
- **Licença por usuário/máquina**: As ferramentas comerciais geralmente têm licenciamento por usuário(s) ou máquina(s). Em termos de custos, as ferramentas podem ser usadas com eficiência quando a organização sabe o número de TAE que trabalharão em um determinado projeto a longo prazo. Geralmente, quanto mais licenças uma organização solicita, melhor é o preço que recebe.

- **Licença flutuante**: Essa é uma licença que pode ser compartilhada entre várias pessoas na organização para ser usada em momentos diferentes. Ela pode ser muito útil quando há muitos TAE que trabalham em máquinas diferentes. O número de licenças é calculado com base no uso simultâneo, não no número total de TAE ou em máquinas específicas nas quais eles executam seus testes.
- **Licença de tempo de execução**: Muitas ferramentas comerciais têm licenças de tempo de execução para executar a automação de teste. Esse tipo de licença ocorre com fornecedores de nuvem que hospedam ferramentas de automação de teste, vários sistemas operacionais (SO) e versões de navegador como um serviço. Há também fornecedores de nuvem que fornecem acesso a fazendas de dispositivos que têm uma variedade de dispositivos móveis, plataformas, versões e redes de celular. Aqueles que utilizam esses serviços são cobrados apenas pelo tempo que usam para executar testes e, portanto, são cobrados com base em uma licença de tempo de execução.

2.1.3 Exemplificar os fatores a serem considerados ao definir uma estratégia de automação de teste.

Muitos fatores podem influenciar as decisões sobre a implementação da automação de teste e sua estratégia.

Os mais importantes são:

- Restrições de tempo;
- Nível necessário de especialização e número de TAE para desenvolver a TAS;
- Hardware de teste;
- Licenças de ferramentas de teste;
- Adaptabilidade;
- Manutenção;
- Suporte a diferentes plataformas (p. ex., web, desktop e/ou celular);
- Suporte à integração contínua/entrega contínua (CI/CD);
- Integração do gerenciamento de testes e relatórios de testes.

O primeiro e mais importante fator de custo é o cronograma do trabalho que deve ser feito da própria automação de teste.

Do ponto de vista do cronograma, se o prazo for curto, uma abordagem frequentemente usada é contratar apenas alguns TAE qualificados para criar a TAS. Em termos de um roteiro e cronograma mais longos, a organização pode decidir sobre o número de TAE necessário com mais contexto. Quando o SUT crescer, melhorar e se tornar mais complexo, pode-se tomar a decisão de contratar mais TAE para implementar e manter a TAS e os testes.

Outros fatores de custo importantes são as ferramentas e suas licenças. O orçamento deve conter as ferramentas de teste necessárias, o hardware e o treinamento adicional para os TAE. Esses fatores têm uma forte conexão com a integração com outras ferramentas e sistemas para realizar quaisquer funções adicionais que não sejam de teste, como carregar os resultados do teste no sistema de gerenciamento de testes ou acionar uma compilação no sistema de gerenciamento de configuração. Para essas funções adicionais, há ferramentas, gratuitas ou pagas, e bibliotecas a serem usadas, mas elas devem ser consideradas durante a criação da estratégia de automação de teste e devem ser incluídas no orçamento.

A quantidade de ambientes de teste e agentes que uma equipe de desenvolvimento tem depende sempre do contexto. Quanto maior e mais complexo for o SUT e o cronograma de lançamento, mais recursos serão necessários para a organização, o que também aumentará o custo. Uma abordagem recente para limitar os custos de recursos é usar provedores de nuvem e pagar por recursos de hardware de teste sob demanda e licenças de tempo de execução, que devem ser usados com cuidado, pois podem ter um efeito contrário. Às vezes, as máquinas podem ser deixadas em execução e se tornar mais caras do que em seu próprio hardware. Essa abordagem pode reduzir os altos custos de manutenção e operação para a equipe da TAE. equipe.

2.2 Funções e Responsabilidades na Automação de Teste

2.2.1 Resumir as funções e habilidades necessárias para uma solução de automação de teste bem-sucedida.

Para uma TAS bem-sucedida, as organizações exigem TAE qualificados, que devem ter sólidos conhecimentos técnicos de programação e arquitetura.

Dependendo do tamanho e da maturidade do projeto, também deve haver pelo menos um especialista forte (p. ex., um líder de teste, um arquiteto ou um analista de negócios) que entenda o domínio real dos negócios e os objetivos do teste. A função dessa pessoa é ajudar a conduzir e criar o conceito, com base nos objetivos do teste, e um roteiro para a TAS real. O gerenciamento da equipe e as habilidades interpessoais serão necessários para treinar, motivar e formar a equipe para o trabalho real [CTEL-TM-MTT].

Um TAE deve ter habilidades e conhecimentos técnicos sólidos sobre diferentes SDLCs e sobre a arquitetura da SUT e seu ambiente de desenvolvimento. Além dos aspectos técnicos, o TAE deve ter a capacidade de cooperar com analistas de teste e outros stakeholders sobre os objetivos da automação de teste. Como não é possível realizar testes exaustivos, ele se aplica à automação de teste: a cobertura de 100% da automação de teste não pode ser alcançada. Como o tempo e o esforço são limitados, os TAE precisam ser capazes de priorizar as condições de teste mais impactantes a serem cobertas a partir de uma perspectiva de negócio e de investimento, contribuindo para as avaliações de risco.

3 Preparando-se para a Automação de Teste (225min)

Palavras-chave

teste de API, componentes testes de contrato, Shift Right, Shift Left, sistema sob teste, abordagem de automação de teste, condição de teste, nível de teste, pirâmide de teste, dublê de teste

Objetivos de Aprendizagem:

3.1 Integração entre os níveis de teste

CT-TAS-3.1.1 (K2) Diferenciar as distribuições de automação de teste.

CT-TAS-3.1.2 (K2) Selecionar uma abordagem de automação de teste com base na arquitetura do sistema em teste.

CT-TAS-3.1.3 (K3) Demonstrar maneiras de otimizar a distribuição da automação de teste para alcançar as abordagens Shift Left e Shift Right.

3.2 Considerações estratégicas em diferentes modelos de ciclo de vida de desenvolvimento de software.

CT-TAS-3.2.1 (K2) Explicar como os projetos de automação de teste estão em conformidade com os modelos de ciclo de vida de desenvolvimento de software legado.

CT-TAS-3.2.2 (K2) Explicar como os projetos de automação de teste estão em conformidade com as práticas recomendadas de desenvolvimento de software ágil que dão suporte à automação de teste.

CT-TAS-3.2.3 (K3) Preparar-se para projetos de automação de teste em conformidade com as práticas recomendadas de DevOps que suportam a automação de teste em testes contínuos.

3.3 Aplicabilidade e viabilidade da automação de teste

CT-TAS-3.3.1 (K2) Explicar os critérios para determinar a adequação dos testes para a automação de teste.

CT-TAS-3.3.2 (K2) Identificar desafios que somente a automação de teste pode resolver.

CT-TAS-3.3.3 (K2) Identificar as condições de teste que são difíceis de automatizar.

3.1 Integração entre os Níveis de Teste

3.1.1 Diferenciar as distribuições de automação de teste

Mike Cohn criou o conceito original de uma pirâmide de automação de teste que descreve três níveis: unidade (o ISTQB chama de componente), serviço e interface do usuário. Desde então, surgiram muitas variações, e todas elas compartilham os mesmos princípios básicos de descrição da função dos níveis de teste e da distribuição dos casos de teste entre esses níveis (consulte o syllabus ISTQB CTFL, seção 5.1.6 Pirâmide de teste).

O foco da pirâmide de automação de teste original de Mike Cohn está em quais tipos de teste são executados pela TAS. O nível de serviço pode ser dividido em três tipos de teste: teste de integração de componentes, teste de contrato e teste de API.

Teste de unidade (o ISTQB chama de teste de componente) destina-se à validação de componentes individuais, com foco na qualidade do código. O teste de integração de componentes permite a validação da interface do usuário (UI) e da API, aproveitando os dublês de teste, como simuladores e emuladores. O teste de contrato permite a validação dos contratos entre os serviços. O teste de API concentra-se na validação funcional de determinados serviços com dados reais por meio de conexões de serviços reais. O teste de UI é um teste de ponta a ponta de um sistema, interagindo com sua GUI.

É útil desenhar o estado atual dos testes como uma baseline e o estado desejado. Isso proporciona uma compreensão clara do que está faltando ou do que é um nível aceitável de testes. Isso indicará a quantidade de testes realizados em cada nível de teste e se o teste é manual ou automatizado. O estado-alvo também depende do cronograma e do que pode ser alcançado dentro desse período de tempo. Se for possível alcançá-lo, a forma de pirâmide deve ser a forma do alvo.

Exemplos de distribuições de teste:

- **Pirâmide**: Essa é uma distribuição equilibrada de testes, com menos testes realizados nos níveis de teste mais altos e mais nos níveis de teste mais baixos, com testes estáveis e mais rápidos. Se os recursos disponíveis e o prazo permitirem, essa costuma ser a pirâmide de estado-alvo.
- **Cone de sorvete**: Esta é a versão invertida da pirâmide. Há uma quantidade equilibrada de testes de serviço, mas os testes dependem muito de encontrar a maioria dos defeitos no nível de teste da interface do usuário que, normalmente, é mais caro de automatizar devido à sua complexidade. Devido à falta de testes de componentes, os defeitos são encontrados mais tarde no SDLC.
- **Ampulheta**: Os testes são pesados nos níveis de teste mais altos e mais baixos, e os testes de nível de serviço estão ausentes em sua maioria, resultando em defeitos de integração. Se a lógica de negócio for fornecida por API (ou seja, serviços), muitos dos testes de interface do usuário poderão ser facilmente transferidos para níveis de teste mais baixos.
- **Guarda-chuva**: O teste é totalmente dependente dos caros testes de interface do usuário. Isso resulta em um retorno lento dos defeitos, manutenção dispendiosa dos casos de teste e da TAS. Se não for tecnicamente possível implementar casos de teste de nível inferior, talvez não seja possível abandonar a forma de guarda-chuva, e o foco deve ser a otimização do conjunto de automação de teste de interface do usuário, a redução do tempo de execução dos testes e o aumento da estabilidade.

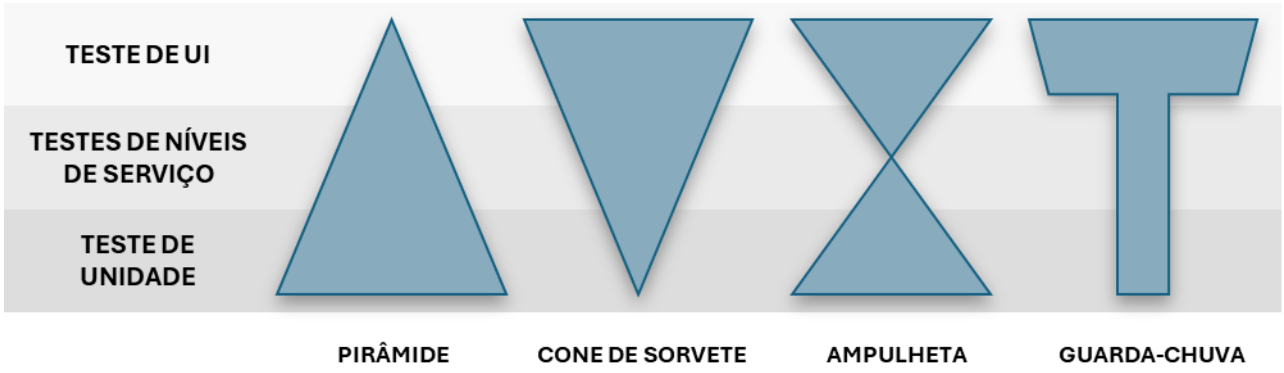

Figura 1: Exemplos de distribuições de teste

3.1.2 Selecionar uma abordagem de automação de teste com base na arquitetura do sistema em teste

Há muitas maneiras de definir os níveis de teste em uma estratégia de teste. A escolha de uma delas depende muito de vários fatores, como a cultura da organização, a maturidade geral da engenharia de software, o SDLC seguido e se o SUT tem uma arquitetura de mainframe ou de micros serviços.

Embora a forma de pirâmide seja considerada a distribuição de teste ideal, nem sempre é possível alcançá-la ou leva mais tempo para fazer a transição para esse estado final. É uma boa prática ter uma forma realista como estado de destino, como a transformação da forma de guarda-chuva para a de ampulheta. Quando isso for alcançado, um novo estado-alvo poderá ser determinado.

A escolha da distribuição apropriada de automação de teste para sistemas de mainframe é diferente do desenvolvimento de software moderno e, mais uma vez, depende de muitos fatores, pois talvez não seja possível automatizar alguns dos níveis. O acesso aos mainframes é tradicionalmente feito por meio de emulação de terminal (ou seja, “telas verdes”). A validação dos processos em lote é possível, mas limitada. O teste de componentes é mais difícil de ser realizado. Como há poucos ou nenhum micro serviços presente, nem sempre é uma opção validar a comunicação de dados entre interfaces com testes de API. O teste por meio de trabalhos em lote, bancos de dados e uma GUI é mais predominante.

As organizações que estão modernizando suas soluções legadas, mudando lentamente para uma arquitetura de micro serviços, podem introduzir o teste de API e testes de contrato para as partes modernizadas do sistema.

3.1.3 Demonstrar maneiras de otimizar a distribuição da automação de teste para alcançar as abordagens Shift Left e Shift Right

Depois que a distribuição do estado atual for identificada e a distribuição do estado de destino for selecionada, um roteiro de melhorias poderá ser planejado. Isso determina o que (ou seja, o escopo) e como testar (ou seja, a estratégia) com a automação de teste. É necessário determinar um backlog priorizado de itens a serem implementados e os critérios de entrada para a automação de teste.

No caso de uma distribuição desequilibrada da automação de teste, recomenda-se adotar uma abordagem *bottom-up*. Se a cobertura do código for baixa, isso é um sinal de falta de casos de teste de componentes e casos de teste de componentes adicionais precisam ser escritos. Em uma segunda etapa, a qualidade dos testes de componentes precisa ser revisada e, se necessário, aprimorada. Sugerir práticas recomendadas, seguir uma técnica de desenvolvimento orientada a testes e realizar workshops sobre técnicas de teste pode melhorar a qualidade da automação de teste.

Apresentando o componente de testes para interface do usuário e de API, o aproveitamento de dublês de teste (p. ex., emuladores, simuladores) ajuda a obter um *Shift Left* de testes caros, lentos e não confiáveis. A remoção da dependência de serviços e dados reais melhora a consistência da execução dos testes e fornece feedback antecipado que é fácil de integrá-los ao pipeline de CI/CD.

Nos últimos anos, os testes de contrato têm se tornado cada vez mais proeminentes. Ao validar o contrato entre um provedor e um consumidor, é mais fácil encontrar as causas básicas dos defeitos mais cedo. As equipes não dependerão de testes de API nem de testes de interface do usuário para detectar defeitos de serviços subjacentes e, em vez disso, poderão reduzir o número desses casos de teste. Criar um gráfico de chamadas para APIs é uma maneira inteligente de rastrear quais APIs estão conectadas umas às outras e quais são os consumidores ou produtores de uma API selecionada. Isso permite uma melhor identificação dos possíveis pontos problemáticos de um sistema.

Enquanto o *Shift Left* é uma abordagem que envolve mover os testes mais cedo no SDLC, o *Shift Right* move os testes mais tarde, quando o SUT tiver sido lançado, como um meio de avaliar a performance em um ambiente de teste de pré-produção ou em produção. Ao aplicar o *Shift Right*, os testadores podem monitorar a performance do aplicativo e da API enquanto obtêm feedback dos usuários reais. Embora a automação de teste seja mais proeminente em uma abordagem de *Shift Left*, se for combinada com a observação, a automação de teste poderá ajudar a tomar decisões mais rápidas sobre se uma versão completa pode ser lançada ou se a versão precisa ser revertida.

O objetivo do teste *Shift Right* é:

- Entender as preferências do usuário;
- Suportar lançamentos *Canarys*[1], e *Dark Launches*[2] para interromper minimamente a funcionalidade do usuário;
- Identificar defeitos na produção com antecedência;
- Ajudar a expandir o escopo e o uso da automação de teste;
- Aumentar a cobertura.

3.2 Considerações estratégicas em diferentes Modelos de Ciclo de Vida de Desenvolvimento de Software.

3.2.1 Explicar como os projetos de automação de teste estão em conformidade com o modelo de ciclo de vida de desenvolvimento de software legado.

No modelo em cascata, a fase de teste vem depois das fases de análise de requisitos, modelagem do sistema e implementação. Devido a essa ordem estrita, as atividades de automação de teste começam mais tarde. Ciclos mais longos entre os testes significam menos oportunidades de aproveitar a automação de teste, e o feedback da automação de teste geralmente chega tarde demais, resultando em um menor retorno sobre o investimento (ROI).

No modelo V, todo o planejamento do teste, juntamente com a preparação da documentação, ocorre nas fases iniciais. Por exemplo, a fase de projeto de arquitetura produz o projeto de teste de integração, que define o teste mais cedo do que no modelo em cascata. Infelizmente, a implementação real de um TAS ainda ocorre mais tarde no SDLC e, embora o ROI da automação de teste seja maior em comparação com o modelo em cascata, ele ainda fica atrás das práticas modernas de desenvolvimento de software ágil.

3.2.2 Explicar como os projetos de automação de teste estão em conformidade com as práticas recomendadas de desenvolvimento ágil de software que dão suporte à automação de teste.

Um dos ideais do desenvolvimento ágil de software é alcançar a automação de teste *in-sprint*. Isso significa determinar todas as atividades de automação de teste necessárias como parte dos critérios de saída para cada uma das histórias de usuário. Isso inclui as definições de casos de teste, a implementação de casos de teste automatizados, atualizações para o TAF e, em alguns casos, a integração com um pipeline de CI/CD. As organizações que não aplicam corretamente as práticas ágeis não incluem uma estimativa do esforço de teste e as administram em um registro separado ou não as monitoram de forma alguma. Ao obter a automação de teste *in-sprint*, as equipes ágeis garantem que estão prontas para implantar o escopo acordado dentro ou até o final de cada sprint. Se uma equipe não estiver madura do ponto de vista ágil, então a primeira meta deve ser a realização de testes *in-sprint*, enquanto a automação ficaria atrasada em um sprint. A partir daí, a equipe pode trabalhar para alcançar a automação de teste *in-sprint*.

[1] *Canary Release*: envolve liberar uma nova funcionalidade ou atualização para um pequeno subconjunto de usuários antes de disponibilizá-la para todo o conjunto de usuários. Esse subconjunto é frequentemente chamado de “grupo canary”. (BSTQB)

[2] *Dark Launches*: envolve liberar uma nova funcionalidade ou atualização para todo o conjunto de usuários, mas mantê-la oculta. Em vez de estar disponível para todos, a nova funcionalidade é seletivamente ativada para um pequeno grupo de usuários ou sob condições específicas. (BSTQB)

3.2.3 Preparar-se para projetos de automação de teste em conformidade com as práticas recomendadas de DevOps que suportam a automação de teste em testes contínuos.

O desenvolvimento ágil de software se concentra em como o trabalho é organizado, enquanto o DevOps é responsável pela entrega de software de ponta a ponta. Isso é obtido por meio da automação das atividades de criação, integração, teste, implantação e produção. Isso facilita o teste contínuo por meio de ciclos de feedback que garantem a melhoria contínua.

A ênfase está mais na implementação de casos de teste automatizados de nível inferior, incluindo testes de componentes, componentes integração e testes de contrato. Reduzindo a dependência de dados e serviços reais, os testes serão executados em um tempo mais curto. Se for viável, a automação de teste deve ser executada no mesmo pipeline em que o SUT é criado. Diferentes conjuntos de testes são acionados após cada fase de desenvolvimento e construção (p. ex., local, solicitação “puxada”, mesclagem e implantação).

Se os testadores tiverem a capacidade de criar conjuntos maiores de testes de interface do usuário e de API, eles serão executados separadamente para fornecer valor adicional. Sugere-se que os esforços de teste manual se concentrem em testes exploratórios e comentários do usuário final, como uma atividade complementar à automação de teste.

3.3 Aplicabilidade e Viabilidade da Automação de Teste

3.3.1 Explicar os critérios para determinar a adequação dos testes para a automação de teste.

A seleção de casos de teste para automação de teste geralmente é feita por um analista de testes que entende quais casos de teste podem ser automatizados ou por um TAE que tenha o conhecimento de domínio necessário para tomar essas decisões.

Os itens a seguir são considerados na seleção e priorização de casos de teste para automação de teste:

- É tecnicamente possível implementar os casos de teste de forma automatizada?
- Existem desafios técnicos que afetam a entrega de casos de teste automatizados? A equipe está preparada e bem treinada para fazer o trabalho de implementação?
- O esforço de codificação proporciona um ROI adequado? (consulte a seção 5.1.1)
- Há algum valor em executar os casos de teste com frequência?
- É um teste funcional ou não funcional? Ele faz parte do conjunto de *smoke test*, suíte de testes de regressão ou da suíte de testes de confirmação?
- O caso de teste é repetível?
- O caso de teste é fácil de manter quando o SUT muda devido a atualizações?
- O caso de teste abrange os fluxos de trabalho comerciais usados com frequência?
- Existe uma sobreposição funcional entre os testes que permite a reutilização das etapas e dos dados de teste?

3.3.2 Identificar os desafios que somente a automação de teste pode resolver.

Há certos testes que só podem ser realizados com a automação de teste. Isso inclui categorias quando:

1. O tempo de execução do teste manual é maior do que o adequado;
2. A execução dos casos de teste precisa ser sincronizada;
3. Os resultados dos testes precisam estar disponíveis em um pipeline;
4. Arquivos de registro grandes precisam ser analisados quanto a defeitos;
5. É necessária precisão no tempo dos testes;
6. As permutações de teste são necessárias em vários sistemas operacionais, navegadores, dispositivos, locais ou configurações;

7. É necessário um grande volume de execuções de teste e/ou entrada de dados para maximizar a cobertura;
8. Qualquer teste não funcional que exija monitoramento e análise automatizados ou entrada de uma grande quantidade de usuários, como teste de estresse ou teste de confiabilidade.

3.3.3 Identificar condições de teste que são difíceis de automatizar

Há certas condições de teste que tornam a automação dos casos de teste um desafio difícil ou até mesmo uma tarefa impossível. Há muitos exemplos da vida real. Alguns deles estão listados abaixo:

- Validação dos requisitos de design, incluindo a consistência da interface do usuário em diferentes plataformas. A aparência geral do software é subjetiva e exige uma análise humana.
- Qualquer caso de teste que envolva muita interação humana. No setor financeiro, um bom exemplo seria um pedido de empréstimo. Nesse processo, pode haver casos de determinados regulamentos ou condições (p. ex., renda insuficiente e detalhes pessoais incorretos). Nessas situações, os agentes precisam verificar novamente o pedido de empréstimo real e tomar decisões manuais ou entrar em contato com o cliente.
- Dificuldades técnicas que bloqueiam as atividades de automação de teste, como restrições no nível do sistema operacional. Um exemplo é o software nativo do sistema operacional que envia mensagens de texto para os usuários. As interações ou validações dessas mensagens de texto não são possíveis com as ferramentas de automação de teste da interface do usuário, pois o sistema operacional não permite interações fora da SUT de destino.
- As condições de teste com uma longa dependência de tempo tornariam a automação de teste ineficiente e, muitas vezes, difícil de ser configurada adequadamente em comparação com a execução manual de testes. Por exemplo, o testador precisa fazer login no SUT, atualizar o conteúdo da página inicial e esperar duas horas para que o SUT faça logout automaticamente devido a um tempo limite da sessão. Se o tempo decorrido não puder ser alterado de nenhuma forma manual (p. ex., simulações, respostas do lado do servidor e alterações de tempo no nível do sistema operacional), não é recomendável implementar esses casos de teste de forma automatizada.

4 Estratégias Organizacionais de Implantação e Liberação para Automação de Teste (135min)

Palavras-chave

componente, teste de confirmação, controle de qualidade, marcos da qualidade

Objetivos de Aprendizagem:

4.1 Planejamento da solução de automação de teste.

CT-TAS-4.1.1 (K2) Identificar maneiras pelas quais a automação de teste ajuda a reduzir o tempo de lançamento no mercado.

CT-TAS-4.1.2 (K2) Identificar maneiras pelas quais a automação de teste ajuda a verificar os defeitos relatados de acordo com os requisitos.

CT-TAS-4.1.3 (K2) Definir abordagens que permitam o desenvolvimento de cenários operacionalmente relevantes para a automação de teste.

4.2 Estratégias de implantação

CT-TAS-4.2.1 (K2) Definir uma estratégia de implantação de automação de teste.

CT-TAS-4.2.2 (K2) Identificar os riscos da automação de teste na implantação.

CT-TAS-4.2.3 (K2) Definir abordagens para mitigar os riscos da implantação.

4.3 Dependências no ambiente de teste.

CT-TAS-4.3.1 (K2) Definir componentes de automação de teste no ambiente de teste.

CT-TAS-4.3.2 (K2) Identificar os componentes da infraestrutura e as dependências da automação de teste.

CT-TAS-4.3.3 (K2) Definir dados de automação de teste e requisitos de interface.

4.1 Planejamento de Soluções de Automação de Teste.

4.1.1 Identificar maneiras pelas quais a automação de teste ajuda a reduzir o tempo de lançamento no mercado.

A automação de teste ajuda a colocar o software no mercado mais rapidamente porque ajuda a reduzir o tempo do ciclo de teste. Os testes realizados antes do lançamento de um software exigem uma quantidade considerável de verificação e validação. Isso pode ser particularmente significativo durante os testes de regressão, pois esse tipo de teste promove a reutilização, reduzindo os custos e proporcionando consistência entre as execuções de teste.

A automação de teste ajuda a reduzir o esforço de testes manuais e fornece feedback rápido aos desenvolvedores, abrangendo o mesmo escopo de testes. Ela também permite a realização de testes no início do ciclo de vida de desenvolvimento do software se os testes de componentes e de integração de componentes e os testes de integração de componentes forem automatizados, o que normalmente não é feito de forma manual.

Um marco da qualidade é uma medida obrigatória incorporada ao seu processo que o software precisa cumprir antes de passar para a próxima fase. A configuração de marcos da qualidade com base na automação de teste permite um processo de implantação acelerado em um ambiente de pré-produção ou produção. Ao aproveitar esses marcos da qualidade, os defeitos podem ser encontrados mais cedo, o que faz com que o tempo de lançamento no mercado seja menor. O tempo de execução do teste pode ser reduzido com a execução de testes paralelos e com a abordagem *Shift Left*. A abordagem *Shift Left* promove uma cultura de consciência da qualidade e incentiva a realização de testes no início do ciclo de vida de desenvolvimento do software. Isso pode incluir vários casos de teste independentes ou testes entre navegadores.

Para obter informações adicionais, consulte as seções 3.1.3, 6.1.2 e 6.2.1.

4.1.2 Identificar as formas em que a automatização dos testes ajuda a verificar os defeitos comunicados de acordo com os requisitos.

O teste de confirmação realizado após uma correção de código pode abordar um defeito relatado. Normalmente, um testador segue as etapas de teste necessárias para replicar o defeito e verificar se o defeito não existe mais.

Os defeitos têm uma maneira de se reintroduzir nas versões subsequentes (p. ex., isso pode indicar um problema de gerenciamento de configuração ou de repositório de código) e, portanto, os testes de confirmação são candidatos adequados para a automação de teste e podem ser adicionados à suíte de testes de regressão existente.

Um teste de confirmação automatizado normalmente tem um escopo restrito de funcionalidade. A implementação pode ocorrer a qualquer momento depois que um defeito é relatado e as etapas de teste necessárias para replicá-lo são compreendidas.

O rastreamento de testes de confirmação automatizados permite informar o tempo e o número de ciclos de teste gastos na resolução de defeitos.

Ao verificar as correções em várias plataformas, dispositivos, navegadores e versões de sistemas operacionais com a automação de teste, o tempo gasto com os testes é bastante reduzido.

4.1.3 Definir abordagens que permitam o desenvolvimento de cenários operacionalmente relevantes para a automação de teste

O teste de aceite operacional garante a prontidão do software para os sistemas de produção e, normalmente, é realizado logo antes de uma liberação. Esse esforço tem o objetivo de fazer uma validação final dos sistemas, componentes e outras infraestruturas do SUT e testar sua prontidão para a produção. Isso é necessário porque, apesar dos melhores esforços de um TAE não há garantia de que o SUT se comportará da mesma forma fora do ambiente de teste e na produção.

Um bom plano de teste operacional incluirá testes de confiabilidade, testes de tolerância a falhas, integridade e capacidade de manutenção. Abaixo estão as abordagens de automação de teste que podem ser usadas:

- **Análise de código estático**: Ferramentas automatizadas que analisam o código quanto à segurança e às vulnerabilidades. Os resultados dos testes são armazenados para análise de tendências ao longo do tempo.
- **Teste end-to-end**: Scripts de teste automatizados que conduzem cenários de usuário de ponta a ponta e testam todos os aspectos do ambiente para descobrir quaisquer defeitos.
- **Teste de failover**: Scripts de teste automatizados que testam especificamente o que acontece quando o hardware de um aplicativo fica off-line. Alguns exemplos incluem como o aplicativo se recupera quando servidores físicos, servidores em nuvem, redes, discos de computador e outros componentes de hardware apresentam mau funcionamento. Os scripts de teste são projetados para medir o sucesso ou a falha da capacidade de recuperação automática do SUT, o que é particularmente importante para as organizações que implementam a engenharia do caos.
- **Teste de backup e restauração**: Scripts de teste automatizados que testam o sucesso de fazer um backup da versão atual e, em seguida, reverter para um ponto anterior (ou seja, uma versão mais antiga do software) .
- **Teste de eficiência de performance** (p. ex., teste de carga): Scripts de teste automatizados que podem ser usados para simular muitos usuários testando o SUT de uma só vez. Os resultados dos testes são armazenados para comparação e têm tendências ao longo do tempo.
- **Revisão da documentação operacional**: Scripts de teste automatizados podem ser usados para essa atividade para comparar versões da documentação do SUT e sinalizar para as equipes de desenvolvimento se é necessária uma atualização com base nos novos recursos adicionados a uma determinada versão.
- **Teste de segurança**: Scripts de teste automatizados podem ser criados em combinação com ferramentas padronizadas de teste de segurança para avaliar o SUT, de forma estática e dinâmica. Os resultados dos testes são armazenados ao longo do tempo para comparação e análise de tendências.
- **Monitoramento com base no acordo de nível de serviço de uma organização**: Os scripts de teste automatizados usados durante o SDLC podem ser reaproveitados para monitorar as operações de produção e enviar alertas se um teste automatizado falhar. Essa é uma medida proativa para detectar interrupções na produção antes que os usuários reais passem por elas.

A automação de teste para validação de software operacional pode proporcionar imensos benefícios, especialmente se os testes automatizados puderem ser executados repetidamente e em vários ambientes. Podem ser criados conjuntos de testes automatizados dedicados para que os resultados dos testes possam ser gerados e comparados com as execuções de testes anteriores. Isso garante a consistência quando novas versões do SUT são lançadas. Uma suíte de testes automatizada, confiável, de condições de teste operacionais específicas, pode reduzir os custos ao longo do tempo.

4.2 Estratégias de Implantação.

4.2.1 Definir uma estratégia de implantação de automação de teste.

Uma boa estratégia de implantação de automação de teste levará em conta, entre outros aspectos, o ambiente geral de teste, as ferramentas de teste disponíveis, o acesso à SUT, o armazenamento de scripts de teste e outras dependências, além do provisionamento de dados de teste. Com essas considerações de alto nível em mente, um TAE pode começar a pensar em uma estratégia para desenvolver e implantar a TAS.

Ambiente de teste: Os TAE devem considerar como acessarão o SUT nos diferentes ambientes de teste. Quando começarem a desenvolver seu testware de automação de teste, quer estejam usando uma abordagem orientada por palavras-chave ou outra solução, devem considerar como essa solução será executada em vários ambientes de teste. Por exemplo, um script de teste deve ser desenvolvido de forma que possa ser executado em um ambiente de teste e em um ambiente de pré-produção com o mínimo de alterações. Normalmente, trata-se de alterar uma URL de um aplicativo baseado na Web e, em seguida, o script de teste é executado da mesma forma, independentemente do ambiente de teste em que se encontra.

Ferramentas: Os TAE também precisarão considerar quais ferramentas estão usando para criar a TAS. Se for uma ferramenta comercial, é provável que eles precisem entender como ela é licenciada. O TAE pode descobrir

que seu ambiente de teste precisa ter acesso ao servidor de licenças da ferramenta. Isso precisa ser considerado quando o script de teste se destina a ser usado em vários ambientes de teste. O fato de o servidor de licenciamento estar disponível no ambiente de teste não significa necessariamente que ele estará disponível no ambiente de pré-produção.

Acesso ao software: considerações importantes precisam ser entendidas sobre como acessar o SUT. Os scripts de teste podem ser modelados para aceitar parâmetros de modo que usuários específicos ou credenciais com acesso especial possam ser rapidamente atualizados quando os pontos finais do SUT forem alterados. Mais uma vez, isso se torna muito importante quando se muda para ambientes de teste diferentes e o URL precisa ser alterado. Além disso, pode ser necessário usar credenciais diferentes (p. ex., por meio de contas de usuário e de teste, biometria e cartões inteligentes), dependendo do ambiente de pré-produção. Um bom projeto de script de teste automatizado incluirá flexibilidade suficiente para que parâmetros simples possam ser definidos e os scripts de teste sejam executados conforme o esperado, independentemente do ambiente de teste.

Armazenamento de scripts de teste: O TAE deverá escolher um local central para armazenar e gerenciar os scripts de teste automatizados. Uma boa estratégia seria incorporar um repositório de código-fonte que utilize o gerenciamento de configuração. Desta forma, os scripts de teste podem ser acessíveis a partir de vários ambientes de teste, desde que tenham acesso ao repositório de código-fonte e as versões possam ser criadas para a versão específica do SUT. Todas as configurações e dependências podem ser salvas no mesmo repositório que os scripts de teste e oferecem portabilidade ideal. Em resumo, a TAS, o TAF e todos os casos de teste podem ser armazenados e gerenciados em repositórios.

Fornecimento de dados: Será importante entender se um script de teste automatizado depende de determinados dados de teste já existentes no ambiente de teste em que o SUT está sendo testado. Um bom projeto de script de teste evitará dados estáticos (ou seja, conjuntos de dados fixos) o máximo possível. Entretanto, haverá situações em que isso não será viável. Nesses casos, o TAE precisará determinar uma solução para ter os dados de teste pré-carregados ou usar a automação de teste para criar scripts de teste de configuração que gerem os dados de teste necessários antes que os scripts de teste reais sejam executados e testem o SUT. Há prós e contras em ambas as abordagens. Por um lado, é conveniente que um administrador faça o pré-carregamento dos dados de teste. No entanto, o TAE depende de outro recurso para ter disponibilidade de suporte. Ser autossuficiente com um script de teste de configuração elimina a necessidade de depender de outro recurso. No entanto, a criação desses scripts de teste leva tempo e se torna outra parte da solução que precisa ser mantida.

Ao levar em conta todos os itens mencionados acima, a estratégia de automação de teste poderá fornecer o planejamento e o controle adequados das atividades de automação de teste.

4.2.2 Identificar os riscos da automação de teste na implantação.

Os problemas técnicos podem levar a riscos de produto e riscos de projeto (para obter mais informações, consulte o syllabus CTFL, seção 5.2.2). Os problemas técnicos típicos incluem:

- O excesso de abstrações pode dificultar a compreensão do que o código de automação de teste está realmente fazendo (p. ex., com palavras-chave na abordagem orientada por palavras-chave).
- As tabelas de dados de teste podem se tornar muito grandes/complexas/incômodas para serem migradas para outros ambientes de teste, resultando em resultados de status inconsistentes.
- Dependência da TAS para usar determinadas bibliotecas do sistema operacional ou outros componentes que podem não estar disponíveis em todos os ambientes de teste do SUT.

Os riscos típicos do projeto de implementação incluem:

- Problemas de pessoal: pode ser difícil conseguir as pessoas certas para manter a automação de teste.
- Manutenção da TAS não planejada devido a atualizações do SUT que fazem com que a TAS opere incorretamente.
- Atrasos na introdução da automação de teste.
- Atrasos na atualização da TAS com base nas alterações feitas no SUT.
- A TAS não pode capturar objetos da interface do usuário não padrão.
- Permitir que casos de teste desatualizados permaneçam nas suítes de teste, desperdiçando tempo de execução do teste.

Os possíveis pontos de falha do projeto da TAS incluem:

- Migração para um ambiente de teste diferente.
- Implementação em um ambiente de produção.
- Esquecer que a automação é um software que também deve ser testado.

É importante perceber que vários problemas técnicos, riscos de projeto e possíveis pontos de falha podem comprometer o sucesso dos projetos de automação de teste. Os problemas técnicos comuns incluem complexidades decorrentes de abstrações excessivas, desafios com o gerenciamento de dados de teste e dependências de componentes específicos. Os riscos do projeto também precisam levar em conta as dificuldades de contratação de pessoal, problemas de manutenção devido a atualizações do sistema, atrasos na implantação e a presença de casos de teste desatualizados. Além disso, os possíveis pontos de falha, como a migração para ambientes diferentes e a negligência da necessidade de testar o próprio software de automação, devem ser abordados com o planejamento adequado. De modo geral, o monitoramento e as medidas proativas garantirão o sucesso dos projetos de automação de teste.

4.2.3 Definir abordagens para mitigar os riscos da implantação.

Há muitas estratégias de mitigação de risco que podem ser empregadas para lidar com essas áreas de risco. Algumas delas são discutidas a seguir.

A TAS tem um SDLC próprio, seja ele desenvolvido internamente ou uma solução adquirida. Um aspecto a ser lembrado é que a TAS, como qualquer outro software, precisa estar sob gerenciamento de configuração e ter seus recursos documentados. Caso contrário, será extremamente difícil implementar diferentes partes dele e fazê-las funcionar juntas ou em determinados ambientes de teste.

Além disso, é preciso haver um procedimento de implementação documentado, claro e fácil de seguir. Esse procedimento depende da versão, portanto, também deve ser incluído no gerenciamento de configuração.

Há dois casos distintos na implantação de um TAS:

- Primeira implantação.
- Implementação de manutenção - a TAS já existe, e uma atualização precisa ser implementada.

Os riscos relacionados à primeira implantação incluem:

- O tempo total de execução da suíte de testes pode ser maior do que o tempo planejado de execução do teste para o ciclo de teste. Nesse caso, é importante certificar-se de planejar tempo suficiente para que a suíte de testes seja executada em sua totalidade antes do início do próximo ciclo de testes programado.
- Existem problemas de instalação e configuração com ambientes de teste (p. ex., instalação e carga inicial do banco de dados e início/parada de serviços). A TAS precisa de um dispositivo de teste (ou seja, um conjunto de dados predefinido) para criar as condições prévias necessárias para que os casos de teste automatizados sejam executados no ambiente de teste.

Para implementações de manutenção, há considerações adicionais. A TAS em si precisa evoluir e as atualizações para ele precisam ser implementadas na produção. Antes de implementar uma versão atualizada da TAS na produção, ela precisa ser testada. Portanto, é necessário verificar a nova funcionalidade, para verificar se uma suíte de testes pode ser executada na TAS atualizada, que os relatórios de teste podem ser enviados e que não há defeitos de performance ou outros problemas de qualidade. Em alguns casos, toda a suíte de testes pode precisar de alteração para se adequar à versão mais recente da TAS.

4.3 Dependências no Ambiente de Teste.

4.3.1 Definir os componentes de automação de teste no ambiente de teste.

Os componentes de automação de teste geralmente consistem em ferramentas, máquinas virtuais, scripts de teste de automação, contêineres e configurações. O ambiente de teste também inclui o SUT. Os componentes de automação de teste no ambiente de teste podem ser definidos da seguinte forma:

SUT: Essa é uma parte óbvia do ambiente de teste. O SUT pode ser testado como um todo ou dividido em subcomponentes (p. ex., API, interface baseada na Web e banco de dados).

Plataforma: A plataforma descreve onde os componentes de automação de teste estão hospedados. Isso inclui a infraestrutura de nuvem, a rede, as máquinas virtuais e os contêineres que podem ser usados para tornar a automação de teste eficiente e portátil.

Casos de teste e suíte de teste: descrevem casos de teste individuais que são compostos de etapas de teste que direcionam ações automatizadas e manuais. As suítes de teste descrevem um agrupamento lógico de casos de teste para que possam ser executados juntos de forma eficiente.

Ferramentas: Diferentes ferramentas de automação de teste são usadas por vários motivos. Uma coleção de ferramentas incluiria aquelas para automação de teste de interface do usuário, teste de pontos de extremidade de API, geração de dados, monitoramento, gerenciamento de requisitos, gerenciamento de defeitos, ferramentas de registro e relatório e ferramentas para gerar tendências com base em métricas.

TAF: Isso inclui todos os itens que compõem o desenho do TAF. Os TAF incluem drivers de scripts driver, bibliotecas comuns, modelos para casos de teste automatizados, scripts de carga/ armazenamento de dados, documentação sobre como usar os componentes do TAF e tutoriais que ajudam os TAE a utilizar o TAF. Para obter mais informações, consulte o syllabus CTAL-TAE, seção 3.1.

A manutenção dos componentes de teste é uma consideração importante, pois complicar demais o ambiente de teste pode resultar em horas de tempo indesejado para corrigir defeitos no TAF em vez de se beneficiar da solução. É importante encontrar a combinação certa de ferramentas, configuração e portabilidade de plataforma para tornar os componentes o mais reutilizáveis possível.

4.3.2 Identificar os componentes da infraestrutura e as dependências da automação de teste.

Há uma série de componentes e dependências de infraestrutura que devem ser considerados ao montar a automação de teste. Coletivamente, eles abrangem todos os pré-requisitos necessários para executar um TAS. Os principais componentes e dependências incluem:

Máquinas hospedeiras: podem ser máquinas virtuais, servidores físicos, laptops e dispositivos (p. ex., tablet e celular). Elas têm o software de automação de teste instalado e é nelas que os scripts de teste são criados e executados.

Rede: é o que dá a TAS acesso ao SUT. Também pode incluir a conexão em rede de várias máquinas hospedeiras para proporcionar a execução paralela de testes automatizados. Normalmente, é necessário que os computadores hospedeiros estejam na mesma rede e configurados adequadamente para se comunicarem entre si.

Plataforma: a automação de teste, como qualquer outro software, pode ser executada em plataformas de nuvem ou projetada para ser executada em contêineres. Desde que a plataforma forneça permissões e acesso ao sistema operacional subjacente, todas as ferramentas e dependências necessárias podem ser instaladas.

Dependências de software: para que as ferramentas de automação de teste funcionem corretamente, é necessário entender todas as outras dependências que a automação de teste tem. Por exemplo, uma determinada ferramenta de automação de teste pode exigir que a versão mais recente de uma linguagem de programação seja instalada primeiro no computador hospedeiro. Os TAE precisam levar em conta essas dependências antes e depois de selecionar uma ferramenta.

SUT: Depois que os componentes, as dependências e a infraestrutura geral tiverem sido considerados e configurados adequadamente, a última etapa é garantir o acesso ao SUT. Além da rede, também é necessário considerar como fazer a interface com o SUT. Por exemplo, em um aplicativo baseado na Web, é necessário instalar um navegador no computador hospedeiro. O tipo de navegador e a versão precisam ser considerados.

4.3.3 Definir dados de automação de teste e requisitos de interface.

Duas considerações que os TAE precisam fazer antes de desenvolver scripts de automação de teste são como eles planejam fazer a interface com o SUT e quais são as dependências de dados dos casos de teste. Veja a seguir alguns exemplos:

API: se o SUT utilizar uma API, ela poderá ser testada no nível do ponto final em vez de exigir uma interface do usuário para fazer a interface no nível do aplicativo. Isso pode exigir uma compreensão mais profunda dos pontos

finais e das comunicações de dados que estão ocultas por uma UI. O TAE teria que entender em que ordem chamar as API para criar um processo de negócio e como correlacionar os dados para manter o caso de teste intacto. As API que são expostas à Internet também são chamadas de API da Web ou serviços da Web.

Interface de banco de dados: algumas ferramentas de automação de teste podem fazer interface diretamente com o banco de dados subjacente do SUT. Os scripts de teste podem ser escritos para verificar os dados em colunas e linhas para garantir que os procedimentos armazenados e outras regras do banco de dados estejam configurados corretamente. Isso pode exigir que valores de dados específicos já existam no banco de dados para testes de confiabilidade.

Compatibilidade da interface: O uso de testes de contrato garante que dois sistemas separados (p. ex., dois micros serviços) sejam compatíveis e possam se comunicar entre si. O teste de contratos pode ser orientado pelo consumidor, pelo produtor e bidirecional. Mais detalhes podem ser encontrados no syllabus CTAL-TAE, seção 5.1.3.

Além disso, os TAE devem considerar cuidadosamente como farão a interface com o SUT e entender as dependências de dados nos casos de teste. Seja testando por meio de APIs, interfaces de banco de dados ou garantindo a compatibilidade de interfaces entre sistemas, a atenção a essas considerações é essencial para uma automação de teste robusta e eficaz. Ao abordar esses fatores cuidadosamente, os TAE podem aumentar a confiabilidade e a eficiência de seus esforços de automação, contribuindo, em última análise, para a qualidade e o sucesso do SUT.

5 Análise de Impacto da Automação de Teste (150min)

Palavras-chave

cobertura, relatório de teste

Objetivos de Aprendizagem:

5.1 Investimento na configuração e manutenção da automação de teste.

CT-TAS-5.1.1 (K3) Demonstrar o retorno sobre o investimento da criação de uma solução de automação de teste.

5.2 Métricas de automação de teste.

CT-TAS-5.2.1 (K2) Classificar métricas para automação de teste.

5.3 O valor da automação de teste no nível do projeto e da organização.

CT-TAS-5.3.1 (K3) Identificar as considerações organizacionais para o uso da automação de teste.

CT-TAS-5.3.2 (K3) Analisar as características do projeto que ajudam a determinar a implementação ideal dos objetivos do teste de automação de teste.

5.4 Decisões tomadas com base nos relatórios de automação de teste.

CT-TAS-5.4.1 (K2) Analisar os dados do relatório de teste a tomada de decisões.

5.1 Investimento na Configuração e Manutenção da Automação de Teste.

5.1.1 Demonstrar o retorno sobre o investimento da criação de uma solução de automação de teste.

É importante estimar os custos de configuração e manutenção da automação de teste antes de iniciar a implementação de um projeto. Além dos valores que a automação pode trazer para um projeto, é vantajoso compreender o cálculo do ROI do projeto.

O cálculo do ROI pode fornecer feedback significativo em qualquer momento do ciclo de vida de um projeto, demonstrando o retorno de uma atividade pelo esforço investido. O cálculo do ROI pode ser ajustado ainda mais com a inclusão dos custos do testador manual e do TAE simplesmente multiplicando o tempo gasto com seus respectivos custos de mão de obra.

Para calcular o ROI, é preciso determinar o investimento na automação de teste (ou seja, tempo e custo) e a economia obtida com ela:

$$\boldsymbol{ROI = \frac{Economia}{Investimento}}$$

É importante observar que a economia e o investimento podem ser calculados considerando diferentes métricas e dados, em diferentes medidas e unidades. No escopo deste syllabus, um modelo simples é usado para demonstrar a abordagem, com unidades de tempo e não de custo. Se determinadas atividades forem medidas apenas em custo, esse valor poderá ser convertido em tempo usando uma taxa específica para o projeto.

Em geral, a economia obtida com a automação de teste se deve ao fato de que os mesmos testes podem ser executados em um tempo significativamente menor do que se fossem executados manualmente. Isso também significa que eles podem ser executados com mais frequência. Portanto, o número de testes que estão sendo executados pode ser aumentado.

Para calcular a economia, você precisa considerar as seguintes métricas:

- Tempo para executar um caso de teste manualmente.
- Tempo para executar um caso de teste automatizado.
- Número de casos de teste.
- Número de execuções de teste.

Para calcular o investimento, você precisa considerar as seguintes métricas:

- Tempo para configurar a automação de teste.
- Tempo médio para desenvolver scripts de teste automatizados.
- Número de scripts de teste automatizados implementados.
- Tempo médio de manutenção de um script de teste automatizado.
- Hora de executar um script de teste automatizado.
- Porcentagem de scripts de teste automatizados com falha.
- Número de casos de teste definidos.
- Número de execuções de teste.

Adaptando o modelo descrito ao desenvolvimento ágil de software, os sprints de um projeto podem ser previstos conforme ilustrado no gráfico abaixo. É possível determinar o sprint a partir do qual a automação de teste retornou seu investimento usando o gráfico.

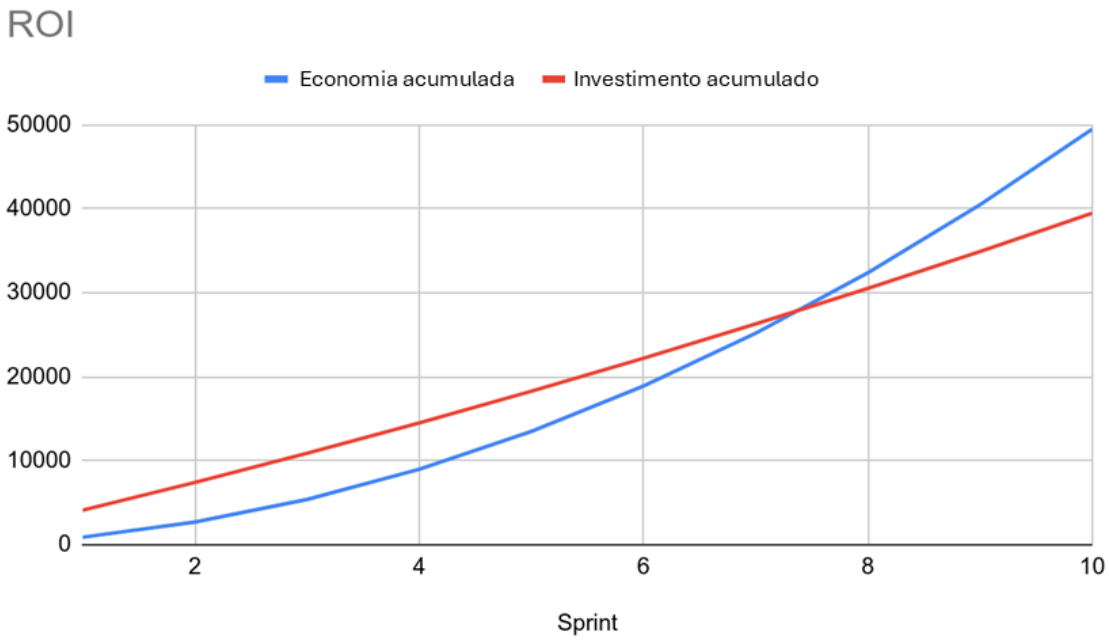

Figura 2: Exemplo de um cálculo de ROI mostrando o ponto de retorno do investimento

Tenha em mente certas conexões entre as métricas medidas para calcular o ROI, incluindo:

- Se a duração planejada do projeto for menor que o ponto de virada do ROI, não vale a pena introduzir a automação de teste. Nesse caso, ao executar os testes manualmente, economiza-se tempo e esforço.
- Como o investimento depende significativamente do tempo de execução dos testes automatizados, a aplicação da pirâmide de testes e a implementação de casos de teste no nível de teste correto podem reduzir esse tempo de execução e melhorar o ROI.

5.2 Métricas de Automação de Teste.

5.2.1 Classificar métricas para a automação de teste.

A análise de tendências com base em dados factuais ajuda na tomada de decisões. Ao coletar métricas de uma TAS os testadores podem avaliá-la e tomar decisões sobre ela:

- Adequação da TAS ao projeto.
- Adaptabilidade da TAS para expandir suas funcionalidade para novas condições de teste.
  - Uma mudança no fluxo de usuário no SUT.
  - Uma mudança na forma como os testes são realizados.
- Capacidade de manutenção da automação de teste devido a defeitos encontrados na TAS.

O custo da medição deve ser o mais baixo possível, e isso pode ser alcançado com a automação da coleta e do relatório de métricas. Alguns exemplos podem ser vistos abaixo.

Taxa de aprovação/reprovação

Essa é uma métrica comum e rastreia a proporção entre os testes automatizados que foram aprovados e os testes automatizados que não conseguiram atingir o resultado esperado.

$$Taxa\ de\ aprovacao/reprovacao = \frac{quantidade\ de\ testes\ aprovados}{quantidade\ de\ testes\ reprovados}$$

Proporção de falhas e defeitos

Um problema comum com os testes automatizados é que muitos deles podem falhar pelo mesmo motivo, ou seja, um único defeito no SUT. Medir o número de testes automatizados que falham por um determinado defeito pode ajudar a indicar onde isso pode ser um problema. Além disso, é possível coletar o número de falhas por defeito e rastrear o motivo da falha: defeito no subsistema, em todo o sistema, problema com os dados de teste ou infraestrutura.

Tempo de execução da automação de teste

Uma das métricas mais fáceis de determinar é o tempo que leva para executar os testes automatizados. No início da TAS isso pode não ser importante, mas à medida que o número de casos de teste automatizados aumenta, essa métrica pode se tornar bastante importante. Essa métrica inclui o tempo de construção da TAS.

Número de casos de teste automatizados.

Essa métrica pode ser usada para mostrar o progresso feito pelo projeto de automação de teste. Mas é preciso levar em conta que o número de casos de teste automatizados não revela muitas informações; por exemplo, ele não indica o nível de cobertura.

Cobertura funcional de automação de teste.

Essa cobertura indica a porcentagem de requisitos funcionais cobertos por casos de teste automatizados.

Cobertura do código

Cobertura de código rastreia quantas linhas de código são executadas por testes de baixo nível (ou seja, teste de componentes).

Não existe uma porcentagem absoluta que indique uma cobertura adequada e 100% de cobertura de código geralmente é difícil de alcançar em qualquer coisa que não seja o software mais simples. No entanto, é consenso que mais cobertura é melhor, pois aumenta a confiança no SUT. À medida que outros testes automatizados de baixo nível são adicionados, espera-se que a cobertura de código aumente.

5.3 O Valor da Automação de Teste no Nível do Projeto e da Organização.

5.3.1 Identificar as considerações organizacionais para o uso da automação de teste.

Antes de iniciar a automação de teste em qualquer projeto, é necessário identificar o seguinte na organização:

Políticas e práticas de desenvolvimento de software.

É bom verificar como as equipes de desenvolvimento estão trabalhando e que tipo de documentação está presente. Qualquer documentação disponível pode ser útil para identificar como a automação de teste pode ser conectada aos processos das equipes de desenvolvimento. A documentação pode incluir a especificação técnica do SUT, o software e as ferramentas de desenvolvimento usadas ou quaisquer políticas disponíveis sobre desenvolvimento, como diretrizes de revisão de código, padrões de codificação definidos e processos de mesclagem.

Projetos de automação de teste ativos existentes e seu status.

No caso de um projeto de desenvolvimento em andamento, pode haver um ou mais projetos de desenvolvimento da TAS por equipes diferentes. Uma recomendação geral para os tomadores de decisão é verificar esses projetos existentes e seu status, para ver se algum deles se encaixa nos objetivos de teste definidos do nova TAS. Ao analisar a solução, é possível determinar se é necessário reutilizar algum dos TAS existentes ou criar um novo com base nas necessidades atuais.

Os Especialistas no Assunto (SME) da organização para automação de teste

Recomenda-se procurar SME que possam ajudar na implementação de um nova TAS dentro da organização. Os SME podem compartilhar histórias e lições aprendidas sobre sua TAS e sua implementação. A partir disso, é

possível identificar diferentes riscos que precisam ser considerados ou evitados para que a implementação da TAS seja bem-sucedida.

Disponibilidade de ambientes de teste

No caso de grandes organizações, uma recomendação geral é reunir informações sobre os ambientes de teste existentes, seu uso e disponibilidade. Essas informações são cruciais para evitar arruinar o trabalho de qualquer outra equipe ao implantar um nova TAS e usar o sistema sem qualquer notificação ou acordo. Muitas vezes, as necessidades de um projeto exigem um novo ambiente de teste, de modo que os existentes podem ser usados como baseline para a criação de um novo e o trabalho pode ser feito mais facilmente se as equipes e pessoas responsáveis forem identificadas e estiverem disponíveis.

Ferramentas e licenças de teste

Sempre vale a pena obter informações sobre as ferramentas e licenças que a organização possui atualmente. Ao identificar essas ferramentas, custos e cronogramas, o planejamento de um TAS pode ser reduzido. E as ferramentas para um novo projeto podem já estar disponíveis. Por exemplo, se o teste em nuvem for configurado por meio de um provedor definido e as licenças estiverem disponíveis para o novo projeto, não faz sentido usar um provedor de nuvem diferente. Recomenda-se usar as mesmas ferramentas e licenças para reduzir os custos do projeto.

5.3.2 Analisar as características do projeto que ajudam a determinar a implementação ideal dos objetivos do teste de automação de teste.

Há várias características importantes do projeto que podem definir uma maneira ideal de trabalhar e ajudar a definir objetivos bem-sucedidos de automação de teste.

Domínio

É importante entender que cada domínio difere em termos de normas ou padrões. Por exemplo, há diferentes regulamentações e riscos no domínio do tema turismo em comparação com o domínio de tema financeiro ou da saúde. É sempre recomendável verificar os diferentes padrões e restrições de domínio para garantir que os objetivos planejados de automação de teste estejam em conformidade.

Plataformas

Em termos de objetivos de teste de automação de teste, também é importante avaliar quais plataformas o projeto abrange e onde seria vantajoso fazer a automação de teste. O planejamento da automação de teste em várias plataformas pode ser mais difícil, pois pode haver a necessidade de várias soluções para cobrir as diferentes plataformas. Em muitos casos, como para dispositivos móveis e Web, as mesmas ferramentas podem ser usadas. Mas o planejamento determinará a reutilização da TAS.

Linguagem de programação e conjunto de tecnologias (technology stack)

Os detalhes de implementação de um projeto, como a linguagem de programação e um conjunto de tecnologias, também determinam que tipo de objetivos de automação de teste uma organização deve definir. Recomenda-se usar as mesmas linguagens de programação que os desenvolvedores estão usando em um determinado projeto. Ao fazer isso, a colaboração pode ser muito mais fácil, e a aprendizagem cruzada ao realizar revisões de código em conjunto melhorará ainda mais a qualidade do SUT e o conhecimento dos TAE.

Maturidade do projeto

Ao analisar a maturidade do projeto, é possível obter informações para desenhar o objetivo ideal de automação de teste. Ao identificar os diferentes fatores, como o número de casos de teste, os pipelines de CI/CD existentes, o número de testadores e TAE, podem ser criados diferentes objetivos de teste e um roteiro. Por exemplo, se houver muitos casos de teste manual com alta prioridade e o tempo de execução do teste manual for longo, é importante tentar automatizá-los primeiro para economizar tempo e esforço. Além dos fatores acima, a linha do tempo do projeto também é muito importante. Se o projeto for curto ou terminar em breve e não houver muito tempo para fazer as coisas, não vale a pena planejar um TAS grande e robusto. Entretanto, se o projeto for um projeto novo, vale a pena elaborar um roteiro incremental passo a passo baseado em um produto mínimo viável.

Participação dos stakeholders

Muitas vezes, a automação de teste não é aproveitada porque os principais stakeholders não a aceitam. Isso pode ser devido ao medo de que a velocidade caia abaixo da meta, aos prazos apertados, ao orçamento ou a outras

preocupações. Após a análise da maturidade do projeto, se houver espaço para iniciar a implementação de uma TAS, um stakeholder estratégico precisa identificar os riscos do produto, listar os benefícios da introdução da automação de teste e elaborar um plano adequado. Esse plano precisa destacar os marcos a serem alcançados com e por meio da automação de teste e indicar como a testabilidade do SUT precisa ser aprimorada para introduzir com sucesso uma TAS que abordará os riscos identificados. Por fim, esse plano precisa ser apresentado aos stakeholders.

Conhecimento e experiência relevante da equipe.

O fator mais importante e relevante para o sucesso da automação de teste são as habilidades e a experiência dos testadores. É vantajoso trabalhar com os testadores e usar seu conhecimento para definir objetivos de teste com os quais eles e os analistas de negócios se sintam confortáveis. Geralmente, os gerentes ou líderes de teste criam uma matriz de competências e habilidades para identificar o conhecimento disponível nas equipes e, ao mesmo tempo, identificar as lacunas. Essas lacunas podem ser identificadas com diferentes objetivos, como treinamento e tarefas de implementação atribuídas.

Suporte ao gerenciamento de testes e orçamento

Dependendo do tamanho e da maturidade de um determinado projeto, o orçamento disponível e o suporte do gerenciamento de testes devem ser considerados ao definir os objetivos de teste para a automação de teste. É importante definir objetivos de teste que possam ser cumpridos, o que, por sua vez, permitirá o apoio do gerenciamento de testes.

Ao fazer uma proposta de estratégia de automação de teste para a gerência, a documentação precisa ser concisa, definindo claramente as lacunas atualmente identificadas ou os custos futuros para a automação de teste se implementada corretamente. Uma lista de recomendações e seus benefícios, apontando o valor comercial e as reduções de custo, incentivará a gerência de testes a aprovar o desenvolvimento ou a melhoria de um TAS.

Características de qualidade

A ISO/IEC 25010:2011 define as características de qualidade que estão listadas no syllabus ISTQB CTFL, seção 2.2.2. Tipos de Teste. Essas características de qualidade podem então ser usadas para avaliar a TAS atual ou para determinar as métricas que serão coletadas por ela.

5.4 Decisões tomadas com base em Relatórios de Automação de Teste.

5.4.1 Analisar os dados do relatório de teste para tomada de decisões.

O formato e o conteúdo de um relatório de automação de teste podem variar de acordo com os stakeholders que o recebem. Ele pode ser criado para os stakeholders gerenciais, operacionais ou técnicos. Informações adicionais podem ser encontradas no syllabus CTAL-TAE, seção 6.1.3.

Ao receber um relatório de automação de teste, ele pode ser incorporado a um relatório de teste mais amplo ou pode ser consolidado e, em seguida, escalado dentro da estrutura organizacional.

Diferentes stakeholders encontram valores diferentes em um relatório de automação de teste. Uma abordagem estratégica é identificar as principais métricas que são importantes para os stakeholders envolvidos, enfatizando essas métricas importantes.

Os dados coletados com a automação podem ajudar:

- Identificar tendências e realizar análises de causa raiz;
- Transferir os esforços de automação de teste para a manutenção;
- Transferir os esforços de automação de teste para melhorias e desenvolvimento adicional de um TAF;
- Adicionar recursos à TAS geral;
- Aumentar as abordagens de teste *Shift Right* e *Shift Left*;
- Expandir a cobertura funcional da automação de teste em sprints futuros;
- Concentrar-se mais em grupos de defeitos;
- Aconselhar os desenvolvedores sobre as áreas para melhoria do código;

- Aconselhar sobre os processos gerais do SDLC;
- Alterar o conteúdo e o formato dos futuros relatórios de automação de teste.

Com base nas informações descritas acima, os TAE, em colaboração com outros stakeholders, podem identificar lacunas e determinados pontos de melhoria na cobertura da automação de teste existente e nos resultados dos testes.

6 Estratégias de Implementação e Melhoria para Automação de Teste (150min)

Palavras-chave

cobertura, pré-condição, suíte de testes

Objetivos de Aprendizagem:

6.1 Transição de atividades de testes manuais para testes contínuos

CT-TAS-6.1.1 (K2) Descrever os fatores e as atividades de planejamento na transição do teste manual para a automação de teste.

CT-TAS-6.1.2 (K2) Descrever os fatores e as atividades de planejamento na transição da automação de teste para testes contínuos.

6.2 Compreensão dos fatores e atividades de planejamento na transição da automação de teste para testes contínuos.

CT-TAS-6.2.1 (K3) Realizar uma avaliação dos ativos e práticas de automação de teste para identificar áreas de melhoria.

6.1 Transição das Atividades de Testes Manuais para Testes Contínuos.

6.1.1 Descrever os fatores e as atividades de planejamento na transição do teste manual para a automação de teste.

A oportunidade mais fácil de fazer a transição do teste manual para a automação de teste é direcionar o teste de regressão. Uma suíte de testes de regressão cresce à medida que os testes funcionais e não funcionais de hoje se tornam os testes de regressão amanhã. É apenas uma questão de tempo até que o número de testes de regressão se torne maior do que o tempo e os recursos disponíveis para uma equipe de teste manual tradicional.

Custos de transição

Durante a transição do teste manual para o teste automatizado, o projeto deve esperar um aumento nos custos, pois tanto o teste manual quanto o automatizado ocorrem simultaneamente. Quando se considera que os testes automatizados substituem ou suplantam adequadamente as atividades de execução de testes manuais, mais esforços podem ser direcionados para testes exploratórios e para a definição de casos de teste adicionais para automação de teste, o que tem um custo diferente em comparação com os testes de regressão manual.

Sobreposição funcional

A sobreposição funcional ocorre quando os desenvolvedores de scripts de teste incluem exatamente as mesmas etapas de automação de teste em diferentes casos de teste. Por exemplo, a maioria dos casos de teste começará com uma sequência de login de etapas de teste. Isso pode incluir a inserção de um nome de usuário, uma senha e a seleção de um botão de login. Adicionar isso em cada caso de teste aumenta as atividades de manutenção. Se alguma vez for acrescentada uma etapa de teste adicional ao processo de login, a mesma alteração terá de ser atualizada em cada caso de teste. Uma abordagem melhor é criar um componente de automação de teste repetível do processo de login e fazer com que todos os casos de teste façam referência a essa funcionalidade. Consulte o syllabus CTAL-TAE, seção 3.1.5 para obter detalhes sobre o padrão do modelo de fluxo.

Compartilhamento de dados

Os testes geralmente compartilham dados de teste. Isso pode ocorrer quando os testes usam o mesmo registro de dados de teste para executar diferentes funcionalidades do SUT. Um exemplo disso pode ser o caso de teste "A", que verifica o tempo de férias disponível de um funcionário, enquanto o caso de teste "B" verifica quais cursos o funcionário fez como parte de suas metas de desenvolvimento de carreira. Cada caso de teste usa o mesmo funcionário, mas verifica parâmetros diferentes. Em um ambiente de teste manual, os dados de teste do funcionário normalmente seriam duplicados muitas vezes em cada caso de teste manual que verificasse os dados de teste do funcionário. Entretanto, em um teste automatizado, os dados de teste compartilhados devem, sempre que possível e viável, ser armazenados e acessados de uma única fonte para evitar a duplicação ou a introdução de erros.

Interdependência de testes

Ao executar testes de regressão complexos, um teste pode ter uma dependência de um ou mais testes. Essa ocorrência pode ser bastante comum. Por exemplo, um novo "ID de pedido" é criado como resultado de uma etapa de teste. Os testes subsequentes podem querer verificar se: a) o novo pedido é exibido corretamente no sistema, b) é possível fazer alterações no pedido ou c) a exclusão do pedido é bem-sucedida. Em cada caso, o valor “ID do pedido”, criado dinamicamente no primeiro teste, deve ser capturado para reutilização em testes posteriores. Dependendo do projeto da TAS isso pode ser resolvido. Se o "ID do pedido" não puder ser encontrado pelos casos de teste subsequentes, eles falharão.

Condições prévias para a execução do teste

Com muita frequência, os TAE iniciam imediatamente o desenvolvimento do script de teste sem antes compreender as condições prévias necessárias para garantir que um caso de teste possa ser executado de forma confiável. Isso pode se tornar extremamente desafiador quando se tenta executar um caso de teste no mesmo SUT em vários ambientes de teste. Exemplos de pré-condições incluem nomes de usuário, funções de conta, valores de tabela de dados e outras entradas de dados específicas que tornam o caso de teste repetível. Uma boa estratégia incluirá uma análise inicial para entender quais informações precisam existir no SUT antes de automatizar um caso de teste específico. Além disso, a estratégia pode ser aprimorada automatizando a criação de condições prévias antes da execução dos casos de teste reais, para economizar tempo. Isso pode incluir a execução de scripts de teste de pré-condição que alimentam o SUT a partir da interface do usuário ou processos automatizados em lote que carregam dados de teste em um banco de dados.

Cobertura funcional

Identifique as lacunas funcionais nos testes que podem ser candidatas à automação de teste, conforme explicado no Capítulo 3. 100% dos casos de teste manual que são automatizados não representam 100% de todos os casos de teste possíveis que podem ser automatizados com ferramentas de teste. À medida que mais testes são automatizados, os testadores recuperam o tempo de execução do teste, que pode ser usado para identificar testes adicionais no SUT para aumentar a cobertura.

Testes executáveis

Antes de automatizar um teste de regressão manual é importante verificar se o teste de regressão manual funciona corretamente. Isso fornece o ponto de partida correto para garantir uma conversão bem-sucedida em um teste de regressão automatizado. Se o teste de regressão manual não for executado corretamente, pode ser porque foi mal escrito, usa dados de teste inválidos, está desatualizado ou fora de sincronia com o SUT atual ou por causa de um defeito no SUT. Automatizá-lo antes de compreender e/ou resolver a causa raiz da falha criará um teste de regressão automatizado que não funciona, o que é um desperdício e uma improdutividade. É importante demonstrar a funcionalidade equivalente que os novos testes automatizados trazem, para transmitir confiança nos testes automatizados que substituirão os antigos testes manuais.

6.1.2 Descrever os fatores e as atividades de planejamento na transição da automação de teste para testes contínuos.

O teste contínuo envolve a utilização de ativos de teste com muito mais frequência do que nos SDLC tradicionais. Isso é feito começando a executar constantemente conjuntos de testes imediatamente após as alterações no código serem feitas e disponibilizadas nos ambientes de teste. Isso ajuda a fornecer uma resposta imediata e reduz os custos com defeitos ao detectá-los mais cedo no processo. Isso pode ser feito com ferramentas de desenvolvimento sofisticadas e com a disposição de mudar os testes para o início do processo de desenvolvimento.

A adaptação da TAS para testes contínuos requer o seguinte:

- As suítes de teste precisam ser alteradas para serem executadas na TAS atualizada: faça as alterações necessárias nos conjuntos de teste e teste-os antes de implantá-los na TAS;
- Simuladores, drivers e interfaces usadas nos testes precisam ser adequadas à TAS atualizada: faça as alterações necessárias na suíte de testes e teste-a antes de implementá-la na TAS;
- A infraestrutura precisa ser alterada para acomodar a TAS atualizada: avaliação dos componentes da infraestrutura que precisam ser alterados;
- A TAS atualizada apresenta defeitos adicionais ou defeitos de performance: faça uma análise dos riscos versus benefícios. Se os defeitos descobertos impossibilitarem a atualização da TAS, talvez seja melhor não prosseguir com a atualização ou esperar por uma próxima versão da TAS. Se os defeitos forem insignificantes em comparação com os benefícios de corrigi-los, a TAS ainda poderá ser atualizada;
- Certifique-se de criar notas de versão dos defeitos conhecidos para notificar os TAE e outros stakeholders e tente obter uma estimativa de quando os defeitos serão corrigidos.

Todos os pontos mencionados nesta seção tornam-se particularmente importantes quando se utiliza a CI/CD. A criação de pipelines e a automação do processo de compilação são uma combinação natural com a TAS que está sendo desenvolvida. Se a ferramenta de orquestração de compilação estiver no ambiente de teste correto, o pipeline poderá ser estendido para incluir testes automatizados para verificar o SUT logo após a implementação. Para isso, é necessário que a ferramenta de automação de teste esteja configurada corretamente e possa acessar o SUT, além de poder entrar em contato com o repositório de código e acessar os scripts de provisionamento de dados.

6.2 Compreensão dos Fatores e Atividades de Planejamento na Transição da Automação de Teste para Testes Contínuos.

6.2.1 Realizar uma avaliação dos ativos e práticas de automação de teste para identificar áreas de melhoria.

Assim como em qualquer outra atividade de desenvolvimento, é importante ter uma estratégia para interromper o desenvolvimento da TAS e procurar oportunidades para refatorar a solução. As áreas a serem monitoradas são a implementação inicial, a manutenção e a capacidade de avaliar a solução do ponto de vista da repetibilidade. Algumas boas categorias a serem monitoradas são quantas horas são gastas no desenvolvimento da TAS, quantas horas são gastas na correção da TAS e quanto tempo os testadores economizam em comparação com os testes manuais. Um indicador de que algo está errado é se o tempo de manutenção da TAS for maior do que o tempo de teste manual.

Antes de iniciar a primeira implementação de uma TAS é importante garantir que ela possa ser executada em seu próprio ambiente, que esteja isolado de alterações aleatórias e que os casos de teste possam ser atualizados e gerenciados. Tanto a TAS quanto sua infraestrutura devem ser mantidas. No caso da primeira implementação, são necessárias as seguintes etapas básicas:

- As ferramentas de cobertura de código indicam quanto do código é executado pela suíte de testes de componentes e onde existem lacunas que podem ser cobertas por testes de componentes adicionais;
- A cobertura funcional do SUT pode ser determinada pela criação de uma matriz de rastreabilidade de requisitos, que revela qual funcionalidade ainda não foi coberta por nenhum caso de teste;
- Definir um uso consistente da infraestrutura da TAS em todos os projetos ou em toda a organização;
- Desenvolver uma estratégia consistente de gerenciamento de configuração para os conjuntos de testes;
- Criar diretrizes comuns de desenvolvimento da TAS aproveitando as práticas recomendadas descritas no syllabus CTAL-TAE, seção 3.1.4;
- Implementar as pré-condições. Muitas vezes, um teste não pode ser executado antes da definição das pré-condições. Essas podem incluir a seleção correta do banco de dados ou dos dados de teste a partir dos quais o teste será realizado ou a definição de valores ou parâmetros iniciais. Muitas dessas etapas de inicialização necessárias para estabelecer as pré-condições de um teste podem ser automatizadas. Isso permite uma solução mais confiável e segura quando essas etapas não podem ser perdidas antes da execução dos testes. À medida que os testes de regressão são convertidos em automação de teste, essas pré-condições precisam fazer parte do processo de automação de teste.

Quando ocorrerem atualizações incrementais da TAS para novos recursos ou para fins de manutenção, deve-se considerar o seguinte:

- Avaliar as atualizações das ferramentas de teste ou outras ferramentas de teste mais recentes que forneçam mais recursos para a TAS;
- Avaliar maneiras de otimizar ainda mais os recursos e o desempenho da TAS;
- Identificar oportunidades para decompor e modularizar ainda mais os scripts de teste para aumentar a reutilização;
- Garantir o conhecimento e a conscientização sobre componentes reutilizáveis e sua utilização consistente;
- Coletar evidências sobre possíveis áreas de melhoria, fazer recomendações e listar seus benefícios;
- Avaliar e corrigir as áreas de sobreposição funcional. Ao automatizar os testes de regressão existentes, é uma boa prática identificar qualquer sobreposição funcional que exista entre os casos de teste e, sempre que possível, reutilizar componentes de automação de teste desenvolvidos anteriormente;
- Avaliar casos de testes manuais adicionais para identificar oportunidades de automatizá-los e criar itens de backlog para implementação;
- Destacar as oportunidades de melhoria do projeto de teste e do gerenciamento de dados de teste;
- Garantir que todos os conjuntos de testes existentes sejam adaptados à versão mais recente da TAS;

- Se a automação de teste estiver causando fila no pipeline, diminua o escopo dos testes integrados para os mais críticos, ou seja, crie um conjunto de *smoke tests*. Uma suíte de testes de regressão maior pode ser acionada separadamente ou executada sob demanda.

7 Referências

Normas

As normas para a automatização de testes incluem, mas não se limitam a:

The Automatic Test Markup Language (ATML) by IEEE (Institute of Electrical and Electronics Engineers) consisting of:

- IEEE Std 1671.1: Test Description
- IEEE Std 1671.2: Instrument Description
- IEEE Std 1671.3: UUT Description
- IEEE Std 1671.4: Test Configuration Description
- IEEE Std 1671.5: Test Adaptor Description
- IEEE Std 1671.6: Test Station Description
- IEEE Std 1641: Signal and Test Definition
- IEEE Std 1636.1: Test Results

ISO/IEC 30130:2016 (E) Software engineering — Capabilities of software testing tools

The Testing and Test Control Notation (TTCN-3) by ETSI (European Telecommunication Standards Institute) and ITU (International Telecommunication Union) consisting of:

- ES 201 873-1: TTCN-3 Core Language
- ES 201 873-2: TTCN-3 Tabular Presentation Format (TFT)
- ES 201 873-3: TTCN-3 Graphical Presentation Format (GFT)
- ES 201 873-4: TTCN-3 Operational Semantics
- ES 201 873-5: TTCN-3 Runtime Interface (TRI)
- ES 201 873-6: TTCN-3 Control Interface (TCI)
- ES 201 873-7: Using ASN.1 with TTCN-3
- ES 201 873-8: Using IDL with TTCN-3
- ES 201 873-9: Using XML with TTCN-3
- ES 201 873-10: TTCN-3 Documentation
- ES 202 781: Extensions: Configuration and Deployment Support
- ES 202 782: Extensions: TTCN-3 Performance and Real-Time Testing
- ES 202 784: Extensions: Advanced Parameterization
- ES 202 785: Extensions: Behaviour Types
- ES 202 786: Extensions: Support of interfaces with continuous signals
- ES 202 789: Extensions: Extended TRI

The UML Testing Profile (UTP) by OMG (Object Management Group) specifying test specification concepts for:

- Test Architecture
- Test Data
- Test Behavior
- Test Logging
- Test Management

Documentos ISTQB®

Identifier	Reference
ISTQB-AL-TM	ISTQB Certified Tester, Advanced Level Syllabus, Test Manager, Version 2.0, October 2012, available from [ISTQB-Web]
ISTQB-EL-TM-MTT	ISTQB Certified Tester, Expert Level Test Management Managing the Test Team, Version 2.0, November 2011, available from [ISTQB-Web]
ISTQB-FL	ISTQB Certified Tester, Foundation Level Syllabus, Version 4.0, April 2023, available from [ISTQB-Web]
ISTQB-PT	ISTQB Certified Tester, Performance Testing Syllabus, December 2018, available from [ISTQB-Web]

ISTQB-TAE	ISTQB Certified Tester, Test Automation Engineering Syllabus, February 2024, available from [ISTQB-Web]
ISTQB-Glossary	ISTQB Glossary of terms, available online from [ISTQB-Web]

Livros

Paul Baker, Zhen Ru Dai, Jens Grabowski and Ina Schieferdecker, “Model-Driven Testing: Using the UML Testing Profile”, Springer 2008 edition, ISBN-10: 3540725628, ISBN-13: 978-3540725626

Efriede Dustin, Thom Garrett, Bernie Gauf, “Implementing Automated Software Testing: how to save time and lower costs while raising quality”, Addison-Wesley, 2009, ISBN 0-321-58051-6

Efriede Dustin, Jeff Rashka, John Paul, “Automated Software Testing: introduction, management, and performance”, Addison-Wesley, 1999, ISBN-10: 0201432870, ISBN-13: 9780201432879

Mark Fewster, Dorothy Graham, “Experiences of Test Automation: Case Studies of Software Test Automation”, Addison-Wesley, 2012

Mark Fewster, Dorothy Graham, “Software Test Automation: Effective use of test execution tools”, ACM Press Books, 1999, ISBN-10: 0201331403, ISBN-13: 9780201331400

Boby Jose, “Test Automation, a manager's guide”, September 2021, ISBN: 9781780175478

James D. McCaffrey, “.NET Test Automation Recipes: A Problem-Solution Approach”, APRESS, 2006 ISBN-13:978-1-59059-663-3, ISBN-10:1-59059-663-3

Daniel J. Mosley, Bruce A. Posey, “Just Enough Software Test Automation”, Prentice Hall, 2002, ISBN-10: 0130084689, ISBN-13: 9780130084682

Casey Rosenthal, “Chaos Engineering: System Resiliency in Practice by Casey Rosenthal”, April 2020, ISBN-13: 1492043869

Colin Willcock, Thomas Deiß, Stephan Tobies and Stefan Keil, “An Introduction to TTCN-3” Wiley, 2nd edition 2011, ISBN-10: 0470663065, ISBN-13: 978-0470663066

Artigos

Robert V. Binder, Suzanne Miller, “Five Keys to Effective Agile Test Automation for Government Programs” August 24, 2017, Software Engineering Institute, Carnegie Mellon University, https://resources.sei.cmu.edu/asset_files/Webinar/2017_018_101_503516.pdf

DoD CIO, Modern Software Practices “DevSecOps Fundamentals Guidebook: Activities & Tools”, Version 2.2, May 2023, https://dodcio.defense.gov/Portals/0/Documents/Library/DevSecOpsActivitesToolsGuidebookTables.pdf?ver=_Sylg1WJB9K0Jxb2XTvzDQ%3d%3d

Naveen Jayachandran, “Understanding roi metrics for software test automation”, 2005, https://digitalcommons.usf.edu/cgi/viewcontent.cgi?article=3937&context=etd

Thomas Pestak, William Rowell, PhD, “Automated Software Testing Practices and Pitfalls”, September 2018, https://www.afit.edu/stat/statcoe_files/Automated%20Software%20Testing%20Practices%20and%20Pitfalls%20Rev%201.pdf

Andrew Pollner, Jim Simpson, Jim Wisnowski, “Automated Software Testing Implementation Guide for Managers and Practitioners”, October 2018, https://www.afit.edu/stat/statcoe_files/0214simp%202%20AST%20IG%20for%20Managers%20and%20Practitioners.pdf

Siglas usadas neste Syllabus

SUT: Sistema em Teste

TAA: Arquitetura de Automação de Teste

TAE: Engenheiro de Automação de Teste

TAF: Framework de Automação de Teste

TAS: Solução de Automação de Teste

8 Apêndice A: Objetivos de Aprendizagem e Níveis Cognitivos de Conhecimento

Os seguintes objetivos de Aprendizagem são definidos como aplicáveis a este syllabus. Cada tópico do syllabus será examinado de acordo com seu objetivo de Aprendizagem.

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

Nível 2: Compreender (K2)

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

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

Exemplos	Notas
Classificar as ferramentas de teste de acordo com sua finalidade e as atividades de teste que elas suportam.	
Compare os diferentes níveis de teste.	Pode ser usado para procurar semelhanças, diferenças ou ambos.
Diferencie teste de depuração.	Procura diferenças entre os conceitos.
Distinguir entre riscos do projeto e do produto.	Permite que dois (ou mais) conceitos sejam classificados separadamente.
Explicar o impacto do contexto no processo de teste.	
Dê exemplos de por que os testes são necessários.	
Inferir a causa raiz dos defeitos a partir de um determinado perfil de falhas.	
Resumir as atividades do processo de revisão do produto de trabalho.	

Nível 3: Aplicar (K3)

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

Verbos de ação: Aplicar, implementar, preparar, usar

Exemplos	Notas
Aplicar a análise de valor limite para derivar casos de teste a partir de determinados requisitos.	Deve se referir a um procedimento / técnica / processo etc.
Implementar métodos de coleta de métricas para dar suporte aos requisitos técnicos e gerenciais.	
Preparar o ambiente de automação de teste.	
Use a rastreabilidade para monitorar o progresso do teste quanto à integridade e à consistência com os objetivos, a estratégia e o plano de teste.	Pode ser usado em um LO que deseja que o candidato seja capaz de usar uma técnica ou um procedimento. Semelhante a "aplicar".

Referência

(Para os níveis cognitivos dos objetivos de Aprendizagem)

Anderson, L. W. e Krathwohl, D. R. (eds) (2001) A Taxonomy for Learning, Teaching, and

Assessing: A Revision of Bloom's Taxonomy of Educational Objectives, Allyn & Bacon

9 Apêndice B: Matriz de rastreabilidade entre Resultados de Negócios e Objetivos de Aprendizagem

Esta seção lista a rastreabilidade entre os resultados de negócio e os objetivos de aprendizagem do especialista em Test Automation Strategy.

Resultados de Negócio do Test Automation Engineering			N01	N02	N03	N04	N05	N06	N07	N08	N09	N10	N11	N12	N13	N14	N15
Cap.1	Introdução e objetivos do Test Automation Strategy																
1.1	Fatores de sucesso de um projeto de automação de teste																
1.1.1	Definir metas e objetivos para Test Automation Strategy	2	X														
1.1.2	Identificar os fatores técnicos de sucesso de um projeto de automação de teste	2	X														
1.1.3	Resumir os critérios de investimento apropriados na seleção de projetos candidatos para automação de teste	2	X														
Cap.2	Recursos de Automação de Teste																
2.1	Custos e riscos de Implementação de solução de automação de teste																
2.1.1	Comparar soluções técnicas alternativas com relação ao custo de propriedade	2		X													
2.1.2	Explicar as considerações sobre o modelo de licenciamento para ferramentas de automação de teste	2		X													
2.1.3	Exemplificar os fatores a serem considerados na definição de uma estratégia de automação de teste	2		X													
2.2	Funções e responsabilidades na automação de teste																
2.2.1	Resumir as funções e habilidades necessárias para uma de solução de automação de teste bem-sucedida	2			X												
Cap.3	Preparando-se para a Automação de Teste																
3.1	Integração entre os níveis de teste																
3.1.1	Diferenciar as distribuições de automação de teste	2				X											
3.1.2	Selecionar uma abordagem de automação de teste com base na arquitetura do sistema em teste	3				X											

Resultados de Negócio do Test Automation Engineering			N01	N02	N03	N04	N05	N06	N07	N08	N09	N10	N11	N12	N13	N14	N15
3.1.3	Demonstrar maneiras de otimizar a distribuição da automação de teste para alcançar as abordagens Shift Left e Shift Right	2				X											
3.2	Considerações estratégicas em diferentes modelos de ciclo de vida de desenvolvimento de software																
3.2.1	Explicar como os projetos de automação de teste estão em conformidade com os modelos de ciclo de vida de desenvolvimento de software legado	2					X										
3.2.2	Explicar como os projetos de automação de teste estão em conformidade com as práticas recomendadas de desenvolvimento ágil de software que dão suporte à automação de teste	2					X										
3.2.3	Preparar-se para que os projetos de automação de teste estejam em conformidade com as práticas recomendadas de DevOps que apoiam a automação de teste em testes contínuos	3					X										
3.3	Aplicabilidade e viabilidade da automação de teste																
3.3.1	Explicar os critérios para determinar a adequação dos testes para a automação de teste	2						X									
3.3.2	Identificar os desafios que somente a automação de teste pode resolver	2						X									
3.3.3	Identificar as condições de teste que são difíceis de automatizar	2						X									
Cap.4	Estratégias organizacionais de implantação e liberação para Automação de Teste																
4.1	Planejamento de soluções de automação de teste																
4.1.1	Identificar maneiras pelas quais a automação de teste ajuda a reduzir o tempo de lançamento no mercado	2							X								
4.1.2	Identificar as formar em que a automatização dos testes ajuda a verificar os defeitos comunicados de acordo com os requisitos	2							X								
4.1.3	Definir abordagens que permitam o desenvolvimento de cenários operacionalmente relevantes para a automação de teste	2							X								
4.2	Estratégias de implantação																
4.2.1	Definir uma estratégia de implantação de automação de teste	2								X							
4.2.2	Identificar os riscos da automação de teste na implantação	2								X							
4.2.3	Definir abordagens para mitigar os riscos da implementação	2								X							
4.3	Dependências no ambiente de teste																

Resultados de Negócio do Test Automation Engineering			N01	N02	N03	N04	N05	N06	N07	N08	N09	N10	N11	N12	N13	N14	N15
4.3.1	Definir componentes de automação de teste no ambiente de teste	2									X						
4.3.2	Identificar os componentes de infraestrutura e dependências da automação de teste	2									X						
4.3.3	Definir dados de automação de teste e requisitos de interface	2									X						
Cap.5	Análise do impacto da Automação de Teste																
5.1	Investimento na configuração e manutenção da automação de teste																
5.1.1	Demonstrar o retorno sobre o investimento da criação de uma solução de automação de teste	3										X					
5.2	Métricas de automação de teste																
5.2.1	Classificar métricas para automação de teste	2											X				
5.3	O valor da automação de teste no nível do projeto e da organização																
5.3.1	Identificar as considerações organizacionais para o uso da automação de teste	3												X			
5.3.2	Analisar as características do projeto que ajudam a determinar a implementação ideal dos objetivos do teste de automação de teste	3												X			
5.4	Decisões tomadas com base em relatórios de automação de teste																
5.4.1	Analisar os dados do relatório de teste para informar a tomada de decisões	2													X		
Cap.6	Estratégias de implementação e melhoria para a Automação de Teste																
6.1	Transição das atividades de testes manuais para testes contínuos																
6.1.1	Descrever os fatores e as atividades de planejamento na transição do teste manual para a automação de teste	2														X	
6.1.2	Descrever os fatores e as atividades de planejamento na transição da automação de teste para testes contínuos	2														X	
6.2	Compreensão dos fatores e atividades de planejamento na transição da automação de teste para testes contínuos																
6.2.1	Realizar uma avaliação dos ativos e práticas de automação de teste para identificar áreas de melhoria	3															X

10 Apêndice C: Notas de versão

O ISTQB® Test Automation Strategy 2024 é um novo syllabus do ISTQB® que combina aspectos estratégicos da versão anterior do syllabus do ISTQB® Test Automation Engineer de 2016 com atualizações adicionais e práticas recomendadas atuais para implementar e medir o sucesso da automação de teste. Por esse motivo, não há notas de versão detalhadas por capítulo e seção.

11 Apêndice D: Termos Específicos

canary release: Uma estratégia de implantação e teste destinada a reduzir o risco e verificar o novo software, liberando-o para apenas alguns usuários.

contêiner: Uma unidade de software que empacota o código e suas dependências, para que o software seja executado de forma rápida e confiável em todos os ambientes.

DevOps: metodologia que integra e automatiza o trabalho de desenvolvimento de software e operações de TI para aprimorar e encurtar o ciclo de vida de desenvolvimento de software.

padrão de modelo de fluxo: Uma visão de alto nível do domínio de trabalho, seus componentes e as interconexões entre eles.