TESTE DE SOFTWARE

AULA 2

Profª Maristela Teixeira

CONVERSA INICIAL

Ao pensarmos na nossa vida, seja pessoal ou profissional, conseguimos parar alguns segundos e revisar mudanças em todos os sentidos dentro de um período. Tal período pode ser mensurado em semanas, meses e anos. Quanta coisa muda, seja por conta de revalidações de formas de pensamento, seja por meio de tecnologias, conhecimento e experiências. Não é diferente no mundo dos testes de software.

O mesmo ocorre quando pensamos nos profissionais ligados à qualidade de software e testes. Testes de software iniciaram desde que os primeiros códigos foram construídos ao longo da história do software, porém, passaram do modelo tradicional para o ágil. Hoje, há quem fale sobre o pós-ágil.

Este conteúdo explora a profissão e os profissionais relacionados à área de testes de software dentro da disciplina de qualidade de software, observando-se técnicas, metodologias, ferramentas e ciclo de vida de software e de testes.

Saímos de um modelo tradicional, no qual automação e testes de riscos eram praticamente inexistentes. Por isso, profissionais devem ser mais técnicos, mais colaborativos com os times de desenvolvimento, enquadrarem-se no estilo ágil, buscar reaprender e aprender constantemente, aplicar boas práticas e aprender a automatizar testes por meio de ferramentas cada vez mais completas e complexas.

Começamos falando sobre o profissional de qualidade e de testes de software, sobre sua percepção de mudanças constantes, sobre como a estruturação de testes deve ocorrer dentro das áreas de desenvolvimento e testes, como gerenciar e planejar testes e a necessidade cada dia mais frequente do uso de ferramentas de automação de testes.

São conteúdos focados em quem gerencia, planeja, controla e executa testes num mundo de projetos de software ágeis.

TEMA 1 – O PROFISSIONAL DE QUALIDADE E TESTES DE SOFTWARE

Antes de falarmos propriamente no profissional, vamos somente detalhar um pouco sobre as várias subdivisões de atividades dentro de uma área de Qualidade de Software e Testes (QA). Para começar, vamos nos unir em um entendimento comum sobre qualidade de software.

A qualidade do software já foi equiparada a um aplicativo livre de bugs, mas qualquer pessoa no mundo do software hoje concorda que não é mais apenas isso. Se pedirmos aos usuários finais que definam qualidade, nós ouviremos falar de facilidade de uso, aparência, privacidade de dados, rapidez na prestação de informações e disponibilidade de serviços 24 horas por dia, 7 dias por semana. Se solicitarmos às empresas que definam qualidade, ouviremos sobre retorno do investimento, análise em tempo real, tempo de inatividade zero, sem dependência de fornecedor, infraestrutura escalável, segurança de dados, conformidade legal e muito mais. Todos esses são aspectos do que torna um aplicativo de software de alta qualidade hoje. Falhas em qualquer uma dessas áreas afetarão a qualidade de uma forma ou de outra, e é por isso que precisamos testá-las meticulosamente! (Mohran, 2022).

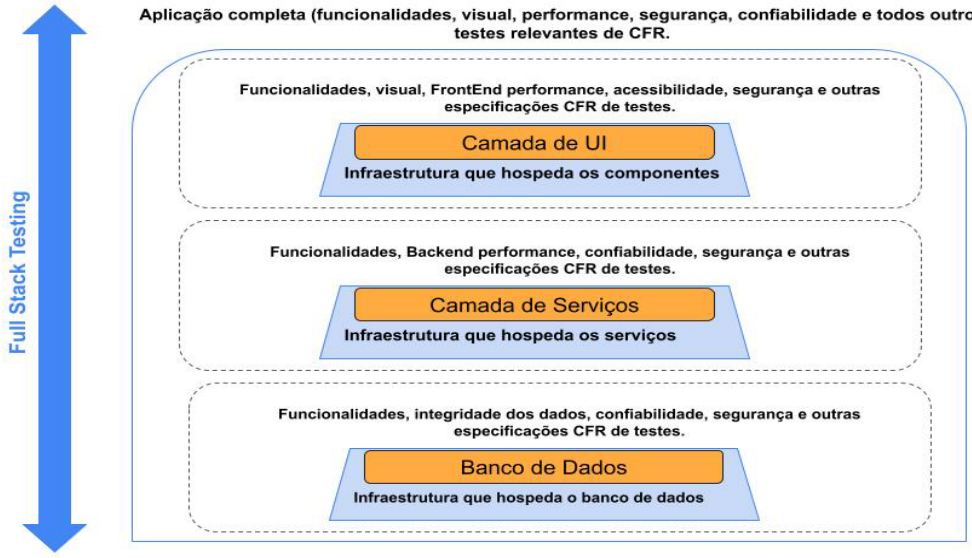
Embora a lista de requisitos de qualidade pareça alta, temos ferramentas e metodologias para atender a maioria dessas necessidades. Assim, a ponte para a alta qualidade é feita pelo conhecimento de ferramentas para qualidade de software, bem como a habilidade de aplicá-las em um determinado contexto, tanto do ponto de vista do desenvolvimento quanto do teste.

1.1 Contextualizando dez habilidades de uma Stack de Testes

Quando falamos especificamente sobre o teste, em poucas palavras, é uma prática para validar se o comportamento do aplicativo é o esperado por todas as partes. Para que o teste seja bem-sucedido, ele precisa ser praticado nos níveis micro e macro e estar entrelaçado com os aspectos granulares do aplicativo, como testar cada método em uma classe, cada valor de dados de entrada, mensagens de log, código de erro, e assim por diante. Da mesma forma, ele deve se concentrar em aspectos macros, como testar recursos, integrações entre recursos e fluxos de trabalho de ponta a ponta (E2E). Mas não podemos parar de testar somente olhando para esses aspectos. Precisamos testar ainda mais as características holísticas de qualidade do aplicativo, tais como segurança, desempenho, acessibilidade, usabilidade, entre outras, para atingirmos o objetivo final de fornecer software de alta qualidade.

Podemos encapsular tudo isso dizendo que precisamos fazer testes da Stack completa! A figura 1 representa essa ideia de Stack de Testes, considerando seus diferentes aspectos de qualidade.

Figura 1 – Stack de Testes dividida em camadas



Fonte: Mohran, 2022.

De fato, o teste e o desenvolvimento Full Stack devem ser inseparáveis, como os dois trilhos de uma ferrovia. Devemos avançar ao longo dos dois trilhos simultaneamente para construir qualidade no produto; caso contrário, temos a garantia de descarrilar. Por exemplo, suponha que estamos escrevendo um pequeno bloco de código para calcular o valor total do pedido para um aplicativo de comércio eletrônico. Precisamos testar se o código está computando a quantidade certa e se é seguro em paralelo. Se não fizermos isso, podemos acabar com lacunas na linha férrea e, se continuarmos a desenvolver em cima dessa linha fraturada, teremos integração deficiente e funcionalidade abaixo do ideal.

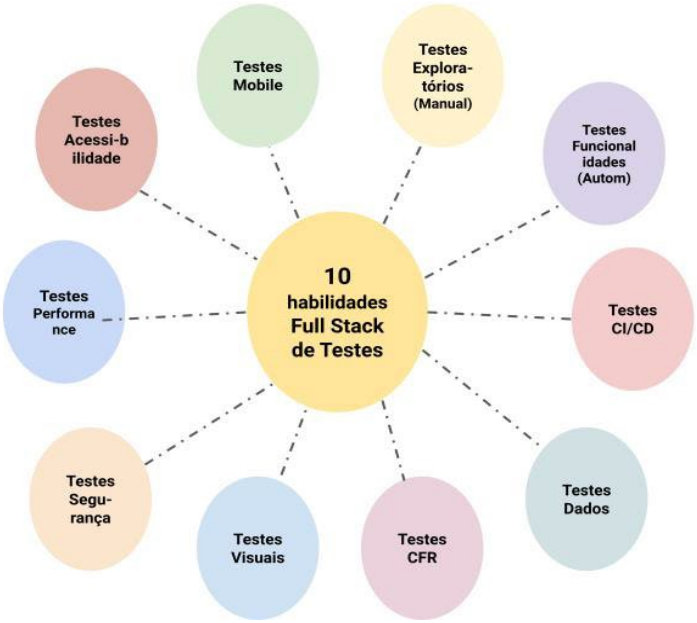
Para incorporar o teste em um nível tão elementar, as equipes precisam parar de pensar nele como uma atividade de pós-desenvolvimento em silos, como era feito tradicionalmente. A stack de testes completa precisa começar em paralelo com o desenvolvimento e ser praticado durante todo o ciclo de entrega, fornecendo feedback mais rápido. A prática de iniciar o teste no início do ciclo de entrega é chamada de teste Shift-Left (figura 2), o qual é um princípio fundamental a ser seguido para que o teste da Stack produza os resultados corretos.

Figura 2 – Testes Shift-left

Crédito: Mohran, 2022.

Quando falamos de Stack, inicialmente falamos em testes manuais e automatizados divididos em verificações, validações e testes entre as dez características da Stack de testes (figura 3).

Figura 3 – Stack de Testes dividida em dez habilidades



Crédito: Mohran, 2022.

Vamos explorar essas dez habilidades dentro da área de controle e garantia de software.

- Testes exploratórios (manuais):
    - O teste exploratório manual é diferente de teste manual. Teste exploratório refere-se à verificação de uma determinada lista de requisitos e não exige necessariamente uma mentalidade analítica;
- Testes funcionais (automatizados):

- Essa é uma das principais habilidades para o teste Shift-left, e fazer testes automatizados também reduz significativamente o esforço de teste manual, especialmente quando o aplicativo cresce para incluir mais recursos. A habilidade aqui é escrever código para testar os requisitos de recursos automaticamente, sem intervenção humana;

- Testes Contínuos:
  - A entrega contínua é uma prática em que os recursos são entregues de forma incremental aos usuários finais em ciclos curtos. Para alimentar a entrega contínua, temos que testar o aplicativo continuamente para que ele esteja sempre pronto para ser lançado. Automatiza-se e integra-se as verificações de qualidade em seus pipelines de CI/CD, que são executados com frequência para facilitar o processo de teste;
- Testes de Dados:
  - Quando os dados dos usuários são perdidos ou o aplicativo mostra os dados errados para os usuários finais, eles perdem a confiança no próprio aplicativo. A habilidade de teste de dados requer conhecimento sobre os diferentes tipos de armazenamento de dados e sistemas de processamento normalmente usados em aplicações web e móveis;
- Testes Visuais:
  - A aparência do aplicativo é um dos principais contribuintes para o valor da marca da empresa, especialmente quando se trata de produtos B2C usados por milhões. Portanto, é essencial validar que os usuários finais tenham uma experiência visual harmoniosa e agradável, realizando testes visuais do aplicativo. O teste visual requer uma compreensão de como os componentes da interface do usuário interagem entre si e com o navegador para aplicativos da web. Essas verificações também podem ser automatizadas, usando ferramentas diferentes daquelas usadas para testes funcionais automatizados;
- Testes de Segurança:
  - As violações de segurança tornaram-se muito prevalentes no mundo de hoje, e nem mesmo gigantes como Facebook e Twitter

estão excluídos de tais ataques. Os problemas de segurança têm um alto custo tanto para os usuários finais quanto para os negócios em termos de perda ou exposição de informações confidenciais, penalidades legais e reputação da marca;

- Testes de Performance (Desempenho):
    - A habilidade do teste de desempenho envolve a medição de um conjunto de indicadores-chave de desempenho em diferentes camadas do aplicativo. Os testes de desempenho também podem ser automatizados e integrados aos pipelines de CI para obter feedback contínuo;
- Testes de Acessibilidade:
    - Tornar aplicativos acessíveis a pessoas com deficiências permanentes ou temporárias não é apenas obrigatório por regulamentos legais em muitos países, mas também eticamente a coisa certa a fazer. Para adquirir a habilidade de teste de acessibilidade, devemos primeiro entender os padrões de acessibilidade exigidos por lei. Podemos então usar ferramentas de auditoria de acessibilidade manuais e automatizadas para validar se esses padrões são atendidos;
- Testes de Requisitos Multifuncionais:
    - Os requisitos de qualidade, como disponibilidade, escalabilidade, capacidade de manutenção, observabilidade, e assim por diante, são chamados de requisitos multifuncionais (CFRs) de um aplicativo. Embora os requisitos funcionais geralmente chamem mais atenção, são os CFRs que imbuem a qualidade no aplicativo, e a falha em testá-los levará a equipes de negócios ou de software insatisfeitas, usuários finais ou ambos. Portanto, a habilidade de teste CFR é uma habilidade de teste fundamental;
- Testes Móveis:
    - A explosão no número de aplicativos móveis decorre principalmente do aumento do uso de dispositivos móveis. Portanto, a capacidade de testar aplicativos móveis e a compatibilidade de sites em dispositivos móveis é uma habilidade crítica na atualidade, considerando características e funcionalidades diferentes de aplicações web.

Todas essas dez habilidades da Stack de testes permitem que o software seja testado desde a definição de seu escopo até as versões dos aplicativos da web e móveis implantadas. A qualidade do software não pode mais ser equiparada apenas à funcionalidade livre de bugs. A stack de testes refere-se ao teste de todas as dimensões de qualidade de um aplicativo de forma holística para cada camada, fornecendo, assim, o software de alta qualidade.

1.2 Profissionais de Testes dentro do Ciclo de Vida do Software

Para conversarmos sobre os profissionais ligados a testes de software, é necessário que falemos sobre a Stack de Testes apresentado no item 1.1. Uma stack de testes atinge seu objetivo de fornecer software de alta qualidade quando as equipes deslocam o teste para a esquerda (*Shift-left*) para que ele comece em paralelo com o projeto e durante a análise, bem como continue durante todo o ciclo de entrega. O teste *Shift-left* tem como princípio que a qualidade seja responsabilidade de todo o time, pois exige que cada membro se responsabilize pela realização de certas verificações de qualidade em diferentes fases da entrega. Isso exige que todos os membros do time se qualifiquem, adquirindo habilidades de teste relevantes em níveis variados de competência.

As equipes de desenvolvimento ágil têm a principal responsabilidade pela qualidade em projetos ágeis. A responsabilidade pela qualidade é uma extensão das responsabilidades e liberdades que vêm com a autogestão. Quando a equipe de desenvolvimento é livre para determinar seus métodos de desenvolvimento, a equipe de desenvolvimento também é responsável por garantir que esses métodos resultem em um trabalho de qualidade.

As organizações geralmente referem-se à gestão da qualidade como um todo como garantia de qualidade, ou QA. Cada empresa emprega sua área de qualidade de acordo com seu tamanho, seus objetivos e suas necessidades, mas, geralmente, tem-se uma área de controle de qualidade com analistas de testes, gerentes de controle de qualidade, analistas de controle de qualidade e todos os outros tipos de títulos com prefixo de controle de qualidade para se referir a pessoas responsáveis por atividades de qualidade.

Os outros membros da equipe Scrum – o SM (Scrum Master) e o PO (Product Owner) – também desempenham papéis no gerenciamento da qualidade. Os POs fornecem esclarecimentos sobre os requisitos e também aceitam esses requisitos como sendo feitos ao longo de cada sprint. Os Scrum

Masters ajudam a garantir que as equipes de desenvolvimento tenham um ambiente de trabalho em que as pessoas nas equipes de desenvolvimento possam trabalhar com o melhor de suas habilidades.

Felizmente, as abordagens ágeis têm várias maneiras de ajudar as equipes na criação de produtos de qualidade. A gestão da qualidade é uma parte diária dos projetos ágeis. As equipes Scrum executam projetos ágeis em sprints, ciclos de desenvolvimento curtos que duram de uma a quatro semanas. Cada ciclo inclui atividades das diferentes fases de um projeto tradicional: requisitos, design, desenvolvimento, teste e integração para implantação. As equipes Scrum testam ao longo de cada sprint.

Quando as equipes de desenvolvimento testam ao longo de cada sprint, elas conseguem encontrar e corrigir bugs muito rapidamente. Com o gerenciamento ágil de projetos, as equipes de desenvolvimento criam requisitos de produto, testam imediatamente esses requisitos e corrigem quaisquer problemas imediatamente. Em vez de tentar se lembrar de como consertar algo que criaram semanas ou meses antes, as equipes de desenvolvimento estão, no máximo, corrigindo o requisito em que trabalharam um ou dois dias antes.

Testar quase todos os dias em um projeto ágil é uma ótima maneira de garantir a qualidade do produto. Outra maneira de garantir a qualidade do produto é criar um produto melhor desde o início. O foco na excelência técnica e no bom design faze parte dos 12 Princípios Ágeis, porque a excelência técnica e o bom design levam a produtos valiosos.

Outras empresas podem subestimar a excelência técnica. As equipes de projetos ágeis dessas empresas podem ter dificuldades ao tentar justificar treinamentos ou ferramentas que ajudarão a criar produtos melhores. Algumas empresas não fazem a conexão entre boa tecnologia, bons produtos e lucratividade.

Scrum masters e Product Owners podem ter que educar suas empresas sobre a importância de uma boa tecnologia e design, podendo ser necessário fazer lobby para obter as equipes de desenvolvimento e o que elas precisam para criar um ótimo produto.

É sempre bom relembrar que, quando damos atenção ao controle de qualidade do software desde o início do ciclo de vida do software, os custos sempre serão menores nas etapas iniciais do projeto, como podemos verificar na tabela 1 e na figura 4.

Tabela 1 – Custo comparativo para correção de erros

Estágio no qual o erro foi encontrado	Comparativo de Custos
1. Requisitos	$1
2. Código	$10
3. Teste das Funcionalidades	$100
4. Teste do Sistema	$1.000
5. Teste de Aceitação	$10.000
6. Durante Manutenção	$100.000

Fonte: Teixeira, 2022, com base em Hambling, 2015.

Figura 4 – Custo comparativo para correção de erros

Fonte: Hambling, 2015.

1.3 FullStack QA Tester

Dentro da área de desenvolvimento de software é comum ouvirmos a função de Full Stack Developer, mas dentro da área de testes isso não é tão comum. Porém, existe a função de Full Stack QA Engineer.

O Full Stack QA Engineer/Analyst precisa ter experiência e conhecimento nas dez habilidades da Full Stack de Testes estudada no item 1.1. Como engenheiro que garante e controla a qualidade de software, ele compreende bem o processo de desenvolvimento de software e acompanha os times de

desenvolvimento, além de estar preparado para criação de documentação, User Stories, entre outros.

Uma função importante é a criação de cenários de testes, cumprindo os prazos, de acordo com os recursos e escopo do projeto, que mudam constantemente. Além disso, também cuidam de todas as novas alterações e garantem que elas sejam implementadas corretamente de acordo com a expectativa do cliente. O Full Stack também deve ter um excelente conhecimento do ciclo de vida do projeto de software, metodologias ágeis, tais como Kanban ou Scrum, preferencialmente. Dentro dessas metodologias, é preciso ter experiência nas principais cerimônias: restrospectiva, *daily*, *planning* e sprint.

Além do conhecimento em desenvolvimento, deve conhecer todo o ciclo de vida dos testes, tipos de testes, ciclos de defeitos, gerenciamento de bugs, conhecimento de testes manuais e automatizados. O ideal é ter uma certificação ISTQB, mesmo que seja de nível básico. Conhecimento de uma linguagem de programação também é necessário, pois o conhecimento em codificação auxilia no entendimento mais completo do software.

Ao trabalhar com o teste de *front-end*, o conhecimento de HTML e JavaScript permite que possa fazer inspeções, verificando tags HTML, por exemplo. O conhecimento de ferramentas de automação, como Cypress, Selênio ou MochaJS, também auxilia na cobertura de testes e testes assíncronos.

O conhecimento em SQL facilita nos testes de APIs e de banco de dados, além claro, de conhecimentos em DevOps, que permitirá avaliar o versionamento do código e monitoramento das rotinas de integração contínua. Finalmente, conhecimento em clouds como Google, Aws e Azure permite maior abrangência e verificação de todo o software implantado.

Realmente, quando olhamos todo o conhecimento e experiência que esperamos de um Full Stack QA Tester, parece-nos algo extremamente abrangente e complexo, porém, ele de fato é o profissional de qualidade que tem a bagagem mais completa para nos auxiliar na gestão da qualidade de software.

Além do Full Stack QA Tester, temo algumas variações de profissionais na área de testes, tais como:

- Software QA Tester;
- QA Engineer;
- Software Tester;

- Mobile App Tester;
- Game Tester;
- Regression Tester;
- Black Box Tester;
- QA Automation Tester;
- Web Tester;
- Software Performance Testing.

Com o crescimento e expansão do uso de software em diversos ambientes e plataformas, a especialização de analistas e engenheiros de testes faz-se necessária por conta da abrangência de ferramentas, frameworks, bibliotecas e outros recursos computacionais.

1.4 Alfa e Beta Tester

Testes Alfa e Beta concentram-se na descoberta de “bugs” em software que já passou por todas as fases de testes, validações e verificações e já está numa fase de implantação em modo Alfa o Beta.

Nesse escopo de testes, o produto, ao ser lançado, vai ganhando experiência de uso por meio de usuários em tempo real. Esses usuários vão retornando feedbacks valiosos e que geram o aumento da usabilidade do software.

Há metas e métodos para que usuários participem como Alfa e Beta testers, os quais são seguidos com base no processo alinhado ao projeto de software.

Esses usuários que se tornam testers auxiliam na economia de milhares de dólares, especialmente quando falamos de empresas grandes, como Apple, Google, entre outras.

O controle de qualidade de software de tais empresas vai acompanhando e dando o devido retorno aos times de desenvolvimento por meio de testes Alfa e Beta. Esse momento é logo após os testes de aceitação permitirem que o software seja liberado.

Tanto o teste Alfa quanto o Beta já são efetuados por usuários reais e ambiente de produção real. Tais versões facilitam também que o aplicativo seja executado por milhares de tipos de equipamentos, apresentando também quaisquer problemas em relação a questões de hardware e infraestrutura.

É normal vermos tais versões (Alfa e Beta) liberadas por empresas como a Microsoft ao lançar suas novas versões de Windows, ou com a Apple, ao lançar suas novas versões de OS. O público auxilia na correção e na melhoria de tais sistemas operacionais.

TEMA 2 – ESTRUTURA DE TESTES

O modelo ágil diz que não há necessidade da criação de um plano de testes para realmente criar um plano de testes muito elaborado, mas é preciso ter uma compreensão clara de quais testes são necessários, comunicando o escopo com a equipe de lançamento e salvando-o em uma área compartilhada é importante.

A seguir, veremos o plano mestre de testes que corresponde à estratégia em testes segundo o modelo ágil contendo objetivos (Nader-Rezvani, 2018):

- Fornecer uma estrutura para testes dentro do desenvolvimento do ciclo de vida para que o esforço permaneça focado e dentro do cronograma;
- Identificar as tarefas necessárias para preparar e conduzir testes manuais e automatizados de nível de lançamento entre as equipes;
- Renunciar a uma abordagem tradicional de testes em silos pela comunicação clara e frequente de coordenação com todos os membros da equipe de lançamento ágil;
- Os itens a serem comunicados incluem quais equipes adicionaram histórias específicas para o backlog do produto e, posteriormente, para sua lista de pendências de iteração. Isso inclui a coordenação de atividades de teste para evitar a duplicação de esforços e reduzir a oportunidade de ocorrência de erros;
- Garantir o compartilhamento de dados de teste em todas as camadas de teste;
- Representam requisitos funcionais e não funcionais do produto/componente e garantem que esses requisitos foram atendidos;
- Se o projeto tiver um componente de terceiros ou de código aberto, então as histórias derivadas devem indicar claramente quem deve realizar testes funcionais, de estresse e de sistema e deve especificar os critérios de aceitação para o componente.

Com o desenvolvimento da estratégia de testes, é necessário considerar algumas questões importantes (Nader-Rezvani, 2018):

- Expectativas de marcos. Por exemplo, pode haver requisitos específicos para alcançar grandes marcos, como demonstração do cliente, certificação de plataforma, infraestrutura requisitos etc.;
- Garantir que os objetivos de incremento/liberação do programa sejam claramente compreendidos e levados em consideração ao criar uma estratégia de teste holística;
- Usar uma abordagem de desenvolvimento orientada a testes (TDD), com forte ênfase na automação desde os estágios iniciais do lançamento;
- Determinar as metodologias de teste que serão mais eficazes em encontrar defeitos críticos com antecedência e eficiência;
- Realizar testes de API ou testes de linha de comando para testar o código no início do ciclo em vez de confiar apenas na “caixa preta” teste. Considerar a melhor forma de alavancar e integrar ferramentas estáticas de análise de código para detecções precoces;
- As abordagens podem incluir testes baseados em risco, baseados em mudanças, testes baseados em requisitos, testes baseados em cenários, testes baseados em história, testes baseados em modelo, testes baseados em ataque testes, injeção de falhas, testes exploratórios, e assim por diante;
- Ter expertise nas ferramentas disponíveis para testes;
- Considerar fortemente testes baseados em mudanças;
- Ter em mente que pode não haver tempo suficiente para executar todo o conjunto de testes de regressão.

Também é fundamental entender completamente os diferentes tipos de testes e suas várias camadas para aplicar a melhor estratégia de teste para as diferentes etapas de desenvolvimento de recursos. A fundação começa no nível de teste de unidade no inferior, e à medida que o código torna-se mais estável, espera-se que camadas adicionais sejam projetadas e executadas.

Uma combinação de testes automatizados contínuos, misturando técnicas de teste, integrando a cobertura de código e utilizando análises para se concentrar em regressão dará uma estratégia de teste sólida. Sem uma estratégia de automação de teste adequada, um modelo de teste contínuo não será eficaz ou mesmo possível. Também é importante considerar a integração

da análise de código estático no modelo de teste para que a detecção precoce seja possível.

2.1 Cobertura de Testes em Camadas

É importante que as equipes reconheçam que é fisicamente impossível profissionais de teste para validar cada configuração e cenário. Definindo estratégias e o nível apropriado de testes em cada estágio de desenvolvimento é fundamental para garantir que a cobertura adequada seja alcançada. O risco de não cobrir certas combinações de teste terá que ser discutido e internalizado pelas equipes de produto. As variações da Pirâmide de Teste de Martin Fowler e Michael Cohn representam as ideia sobre a cobertura de testes, como se visualiza na figura 5.

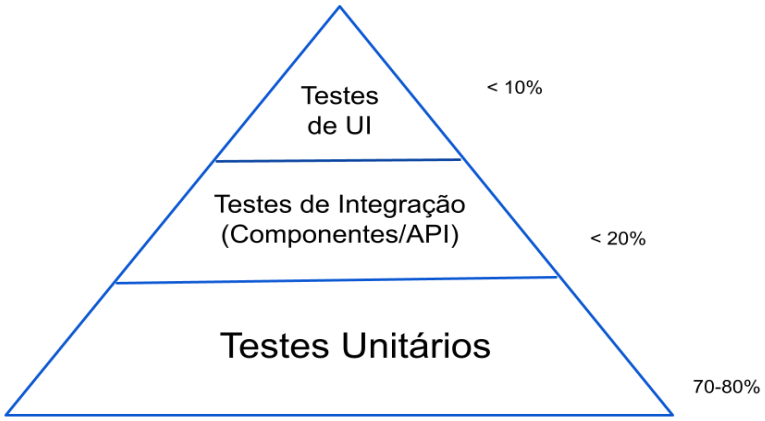
A principal conclusão é que as equipes devem se concentrar em produzir muito mais testes de unidade de baixo nível do que testes automatizados e manuais baseados em interface de usuário de alto nível, incluindo testes exploratórios e de usabilidade. Essa estrutura de camada de teste é usada para integração contínua e entrega contínua, que são essenciais para desbloquear os benefícios do teste ágil.

Na realidade, muitas organizações têm uma estrutura de teste automatizada semelhante à mostrada na Figura 6, em que a maior parte do foco está no manual e em testes automatizados de interface do usuário. Isso é conhecido como o Ice Cream Cone of Testing!

Isso não permite um modelo de CI/CD bem-sucedido, que é um pré-requisito para ter um modelo de teste contínuo com boa cobertura. Essas equipes não investem tempo no desenvolvimento de testes unitários porque não é “fácil” fazê-lo. O que eles não percebem é que com a implementação sólida de cobertura de teste de unidade automatizada, economiza-se muito dinheiro e tempo encontrando problemas no início do ciclo de desenvolvimento.

Agora, explicando um pouco sobre cada tipo de teste, são apontadas tanto a pirâmide de testes quanto a cobertura de testes em camadas nas figuras 5 e 6, respectivamente.

Figura 5 – Pirâmide de Testes



Fonte: Nader-Rezvani, 2018.

Figura 6 – Cobertura de Testes em Camada

Fonte: Nader-Rezvani, 2018.

- Teste de unidade:
    - Testes de unidade para todos os novos códigos precisam fazer parte da Definição de Pronto e, dessa forma, são obrigatórios. As ferramentas de cobertura de código validarão a eficácia;
- Teste de recurso/funcional:
    - Essa camada de teste concentra-se nos recursos de um módulo, componente ou nível do produto. Ele garante que o teste de regressão impactado seja incorporado ao teste de novos recursos;
- Teste de regressão:

- Esse nível concentra-se em garantir que novas funcionalidades/recursos não introduzam bugs ou impactem negativamente a funcionalidade do produto existente. Considere usar testes automatizados contínuos focados nas áreas de mudança, como novas funcionalidades são implementadas. Orientação para áreas de teste de regressão geralmente vem do arquiteto de QA, com a contribuição de toda a equipe Scrum. Estratégia de regressão impactada para garantir que novos recursos não violem os existentes funcionalidades devem ser discutidas em detalhes;

- Teste de Documentação:
  - Testes serão feitos para verificar se a documentação do produto visível ao usuário é suficiente para instalar, administrar e usar o produto. Cada equipe Scrum tem a responsabilidade de revisar a documentação quanto à integridade e precisão, conforme nova funcionalidade é desenvolvida;
- Teste do sistema:
  - Esse nível concentra-se em testar o sistema integrado de módulos ou componentes dentro de uma área de produto para verificar se produz os resultados desejados. É importante descrever a abordagem de teste do sistema para as áreas no escopo e garantir abordagem ao longo do lançamento;
- Teste de integração:
  - Esse nível concentra-se em testar o sistema integrado de produtos ou soluções (incluindo software de terceiros) para verificar se os produtos integrados atendem aos requisitos desejados. Ele fornece uma abordagem de teste para garantir o produto e garante mecanismos uniformes para integração com produtos, bem como revisão da metodologia de teste de integração com toda a versão. A equipe fornecerá uma abordagem de teste consistente em várias equipes de scrum;
- Teste do Cenário do Cliente E2E:
  - Também conhecido como Business Process Testing ou Model-Based Testing, deve incluir clientes externos e internos;
- Externo:

- Estabeleça um programa de validação do cliente para envolver pelo menos três clientes durante todo o ciclo de lançamento quando novos recursos são disponibilizados para buscar feedback cedo e frequentemente. Isso permitirá que os clientes executem seu E2E em casos de uso, destaquem desafios ou forneçam opções para aprimorar os recursos. Algumas equipes construíram parcerias com clientes específicos e têm acesso aos detalhes do ambiente e conjunto de dados para executar casos de uso genéricos. Essa é uma ótima maneira de testar novos recursos, garantir que não haja regressões e receber feedback antecipado;

- Interno:
  - Testes exploratórios por clientes internos (ou seja, Suporte, Pré-vendas, Services e SWAT) é uma ótima abordagem a seguir aqui;
- Testes não funcionais (habilidades):
  - Testes que determinam até que ponto o aplicativo atende aos requisitos não funcionais esperados são, muitas vezes, referidos como “habilidades”. A seguir, estão os tipos específicos de “habilidades” que as equipes devem revisar e considerar conforme os requisitos para sua aplicação;
- Teste de acessibilidade:
  - Testes que serão feitos para garantir que o produto/componente esteja em conformidade com as normas de acessibilidade. O teste será concluído com o envio de um relatório de conformidade;
- Teste de interoperabilidade:
  - Testes para garantir que o produto é capaz de coexistir de forma graciosa com componentes e agentes comuns que provavelmente estão na mesma lógica em ambientes comuns de clientes. Isso inclui a capacidade de ser instalado e desinstalado sem afetar outros componentes, bem como ser capaz de compartilhar os recursos disponíveis de forma eficiente;
- Teste de segurança:
  - Existem vários tipos de metodologias de teste de segurança a serem usadas. Análise de código estático de vulnerabilidade e segurança, testes de penetração, ética hacking e avaliação de risco estão entre as principais técnicas usadas pela maioria das

empresas. Isso é para garantir que o produto/componente passe por análise adequada de segurança e risco para documentar vulnerabilidades e formular um plano de remediação adequado. O teste de segurança não é mais considerado uma reflexão tardia e deve ser integrado ao ciclo de desenvolvimento de código. As vulnerabilidades de segurança devem ser priorizadas e abordadas ao longo do caminho;

- Teste de localização:
    - A localizabilidade consiste em internacionalização (i18n) e localização (l10n) teste. Depois que o conteúdo é traduzido, é importante fazer a verificação do conteúdo para garantir que o material seja traduzido adequadamente de acordo com o idioma e a cultura;
- Teste de prontidão para internacionalização:
    - Testes que verificam se o componente/recurso e, em geral, o produto como um todo está devidamente internacionalizado para que possa ser localizado de forma rápida e econômica em qualquer idioma;
- Teste de localização:
    - Testes que verificam se o material traduzido está linguisticamente correto e apropriado e que o produto/componente localizado pode ser instalado, parece e se comporta conforme o esperado em ambientes localizados e em inglês dos EUA (sistema operacional, navegador, banco de dados, terceiros e tecnologias subjacentes). Embora o foco esteja principalmente na interface do usuário da web, é importante também planejar a tradução da documentação do produto, ajuda online, interface de linha de comando, arquivos de log, e assim por diante;
- Teste de suporte:
    - Testes que garantem que seu produto implemente um conjunto conhecido e consistente de capacidades para permitir que os clientes consumam produtos de forma eficaz em todo o ciclo de vida da implantação. Por exemplo, ao fornecer erro/aviso significativo, mensagens permitirão que os clientes e o departamento de suporte resolvam problemas mais rápido. Uma

grande consideração de suporte é incorporar a telemetria ou uma chamada Recurso de casa. Isso permite um diagnóstico mais rápido e, muitas vezes, leva a melhorias, pois resolver os problemas dos clientes antes que eles o percebam é uma ótima maneira de melhorar a satisfação do cliente!

- Teste de capacidade de atualização:
    - Testes que garantem que seu produto possa ser atualizado para a nova versão sem um grande investimento de esforço/hardware por parte do cliente. Os produtos básicos e as personalizações com suporte devem ser atualizados dentro de prazos predefinidos (esse prazo precisa ser discutido e acordado durante o planejamento da liberação);
- Teste de usabilidade:
    - Teste do aplicativo para garantir que os usuários pretendidos de um sistema possam realizar suas tarefas de forma eficiente, eficaz e satisfatória. Testando usabilidade é realizado pré-lançamento para que quaisquer problemas significativos sejam identificados, também como validar o aplicativo em um limite de iteração lógica para novos recursos;
- Testes de desempenho e escalabilidade:
    - Testes para garantir que o comportamento do sistema seja o desejado durante a carga normal e para determinar o limite superior de transações ou cálculos permitidos e o nível do produto. Isso inclui as abordagens de teste de escalabilidade para vários níveis de teste do produto. Por exemplo, testes que verificam o comportamento do programa durante picos intensos de atividade, ou imprimindo 1.000 documentos de uma vez ou abrindo o número máximo de arquivos. É importante, no entanto, estabelecer limites adequados e aceitáveis com antecedência. Com certas aplicações e sem quaisquer desses limites, o sistema eventualmente falha. É melhor relaxar os limites, como nos casos de uso do cliente, que são mais bem compreendidos, se necessário.

2.2 Evolução de Testes ao Longo do Tempo

Enquanto o conjunto anterior de diretrizes, tipos de testes e planejamento será relevante por muitos anos, os departamentos de desenvolvimento devem manter-se sempre atualizados com as novas tecnologias emergentes para atualizar continuamente e ajustar sua estratégia de teste.

Nos últimos anos, as tendências mencionadas a seguir e tecnologias ganharam muita publicidade.

- **Internet das Coisas (IoT)**
  - Por melhor que pareça, o mundo dos dispositivos conectados e a integração pesadelo que vem com ele trará desafios interessantes para lidar. Como exemplo, vulnerabilidades resultantes de produtos conectados criarão muitas dimensões adicionais a serem consideradas em sua estratégia de teste;
- **Big Data:**
  - Clientes e usuários coletam e carregam terabytes de dados em várias plataformas. A consideração cuidadosa para testes com uma grande quantidade de dados precisa ser feita para um melhor alinhamento com o ambiente semelhante ao do cliente. Lembre-se de que nem sempre é possível construir um ambiente de cliente exato para validação interna, mas chegar o mais próximo possível da configuração do cliente possível é o objetivo aqui;
- **Usuários móveis e automação de testes:**
  - Cada vez mais clientes exigem suporte móvel para seus aplicativos. Testando aplicativos móveis e considerando ângulos únicos em relação ao número de atualizações regulares de software e tipos de dispositivos que merecem suas próprias estratégias e planejamento de testes;
- **API e microsserviço:**
  - Microsserviços e ênfase em um design rico baseado em API encaixam-se bem com o Agile disciplina e metodologia. Existem muitas diferenças em relação ao teste tradicional de todo o sistema em uma abordagem em cascata. Receber pedaços e peças disponíveis para teste exigirá melhor planejamento, coordenação e otimização do ambiente utilizado. Cada serviço independente terá

de ser testado separadamente e, em seguida, como parte de uma estrutura interconectada. A melhor abordagem aqui é voltar ao básico da Pirâmide de Teste e começar com uma forte estratégia de teste de unidade, seguida por testes funcionais, de integração e E2E, de que já falamos;

- **Adoção de ferramentas de código aberto:**
    - Ferramentas de código aberto têm sido populares em muitos setores para testar seus formulários. Os dias em que uma ferramenta era usada para testar todo o aplicativo se foram, com otimização, determina-se qual ferramenta de código aberto usar e qual camada de teste pode reduzir custos e melhorar a produtividade;
- **IA e aprendizado de máquina:**
    - Muitas organizações conseguiram otimizar seus testes e cobertura em todos os níveis, aproveitando várias fontes de dados que estão disponíveis para eles. Os dados residem no gerenciamento de casos de teste e nos relatórios de defeitos existentes das equipes de sistemas (registro, detalhes de resolução e informações de regressão), com seus dados do repositório de código-fonte. A utilização desses dados pode ajudar a identificar mapa de calor e áreas problemáticas no produto. Os dados históricos são um tesouro que pode ser usado para treinar algoritmos de IA para medir riscos, classificar defeitos e prever entrega de soluções aos clientes;
- **Teste de Crowdsource:**
    - Isso está se tornando cada vez mais popular, especialmente com orçamento em constante pressão. Não ser capaz de contratar recursos internos ou contratados para fazer testes exploratórios levou algumas organizações a adotar essa abordagem. O maior desafio aqui são as preocupações de segurança que existem e precisam ser tratadas. Tais tipos de testes são evoluções dos apresentados nas figuras 5 e 6, de acordo com a inovação tecnológica a nível de hardware e infraestruturas.

TEMA 3 – GERENCIAMENTO DE TESTES

O gerenciamento de testes segue uma ideia similar ao gerenciamento de desenvolvimento de software. Devemos considerar processos de testes, tipos de ferramentas e como utilizá-las dentro do processo.

O primeiro momento seria planejar estrategicamente quais testes podemos aplicar ao processo de gerenciamento da qualidade e testes. Uma vez que decidimos quais testes vamos aplicar ao nosso processo de gestão de qualidade, então precisamos passar pela fase de planejamento e controle. Os testes devem ser bem definidos, assim como os recursos necessários para que ocorram conforme o planejado. Por exemplo, é preciso pensar em time para execução das tarefas, tempo estimado, infraestrutura e ferramentas que serão necessárias. Todas as especificações devem ser documentadas num plano de teste. E, obviamente, como estamos falando do planejamento e controle dos testes, devemos não somente aplicá-los, mas também controlá-los e gerar feedback para os times de qualidade, de desenvolvimento e infraestrutura, com a finalidade de melhoria de todo o processo e também do produto final (software) (Spillner, 2007).

Os testes estão intimamente relacionados ao processo de desenvolvimento de software, ou seja, ao seu ciclo de vida. Dentro do ciclo de vida, precisamos estabelecer como avaliar as particularidades de cada teste planejado em cada fase.

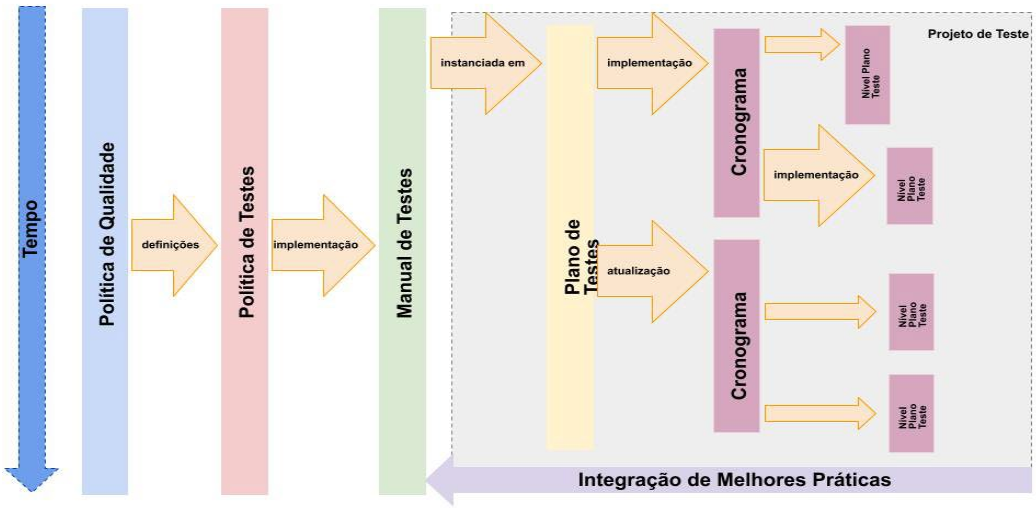
Por exemplo, mesmo na fase de requisitos, é possível adaptarmos verificações e validações que possam excluir problemas futuros em termos de má definição dos requisitos ou até mesmo estabelecimento do escopo e delimitações do projeto.

Não basta que planejemos e controlemos os testes a serem aplicados ao ciclo de vida do software, mas também é de suma importância a definição de políticas de testes, sejam eles manuais ou automatizados. E aqui falamos em termos gerais, para quaisquer projetos de software que sejam elaborados dentro da organização. O objetivo das políticas é a padronização e simplificação na criação do planejamento, cronogramas e utilização das melhores práticas para projetos atuais e futuros.

O plano do projeto de desenvolvimento de software, bem como o plano de testes, apresenta componentes básicos e a implementação a ser seguida dentro de um ciclo de vida.

A figura 7 demonstra visualmente a ordem e dependência entre políticas e planejamentos de testes dentro do ciclo de vida de testes.

Figura 7 – Da política de qualidade para o nível de plano de testes



Fonte: Spillner, 2007.

Não devemos nos esquecer dos testes que se relacionam aos dados, porque o dado é um elemento de grande importância dentro das organizações. O planejamento deve sempre ser ajustado durante todo o ciclo de vida do projeto. Com isso, o controle do processo de teste deve ter base sólida em relatórios de progresso dos testes.

Os testes de avaliação de risco são instrumentos importantes dentro da gestão de testes, pois é necessário que sejam previstos alocação de recursos e limitações de testes, minimizando os riscos.

Finalmente, times de profissionais e métricas são critérios fundamentais para que a gestão de testes seja efetiva. Lembrando que tudo aquilo que não é mensurado dificulta ou até mesmo impossibilita o gerenciamento.

TEMA 4 – MENTALIDADE DO TESTADOR E DO DESENVOLVEDOR DE SOFTWARE

As mudanças entraram na área de desenvolvimento de software desde o primeiro dia de sua existência. Não há como falarmos de computação sem

pensarmos em inovação e evolução de tecnologias. Porém, mesmo diante de tal situação, há muitos profissionais que resistem ainda às mudanças, até mesmo pelo medo do desconhecido, falta de informações, medo de perder poder ou recursos. Enfim, a abordagem construtiva à resistência é reconhecermos que o importante é não ter medo do novo.

Se olharmos para o histórico da área de desenvolvimento de software, bem como para qualidade e controle de processos e de software, podemos observar que saímos de um modelo “caótico”, ou seja, sem nenhum tipo de metodologia de desenvolvimento, e passamos pelos modelos tradicionais e pelas metodologias ágeis, e há quem diga que estamos no modelo pós-ágil.

Na fase considerada tradicional, e aí vamos pular a fase “sem nenhum tipo de metodologia”, o mindset dos antigos analistas de testes era de efetuar a tradução da documentação em casos de testes, sem contato com o código e aguardando que os desenvolvedores entregassem o sistema para o início dos testes e escrita dos bugs detalhadamente, pois não havia contato entre analistas de testes e desenvolvedores.

Com a abordagem ágil, o analista de testes já muda o foco. Agora, ele analisa, testa e automatiza casos de testes. Também conhece e usa testes exploratórios, ajudando na identificação de riscos da aplicação ao escrever testes de aceitação. Porém, as User Stories ainda são provenientes dos engenheiros de QA.

O que chamamos hoje de modelo pós-ágil permite que o analista de testes trabalhe imerso com os times de desenvolvimento, demonstrando como testar e incentivando a melhoria da qualidade.

Ao revisar o código, contribui como desenvolvimento, incluindo na correção dos bugs. Esse perfil de analista de testes também utiliza testes automatizados das várias camadas de cobertura de testes.

Quando aprendemos sobre todos os tipos de possibilidades e necessidades de testes que temos na atualidade, em virtude da evolução das tecnologias, o analista de testes, além de estar envolvido diretamente no ciclo de desenvolvimento de software com os desenvolvedores, tem se especializado também em tecnologias, trazendo consigo a especialização nos títulos dos analistas e engenheiros de testes, como Mobile App Tester, o qual se dedica, muitas vezes, a uma das duas plataformas (Android ou Apple).

Também temos os Analistas de Game Tester, que são envolvidos não somente com o desenvolvimento de games, mas também conhecem profundamente o negócio e as modalidades de games existentes, que variam de acordo com faixa etária, com os consoles, com atividades motoras e com o gênero, podendo ser intelectuais, de competição e de azar, sendo caracterizados como jogos eletrônicos independentes, arcades, advergames, educacionais, sociais, de produção comercial, 2D, 3D, de ação, de aventura, de estratégias, RPGs, de esportes, de corridas, online, de simulação, de tabuleiro, casuais, entre tantos outros tipos. Somente nesse universo de games é compreensível que o analista de testes de fato seja também um “jogador”, especialmente do tipo de jogo para o qual ele atua profissionalmente.

Há ainda analistas e engenheiros de testes que estão mais envolvidos com ferramentas de automação, como os QA Automation Testers, os analistas que estão focados em aplicações Web, como B2C, B2B, entre outros, designados como Web Testers. E outra área que tem evoluído e trazido consigo outro título de profissional de testes está diretamente ligada com a infraestrutura responsável pela implantação do software. Esse analista é chamado de Software Performance Tester, responsável pela verificação e validação de testes em relação à experiência de uso dos usuários em relação ao desempenho, segurança e outras características que envolvam toda a área de DevOps e SRE.

Falamos aqui apenas de algumas especializações de analistas e engenheiros de testes que estão surgindo em decorrência de toda evolução de tecnologias, frameworks, finalidades do software, características de hardware e infraestrutura.

Com o advento de IoT, IA, Big Data e outras tecnologias, não mais futurísticas, mas do momento, também surge a necessidade de profissionais de testes mais alinhados a tais tipos de software.

TEMA 5 – AUTOMAÇÃO DE TESTES

Teste de software é o processo usado para validar se a solução de software ou produto atende aos requisitos e expectativas. O teste de software também visa encontrar defeitos e demonstrar que o software é adequado para o propósito. Existem inúmeras metodologias, tipos e técnicas de teste disponíveis para validar os requisitos funcionais e não funcionais (Boby, 2021).

Porém, o volume de testes repetitivos de software aumenta sensivelmente em decorrência do número e diversidade de tipos de testes. A automação de teste é um processo que se utiliza de outro software para automatização de ações manuais do usuário executadas no aplicativo ou de testes manuais. É uma boa ideia validar o software com a ajuda de outro software, quando o teste manual não é possível ou demorado.

O teste automatizado por meio de um software permite executar testes, incluindo a comparação e o relatório de resultados reais com o resultado previsto. O teste é necessário para todos os tipos de software e há várias instâncias de aplicações/sistemas que entraram em operação sem testes apropriados e acabaram com defeitos, causando problemas financeiros e de enfraquecimento da marca/software no mercado. O teste é uma atividade central em qualquer desenvolvimento de solução e é independente das abordagens do ciclo de vida de desenvolvimento de software (SDLC).

Projetos de desenvolvimento de software apresentam três restrições básicas: custo, tempo e escopo (Figura 8).

Figura 8 – Restrições em projetos de software

Fonte: Boby, 2021.

O escopo economiza tempo de teste, porém, apenas um conjunto de recursos é testado. O tempo ajuda na obtenção de confiança na qualidade, mas seu custo é maior. E o custo é diretamente proporcional ao tempo e ao escopo. Tais restrições triplas são acompanhadas desde o primeiro dia do início do projeto de desenvolvimento de software.

A automação de teste é um projeto de desenvolvimento de software que inclui a maioria das fases do ciclo de vida de desenvolvimento de software. A automação de teste é amplamente aprimorada por estruturas de automação de teste.

Normalmente, testes automatizados seguem uma estrutura denominada framework, que garante um conjunto de diretrizes para produção de resultados importantes para automação de testes. Geralmente, um framework de testes fornece uma estrutura para ferramentas de automação que atendem à sua finalidade. A maioria das ferramentas de automação de testes fornece uma estrutura padrão para automação.

Existem várias maneiras de tornar o teste de software eficiente, e a automação de testes é uma delas. Ela é fundamental para reduzir o esforço de teste, mas não reduz dificuldades para todas as atividades deste. A automação de teste pode ser introduzida em diferentes estágios e fases do ciclo de teste, como:

- Desenvolvimento de produtos ou implementação de soluções;
- Gerenciamento de testes;
- Testes funcionais e de regressão;
- Suporte;
- Geração de testes;
- Geração de dados de testes;
- Inspeção e avaliação dos resultados dos testes;
- Observância.

É importante observar alguns conselhos práticos para utilização efetiva de testes automatizados.

- Projetar casos de testes para depois automatizá-los, nunca o contrário;
- É muito difícil a automação de absolutamente todos os testes;
- Ferramentas de testes não são estratégias de testes;
- As estratégias de testes devem ser baseadas no contexto do projeto e em seus requisitos;
- Melhorar gradativamente os scripts de automação para que não sejam apenas repetidos;
- Aquilo que não é importante não deve ser automatizado;
- Testes manuais que não sejam possíveis de repetir não são adequados para automação;
- Testes de regressão não são para a busca de novos bugs;
- As UIs geralmente são modificadas após a aplicação de testes automatizados.

Automatizar testes de software segue processos, assim como o desenvolvimento do software, considerando-se projeto, desenvolvimento, execução e outras atividades, além da preocupação com a facilidade nas revisões de tais automações.

A automação de testes é amplamente utilizada em vários setores e aplicações, produzindo ótimos resultados. Certamente, ao ser bem aplicada, auxilia na redução de custos, tempo e prazos.

FINALIZANDO

O sucesso da aplicação de testes no ciclo de vida de projetos de software encontra-se em toda a evolução de técnicas, ferramentas e metodologias, porém, o profissional de testes é a maior revolução dentro da área de testes.

Compreender o papel e a importância dos profissionais de testes de software nos traz uma nova visão e concepção da importância dos testes, bem como o motivo pelo qual agora falamos em gestão de testes, engenheiro de testes e QA, analistas de testes, alguns tão especializados que tratam apenas dos testes específicos de aplicações móveis, para web, games, entre outros.

A criação de estratégias e planejamento de testes de software é importante, mas o controle por meio de métricas e relatórios causa o amadurecimento dos processos que poderão ser replicados em novos projetos de desenvolvimento de software.

As falhas de software, dependendo da sua aplicação, pode levar uma empresa a perder dinheiro, tempo e reputação empresarial, bem como prejuízos.

O teste insuficiente ou o tipo errado de teste feito pode custar muito mais caro para uma empresa do que todo o planejamento e implantação de garantia da qualidade com o envolvimento de testes de software.

Profissionais cada vez mais específicos para gestão e criação de testes mais eficazes garantem que produtos de software estejam mais alinhados com a qualidade esperada pelos clientes.

REFERÊNCIAS

BOBY, J. **Test automation**: a manager's guide. O'Reilly Media, Inc., 2021.

HAMBLING, B. et al. **Software testing**: an ISTQB-BCS certified tester foundation guide. 3. ed. Swindon: BCS Learning & Development, 2015.

NADER-REZVANI, N. **An executive's guide to software quality in an agile organization**: a continuous improvement journey. Los Altos: CA Press/Apress, 2019.

SPILLNER, A. **Software testing practice**: test management. O'Reilly Media, Inc., 2007.