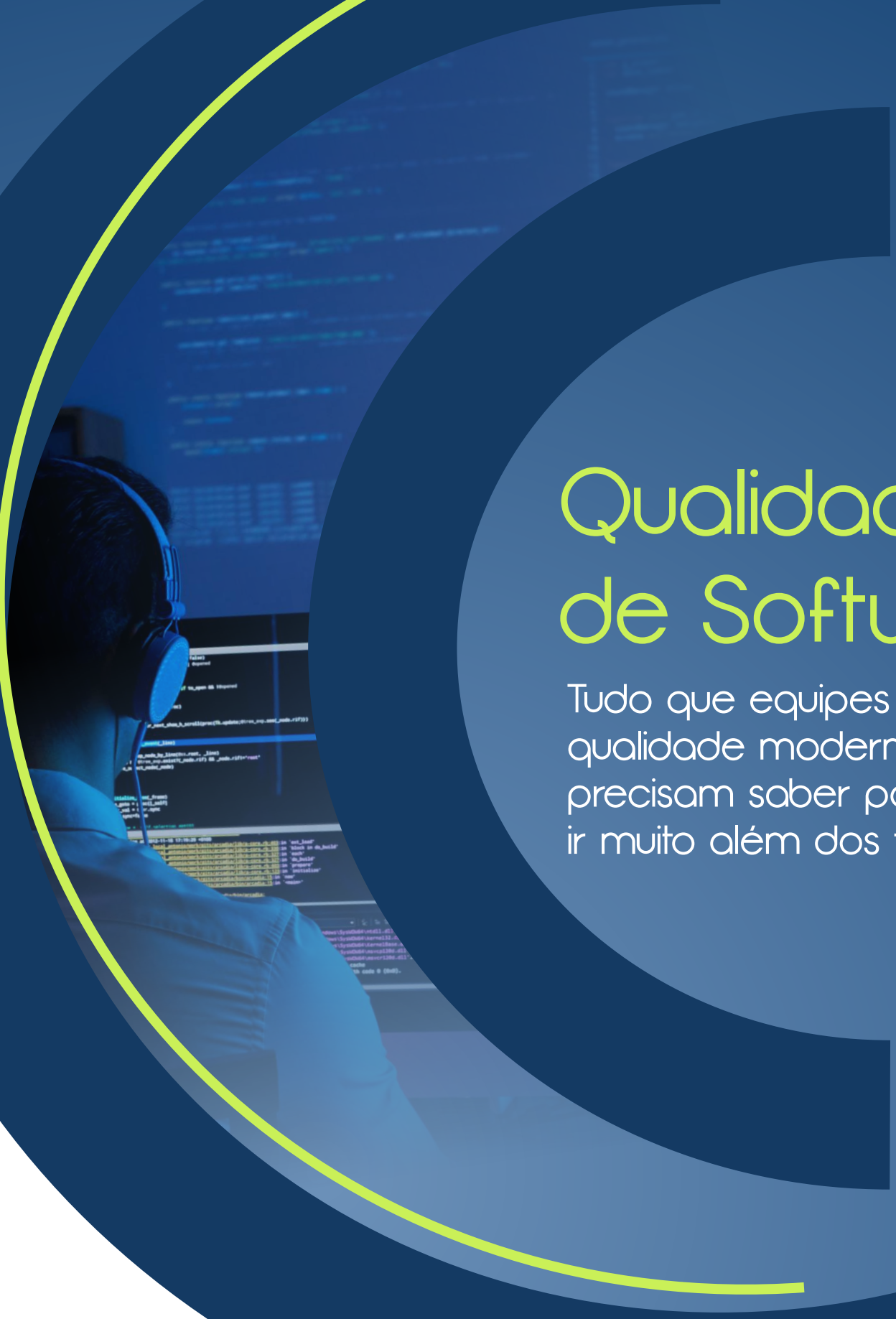


Qualidade de Software

Tudo que equipes de qualidade modernas precisam saber para ir muito além dos testes

DevTrends

Introdução

É cada vez maior e mais notório o aumento da influência da tecnologia nas mais diversas áreas da sociedade. Um processo aparentemente irreversível. O nível de dependência de vários setores da sociedade com relação aos sistemas de informação está em constante crescimento.

Assim, uma estruturação do setor de Tecnologia da Informação com o objetivo de promover o aumento da qualidade e confiabilidade de seus produtos não só se tornou um diferencial competitivo, mas também pré-requisito para sua existência.

Ao mesmo tempo, uma miríade de novas metodologias, ferramentas de produtividade e soluções vêm surgindo ao longo dos últimos anos para auxiliar os profissionais da área de TI na consolidação de um processo eficiente e que garanta a qualidade de seus produtos. Esse processo de organização encontra um paralelo na história do setor industrial, quando, em meados do século XVIII, se empregava um processo produtivo artesanal e primitivo, que foi, ao longo de dois séculos, evoluindo até culminar nas fábricas totalmente robotizadas que conhecemos.

Seguindo esses mesmos passos, a área de Engenharia de Software chega ao início do século XXI com um enorme desafio pela frente: o de consolidar um processo que assegure total qualidade a seus produtos e serviços, acompanhando a velocidade das mudanças tecnológicas atuais, as milhares de soluções disponíveis no mercado e a divergência de interesses das empresas envolvidas na área.

Diante desse cenário, verificamos esforços de vários profissionais em todo o mundo para superar esse desafio. É possível afirmar que a área de testes de software passa por um momento de dualidade onde tornou-se mais difícil e mais fácil do que nunca.

Mais difícil devido à vasta gama de linguagens de programação, sistemas operacionais e plataformas de hardware que surgiram nas últimas décadas. Se em 1970 poucas pessoas usavam computadores, hoje praticamente ninguém consegue completar um dia de trabalho sem um.

E não só existem computadores em nossas mesas, como também softwares estão presentes em quase todos os dispositivos do nosso dia a dia. Do smartphone à Alexa ao indicador digital de nível de combustível dos modelos de veículos mais recentes. Portanto, hoje, ao criar um software, potencialmente existe a possibilidade de atingir milhões de pessoas, seja positiva ou negativamente. Você pode criar algo que permita que um problema ou tarefa seja resolvido de forma eficiente, ou causar uma frustração incalculável e trazer prejuízo na forma de trabalho ou negócios perdidos.

O teste de software também se tornou mais fácil, de certa forma, porque a variedade de software e sistemas operacionais é muito mais sofisticada do que no passado, fornecendo rotinas intrínsecas e bem testadas que podem ser incorporadas através de aplicativos sem a necessidade de um alto conhecimento em programação ou linguagem específica. Seja através de drivers de automação na parte Gráficas de Usuário (GUIs), seja através de alguma aplicação que permita a gravação de scripts para simulação de ações no ambiente Web/Mobile.

O próximo passo? Passar a olhar para um novo horizonte chamado automação de testes

DevTrends

Índice

1

4

A busca pela qualidade

Nos primórdios do desenvolvimento de software, a atividade de teste era encarada como a simples tarefa de navegar pelo código e corrigir problemas já conhecidos. Tais tarefas eram realizadas pelos próprios desenvolvedores, não existindo recursos dedicados a essa atividade. Dessa forma, os testes eram somente realizados tardiamente, quando o produto já estava pronto ou quase pronto. Apesar de essa situação estar associada a uma má prática de desenvolvimento de software, ela ainda continua presente em muitas organizações.

Em 1957, o conceito Teste de Software conseguiu ampliar seus valores e se tornou um processo de detecção de erros de software, mas testar ainda era encarado como uma atividade que ocorria no final do processo de desenvolvimento.

No início da década de 1970, o processo de desenvolvimento de software passou a ter uma abordagem mais profunda com a engenharia de software sendo adotada como modelo para as universidades e organizações, porém, havia pouco consenso sobre o que viria a ser teste. Somente em 1972 é que haveria a primeira conferência formal sobre testes na Universidade da Carolina do Norte.

Foi Myers, em 1979, quem produziu um dos primeiros trabalhos mais completos e profundos sobre um processo de teste. Nesse trabalho, Myers já definia testes como um "processo de trabalho com a intenção de encontrar erros". Sua premissa era a de que se o objetivo do teste fosse apenas provar a boa funcionalidade de um aplicativo, seriam encontrados poucos defeitos, uma vez que toda a energia do processo de testes seria direcionada apenas na comprovação desse fato. Porém, se o objetivo for identificar erros, um maior número de problemas será encontrado, uma vez que os profissionais de qualidade buscarão vários cenários para avaliar o comportamento do software. Essa premissa provocou uma revolução na forma de abordar o problema, porém os testes continuavam a ser executados tardiamente.

No início dos anos 1980, surgiram os primeiros conceitos de qualidade de software. Nessa abordagem, desenvolvedores e testadores trabalhavam juntos desde o início do processo de desenvolvimento. Cada fase tinha sua atividade de conferência, de forma a garantir que a etapa estivesse completa e bem compreendida. Muitas organizações foram formadas e muitos dos padrões que utilizamos hoje nasceram nessa época, como os padrões americanos formados pelo IEEE (Institute of Electrical and Electronics Engineers) e ANSI (American National Standards Institute) e os internacionais como ISO (International Organization for Standardization. No entanto, foi o modelo CMM (Capability Maturity Model), elaborado pelo Software Engineering Institute, que ganhou maior dimensão e importância para as organizações de software, tornando-se um modelo de avaliação mais reconhecido internacionalmente.

Somente nos anos 1990 é que ferramentas de teste começaram a ser produzidas. Determinados testes que não eram possíveis de serem executados agora poderiam ser feitos através de ferramentas desenhadas para determinados objetivos. As ferramentas trariam alta produtividade e qualidade no processo de teste. Hoje, entende-se que a aquisição de ferramentas é vital ao sucesso e viabilização de um trabalho desse porte – a implantação de um processo de garantia da qualidade de software.

Cenário atual do desenvolvimento de software

É possível perceber que, de forma rápida e constante, as organizações estão aumentando sua dependência tecnológica e isso significa que suas operações internas estão sendo conduzidas e direcionadas por um conjunto cada vez maior de sistemas informatizados.

Trata-se de uma tendência natural. As organizações buscam sobreviver em um ambiente cada vez mais hostil e competitivo. A tecnologia é um instrumento para reduzir custos e ampliar sua forma de atuação. Elas estão sofisticando seus sistemas para tomar decisões cada vez mais complexas, com a intenção de ganhar eficiência e controle. Tudo isso pode ser observado através de diversos indicadores econômicos, como volume de negócios feitos pela indústria de software e hardware, proporção de horas de profissionais diretamente ligados à tecnologia por total de horas de profissionais, entre outros.

Nesse cenário, todas as variáveis envolvidas no processo de desenvolvimento de software têm um nível crescente de complexidade. Com isso, os riscos de mau funcionamento aumentam proporcionalmente à complexidade desse ambiente, tornando-se mais difícil produzir softwares com o "nível de qualidade" desejado.

Evolução do processo de qualidade e de testes de software

Evolução das Organizações Desenvolvedoras de Software			
Características	2000	2010	2021
Tamanho do Software	Médio	Grande	Muito Grande
Complexidade do Software	Média	Alta	Altíssima
Tamanho da Equipe de Desenvolvimento	Médio	Moderado	Muito Grande
Padrões e Metodologias de Desenvolvimento	Interno	Moderado	Sofisticado
Padrões e Metodologias de Qualidade e Testes	Interno	Emergente	Sofisticado
Organizações de Qualidade de Testes	Bem Poucas	Algumas	Muitas
Reconhecimento da Importância da Qualidade	Pequeno	Moderado	Significante
Tamanho da Equipe de Qualidade de Testes	Pequeno	Pequeno	Grande

Apesar do enorme avanço do desenvolvimento de software nos últimos 20 anos, muitas empresas estão presas a antigos paradigmas, o que impede seu amadurecimento no processo de desenvolvimento. Elas não percebem que seus ambientes estão cada vez mais complexos, o que exige posturas cada vez mais difíceis.

Não percebem que implantar um processo de garantia da qualidade de software não é uma opção a ser estudada, mas parte de uma estratégia de sobrevivência em um mercado cada vez mais exigente e competitivo.

3

Definindo qualidade de Software

Qualquer decisão tomada durante o processo de desenvolvimento do software pode comprometer sua qualidade final. Na verdade, o produto final do processo de desenvolvimento é exatamente o somatório de todas as decisões e realizações geradas durante todo o ciclo de desenvolvimento. Se desejarmos produzir software com alta qualidade, é necessário investir em qualidade em todos os pontos do processo.

Qualidade de Software é um processo sistemático que focaliza todas as etapas e artefatos produzidos com o objetivo de garantir a conformidade de processos e produtos, prevenindo e eliminando defeitos

Softwares mal testados provocam prejuízos enormes às organizações. Um simples erro interno do projeto poderá encadear requisições de compras desnecessárias, solicitar manutenções de equipamentos antes do período ideal, produzir estatísticas falsas de produtividade, distribuir rendimentos de forma desproporcional, entre outros. Os problemas podem afetar até a tomada de decisões de gerentes, diretores e acionistas da organização. São profissionais que se apoiam nas informações de sistemas informatizados para minimizar riscos, direcionar esforços, promover investimentos, sempre com o objetivo de tornar a organização mais eficiente e rentável. A qualidade das decisões está intimamente ligada à qualidade das informações disponibilizadas pelos diversos sistemas organizacionais.

É impossível obter um software de qualidade com processos de desenvolvimento frágeis e deficientes. Da mesma forma, é impossível estabelecer um processo de garantia da qualidade que não enfoque simultaneamente o produto tecnológico e o processo de desenvolvimento desse software. Assim, a qualidade do software pode ser compreendida em duas dimensões fundamentais: a dimensão da qualidade do processo e da qualidade do produto.

3.1 Dimensão da Qualidade do Produto

A dimensão da Qualidade do Produto é muito evidente dentro do processo de desenvolvimento de software. Qualquer empresa de software possui uma abordagem para realizar testes nos produtos de softwares gerados durante o ciclo de desenvolvimento. É comum que nos cronogramas existam fases específicas para testes, apesar de elas serem, na maior parte das vezes, substituídas por atividades de correção e manutenção do software. Essa dimensão da qualidade tem por principal objetivo garantir a qualidade do produto tecnológico gerado durante o ciclo de desenvolvimento. Todas as atividades que tenham por objetivo "estressar" telas e funcionalidades de um sistema informatizado podem ser categorizadas na dimensão da garantia da qualidade do produto tecnológico. Apesar de bastante empregadas nas organizações, o grau de eficiência dessas atividades é assustadoramente baixo, o que incentiva as empresas a substituírem tais atividades pela correção de problemas. São vários os motivos para a constatação desse fato, mas os principais pontos são: falta de planejamento das atividades de testes, ausência de testes que validem funcionalidades antigas e ausência de um processo de automação dos testes e conferências.

3.2 Medindo a qualidade através dos testes

Uma constante confusão observada nas empresas é estabelecer uma visão limitada sobre testes durante o ciclo de desenvolvimento de software. Por questões históricas, os profissionais de tecnologia somente visualizam os testes como atividades a serem aplicadas em softwares. Essa confusão é reforçada em algumas metodologias que utilizam outros nomes para representar essas atividades como revisões, inspeções e auditorias, entretanto, esses nomes apenas categorizam técnicas diferenciadas para testes de documentos que indiretamente avaliam a qualidade de uma fase do ciclo de desenvolvimento. Na verdade, o processo de qualidade de software utiliza-se dos testes em todo o ciclo de desenvolvimento, de forma a garantir tanto o processo de engenharia quanto o produto de software desenvolvido.

3.3 Testes que garatem a qualidade do processo

A qualidade dos processos pode ser medida através de testes aplicados em documentos gerados em cada fase do desenvolvimento. Como cada etapa deve produzir um conjunto de documentos, é possível estabelecer a qualidade nas várias fases do ciclo de desenvolvimento do software através da qualidade dos documentos produzidos. Se esses documentos apresentarem um alto nível de defeitos e não atingirem um nível mínimo de qualidade, é possível reconstruir o documento ou até mesmo reexecutar a fase inteira. Esses testes são conhecidos como testes de verificação. Os testes de verificação são também conhecidos como testes estáticos, pois seus principais alvos são os documentos gerados durante todo o ciclo de desenvolvimento do software. O processo de engenharia decompõe as atividades de desenvolvimento visando criar um processo sistemático de condução dos projetos de software. Para avaliar a qualidade desse processo, é necessário garantir a qualidade dos documentos produzidos em cada etapa do desenvolvimento.

3.4 Testes que garatem a qualidade do produto

A qualidade dos produtos de softwares pode ser garantida através de sistemáticas aplicações de testes nos vários estágios do desenvolvimento da aplicação. Esses testes são conhecidos como testes de validação porque, quando construímos uma unidade de software, validamos sua estrutura interna e sua aderência aos requisitos estabelecidos. Avaliamos sua integração com as demais unidades de softwares existentes, validando as interfaces de comunicação existentes entre os componentes de software. Quando determinado subsistema ou mesmo a solução estão finalizados, validamos a solução tecnológica como um todo, submetendo a testes de todas as categorias possíveis. Os procedimentos de testes aplicados diretamente em softwares são também conhecidos pelo nome de testes de software ou testes dinâmicos. Os testes de software, em sua maioria, podem sofrer um alto nível de automação, o que possibilita a criação de complexos ambientes de testes que simulam diversos cenários de utilização. Quanto mais cenários simulados, maior o nível de validação que obtemos do produto, caracterizando maior nível de qualidade do software desenvolvido.

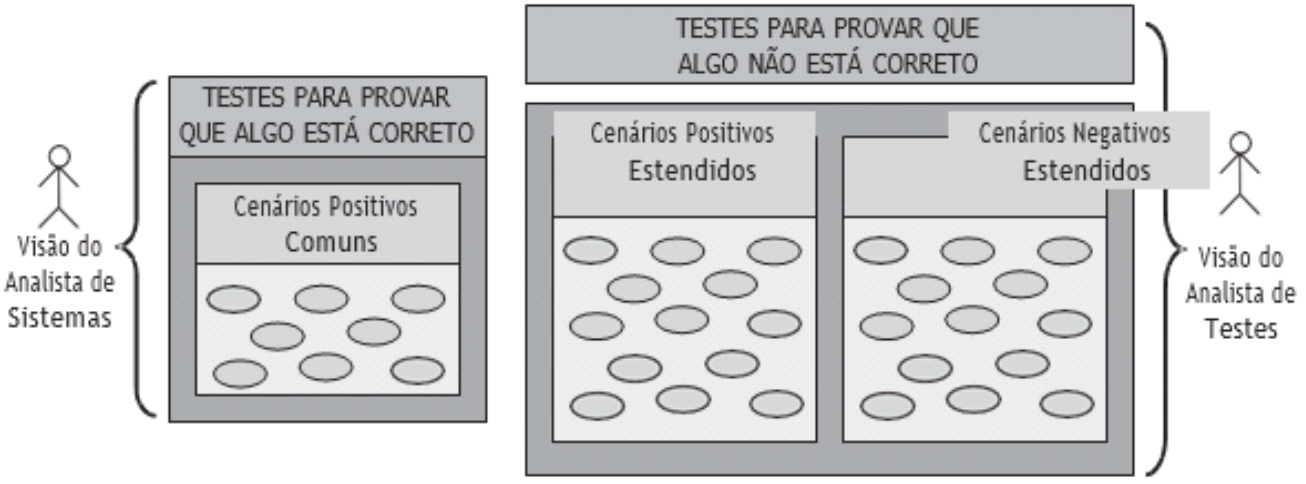
3.5 Definição comum de testes

De forma geral, todas as equipes de desenvolvimento aplicam testes em seus softwares, independentemente se os testes são bem planejados, bem estruturados e de como são executados. O fato é que esses testes não são suficientes para detectar os erros que estão inseridos dentro de uma aplicação. Um dos motivos básicos para que isso ocorra é a forma com que esses profissionais encaram os testes de software. Abaixo, algumas definições encontradas pelas diversas equipes de desenvolvimento:

Simplificação das definições dos testes

Todos enxergam os testes como uma forma de provar que tudo está bem e funcionando conforme o estabelecido. Todas as definições difundidas sobre testes dão uma dimensão positiva sobre o problema, ou seja, o entendimento sobre testes é sempre colocado sobre o prisma de avaliar se tudo está funcionando adequadamente. O fato é que é mais fácil provar que "algo funciona" do que provar que "algo não funciona", o que significa que teremos um menor esforço em provar o funcionamento de um software do que o contrário. E é exatamente isso que sentimos quando colocamos uma equipe independente de qualidade para avaliar determinado projeto de software – como essa equipe não está envolvida emocionalmente, nem politicamente com o projeto, terá um comportamento muito mais objetivo e direto, indo exatamente nos pontos que inicialmente o projeto deveria atender e, por motivos desconhecidos, foram abandonados ou não atendidos corretamente.

Visões dos testes por perspectivas diferentes



Se o objetivo é apenas provar que as coisas estão funcionando, significa que será idealizado um conjunto de cenários favoráveis à utilização de um software ou um documento qualquer, mas se o objetivo passa a ser o de provar a não adequação de algo, é preciso ir além dos cenários positivos comuns, estender a abstração e identificar um maior volume de cenários positivos. Nesse processo de refinamento dos testes, deve-se incluir um esforço adicional em identificar os cenários negativos.

A definição correta sobre testes

Entender que o objetivo dos testes é "provar que algo não funciona" é um avanço significativo na compreensão de um processo de qualidade de software. Não adianta ter documentações incompletas, imprecisas e ambíguas. É necessário buscar um maior nível de qualidade em todos os produtos (sejam documentos ou softwares) produzidos em todas as fases do ciclo de desenvolvimento.

Esses documentos auxiliarão as equipes a tomarem decisões mais corretas sobre o projeto de software que refletirá em um produto com maior conformidade com as necessidades dos clientes e usuários.

3.6

Portanto, os testes em documentos deverão não somente analisar se as definições foram registradas, mas se estas foram escritas de forma a não dar margens a dúvidas e se estão em conformidade com as demais. Se o documento registra decisões que foram analisadas, devemos avaliar se tais decisões estão apoiadas em informações e análises objetivas e não por dados infundados ou meras suposições.

Teste é um processo sistemático e planejado que tem por finalidade única a identificação de erros

Atitude zero-defeito

3.7

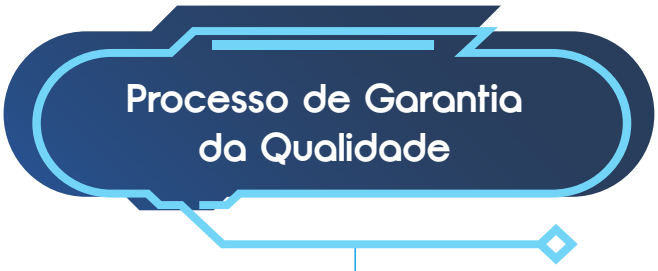
Foi Glenford J. Myers no livro The Art of Software Testing que corretamente vinculou a palavra testes com o compromisso da identificação de erros. O autor é um respeitável estudioso no assunto e possui diversos estudos e publicações sobre processos de testes de software. Uma de suas conclusões sobre o tema é entender que zero-defeito é algo inatingível, ou seja, pela complexidade envolvida e pelo número altíssimo de situações existentes, torna-se impossível imaginar um produto de software "livre de erros". Não podemos nos enganar, sempre existirão erros a serem descobertos, porém o desafio de um processo de garantia da qualidade é justamente tornar esse risco o mais próximo possível do zero.

A qualidade de software trabalha com o conceito "zero-defeito", que representa a não-tolerância a erros de dentro de um processo de qualidade de software. O objetivo é definir um processo que contenha mecanismos de inibição de defeitos, impedindo que falhas sejam criadas e propagadas para as fases seguintes. Todo o processo é desenhado para minimizar a incidência de problemas.

4

Os pilares da qualidade de Software

Aqui faremos referência aos trabalhos realizados no PMI (Project Management Institute) que organizou o chamado PM6OK (Project Management 6 of Knowledge) no qual o processo de gerenciamento da qualidade é subdividido em três subprocessos complementares:



Planejamento da Qualidade

Todas as atividades referentes ao planejamento das atividades da qualidade e os esforços na prevenção de defeitos

Garantia da Qualidade

Todas as atividades técnicas e procedimentos realizados com o objetivo de identificar erros em artefatos do software

Controle da Qualidade

Todas as atividades técnicas e procedimentos relacionados a medir e monitorar a qualidade do processo e do produto de software

4.1 Planejamento da qualidade

Processo destinado a identificar quais padrões de qualidade são relevantes para o projeto e determinar como satisfazê-los. É realizado em paralelo com outros processos de planejamento e tem como produto o Plano da Garantia da Qualidade de Software (SQA Plan – Software Quality Assurance Plan) e todos os planejamentos mais direcionados (Estratégias de Testes das diversas categorias existentes).

4.2 Garantia da qualidade

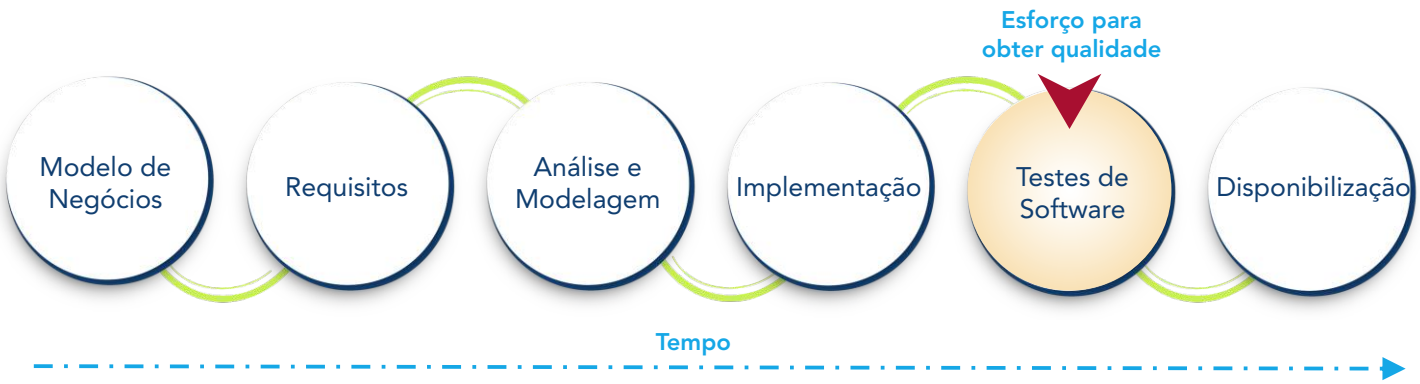
Processo que engloba a estruturação, sistematização e execução das atividades que terão como objetivo garantir o adequado desempenho de cada etapa do desenvolvimento, satisfazendo os padrões de qualidade definidos no processo. Aqui estarão relacionados os testes de verificação (testes estáticos) e os testes de validação (testes dinâmicos ou testes de softwares) previstos em todas as etapas do ciclo de desenvolvimento.

4.3 Controle da qualidade

Processo que se concentra no monitoramento e desempenho dos resultados do projeto para determinar se ele está atendendo aos padrões de qualidade no processo de desenvolvimento. Trata-se de um processo contínuo e sistemático de acompanhamento da eficiência do desenvolvimento em diversos pontos de controle, possibilitando às gerências e profissionais envolvidos acompanhar variações de qualidade e promover ações corretivas e preventivas para manter o nível de qualidade desejado. Avaliará sistematicamente a qualidade do processo em execução e a qualidade do produto tecnológico que está sendo desenvolvido.

4.4 Onde aplicar a qualidade?

Um dos erros mais comuns sobre qualidade é o modo como este conceito é inserido dentro do processo de engenharia de software. Todos tendem a pensar o desenvolvimento de software como uma linha do tempo. Dessa forma, são definidos metodologias e processos de trabalho para produzir software e garantir que as etapas serão cumpridas e que o produto terá seu ciclo completo de desenvolvimento. O erro é acreditar que dentro desse processo existe um período alocado especialmente para a realização dos testes.



Dessa forma, comete-se o equívoco de acreditar que somente obteremos qualidade após a codificação de partes do produto a ser desenvolvido. Isso fere a ideia de identificar os erros nas fases iniciais do processo de desenvolvimento de software, evitando a migração dos erros e, consequentemente, elevando os custos de correção.

Qualidade não é uma fase do ciclo de desenvolvimento de software...

... é parte de todas as fases.

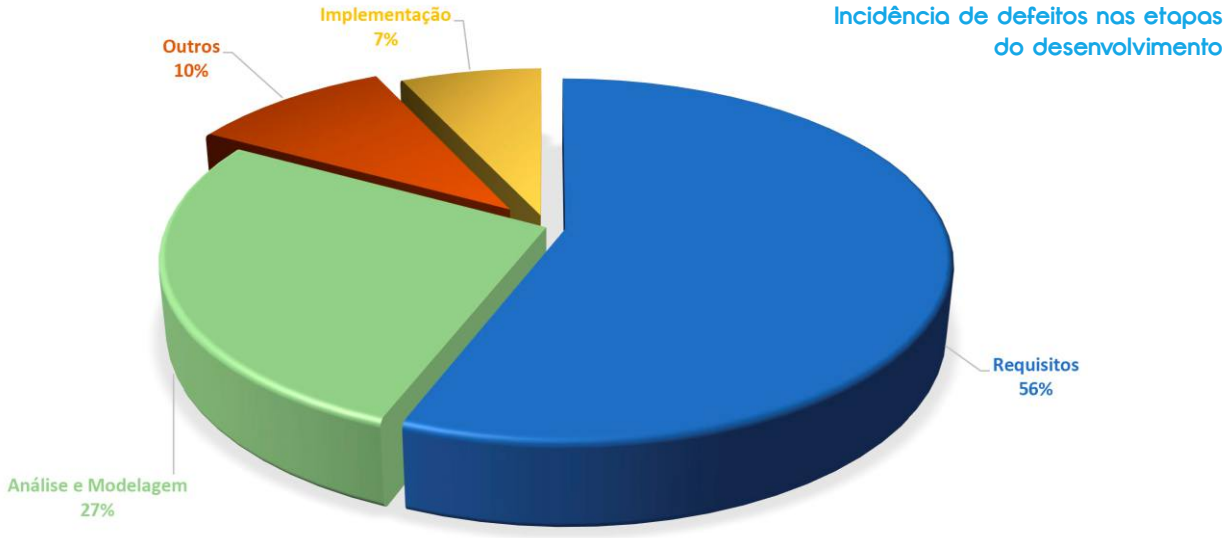
Isso significa que devemos garantir a qualidade de todas as etapas do processo de desenvolvimento do software, de forma a garantir que uma nova fase não será iniciada caso a anterior ainda não tenha sido bem finalizada e entendida. Em outras palavras, é preciso garantir a concepção do sistema desejado antes de analisar os requisitos de negócios; a análise dos requisitos antes de iniciar o processo de arquitetura do software; a modelagem do software antes de codificar o produto; o código antes de executá-lo. Dessa forma, você ampliará a concepção de garantia da qualidade dentro do processo de engenharia de software como um todo.

E os defeitos?

Os defeitos são conhecidos por seus vários nomes. Podem ser chamados de erros, problemas, falhas, ocorrências, incidentes, não conformidades e inconsistências. Também podemos empregar palavras existentes na língua inglesa como bugs, crash, abends. Apesar de esses nomes representarem naturezas de defeitos diferentes, eles demonstram um desvio de qualidade no ato da elaboração de um documento, na execução de uma atividade, na construção de um componente de software. São esses desvios de qualidade que produzem retrabalho, aumentam os custos do projeto, dilatam os prazos de entrega do software, diminuem a produtividade e aumentam a insatisfação do cliente.

Existe uma visão comum de entender que os defeitos somente são provenientes do código-fonte do produto. Também se entende que somente os profissionais de desenvolvimento, qualidade e testes são os responsáveis por um software sem defeitos. Em ambas as afirmações, estão cometendo um grave erro: primeiro, os erros são gerados durante todo o processo de engenharia do software – são os resultados de traduções imperfeitas de requisitos e especificações que provocam a maior parte dos problemas. Através de estudos realizados, é possível constatar que a participação das diversas áreas reduz drasticamente o risco de insucesso do projeto, ou seja, as áreas de desenvolvimento e qualidade devem interagir continuamente com as áreas de negócio, produção, suporte, infraestrutura e atendimento ao cliente.

Os erros ocorrem em todas as fases do processo de desenvolvimento de software, porém, estudos demonstram que a maior incidência dos erros está concentrada nas fases iniciais do processo de desenvolvimento.



Muitos dos erros identificados no produto final são provenientes da má especificação e entendimento sobre os objetivos a serem alcançados. Se o seu objetivo é reduzir, ao máximo, o nível de erros dentro do processo de desenvolvimento de software, deve se concentrar nas atividades iniciais do processo. Dessa forma, você identificará prematuramente os erros e impedirá que estes migrem para outras fases.

Qualidade em todo o ciclo de desenvolvimento

Se os erros estão concentrados nas fases iniciais do ciclo de desenvolvimento, deve-se investir mais tempo nas atividades de especificação e modelagem da solução. Isso faria com que muitos dos erros fossem eliminados em função de um levantamento mais criterioso e bem estruturado. Contudo, a redução dos erros somente ocorrerá se em todas as etapas do processo de desenvolvimento existir procedimentos que avaliem a qualidade do que foi produzido. Dessa forma, cada etapa teria seus produtos adequadamente avaliados e vistoriados, atestando a qualidade destes e impedindo que uma nova fase seja iniciada sem que tenha sido concluída adequadamente.

Ter um processo de garantia da qualidade em todo o ciclo de desenvolvimento do software permite que um número maior de defeitos seja descoberto antecipadamente, evitando a migração destes para as fases seguintes. Isso fará com que os custos referentes às não-conformidades caiam drasticamente, tornando o processo mais estável e menos caótico. Também fará com que os prazos para implantação do software sejam reduzidos em função do menor índice de retrabalhos, refletindo em um aumento na satisfação do cliente (produto mais confiável), uma equipe mais motivada (menos retrabalho e atividades mais construtivas) e uma empresa mais fortalecida (clientes querem trabalhar com empresas que honram prazos e compromissos). desenvolvimento.

Processo de Garantia da Qualidade de Software

Dessa forma, a resposta à pergunta “Onde devemos aplicar Qualidade?” torna-se simples: devemos aplicar qualidade em todo o ciclo de desenvolvimento. A cultura da qualidade cria um ambiente favorável para prevenção e detecção de erros, transformando o processo de desenvolvimento em uma atividade com etapas monitoradas e constantemente avaliadas, tornando o processo confiável.

7

O custo da qualidade de Software

O custo da qualidade pode ser entendido como todo o investimento realizado com a finalidade de um produto ou serviço atingir a qualidade desejada. Esses investimentos alocam todos os esforços relacionados aos custos das não-conformidades (defeitos e suas correções), assim como todos os custos relacionados à manutenção da conformidade dos produtos e serviços a serem produzidos (esforço de garantir a qualidade).

7.1 Custos da Conformidade

Os custos da conformidade são todos os investimentos realizados para planejar e manter toda uma infraestrutura de pessoas, processos e ferramentas cujo objetivo é prevenir e detectar erros do processo. Tudo que é realizado com a intenção de melhorar e garantir o processo de desenvolvimento deve ser considerado custo da conformidade, ou seja, o custo para se obter e garantir qualidade.

- ✔ Planejamento dos trabalhos
- ✔ Treinamentos (processos, técnicas e ferramentas)
- ✔ Controles do processo de desenvolvimento
- ✔ Testes (estáticos e dinâmicos)
- ✔ Revisões de documentos
- ✔ Auditorias de processos

7.2 Custos da não conformidade

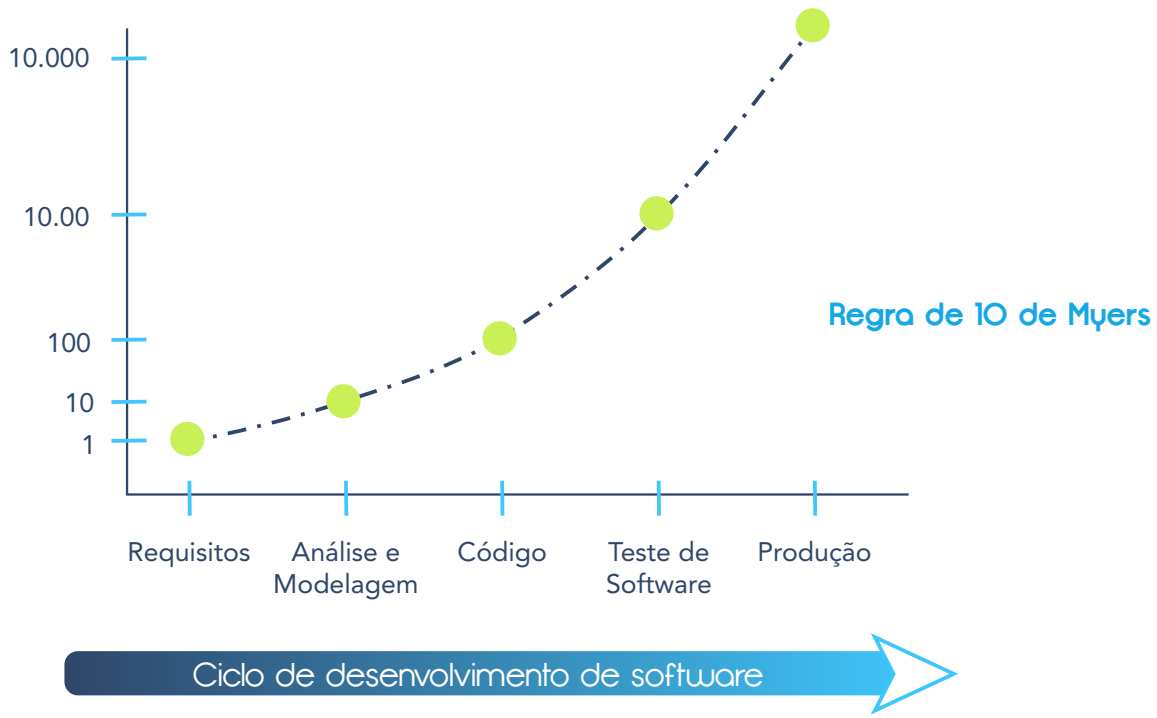
Os custos da não conformidade são todos os custos de atividades ligadas ao esforço de reparar falhas de produtos originados no decorrer do processo de desenvolvimento. Todas as consequências financeiras causadas por esses defeitos devem ser computadas nos custos da não conformidade. Tudo que é realizado ou gerado em função de defeitos produzidos durante os projetos de software deve ser encarado como não conformidade, ou seja, os custos provenientes da falta de qualidade.

- ✔ Refugos
- ✔ Retrabalhos
- ✔ Ações corretivas
- ✔ Atrasos nos cronogramas
- ✔ Perdas financeiras e operacionais
- ✔ Perdas de oportunidades

7.3 O custo da propagação dos defeitos

Todo e qualquer tipo de erro custa dinheiro. Não estamos só falando da alocação de tempo e recursos para produzir algo defeituoso, mas também dos danos provocados pelo erro, bem como sua identificação, correção, teste e implantação da correção. Porém, segundo estudos de Myers, quanto mais tardiamente descobrimos os erros, mais caros estes se tornam. Ele aplica a chamada "regra de 10" aos custos da correção dos erros. Significa que quando um erro não é identificado, os custos de sua correção multiplicam-se por 10 para cada fase em que o erro migra. Analisando o gráfico mostrado a seguir, fica evidente a vantagem de possuir um processo de detecção de problemas de forma a evitar que um erro "migre de uma fase a outra" e que os custos de correção se multipliquem. É preciso entender de uma vez por todas que testar nas fases iniciais sai mais barato do que nas fases mais tardias do processo de desenvolvimento.

Erros na produção são extremamente caros



7.4 A relação custo versus qualidade

Em um mundo sem restrições, a decisão de implementar um processo completo de qualidade em todo o ciclo de desenvolvimento de um software não seria problema. Porém, aplicar esses conceitos envolve grande montante de dinheiro, recursos profissionais e tempo, e nem sempre teremos condições para fazê-los. No mundo real, bons profissionais conseguem direcionar seus esforços e contornar as restrições que lhes são impostas.

Isso significa priorizar as atividades importantes e dar maior ênfase aos softwares e etapas mais críticas e problemáticas. Dessa forma, pode-se minimizar os riscos envolvidos na atividade de desenvolvimento de um software e implantar um efetivo processo de garantia da qualidade.

Outros aspectos estão também relacionados à maturidade organizacional do cliente, ou seja, se ele ainda não possui um processo estruturado de desenvolvimento de software, com certeza não será possível estabelecer um processo de qualidade completo. A falta de documentos básicos inviabiliza esse processo, pois o desenvolvimento é tão informal que não haverá documentações e especificações a serem inspecionadas.

Custo da qualidade versus efetividade do processo

Auditoria

Nível superficial de um processo de qualidade. Verifica se as atividades e documentos estão respeitando as estruturas e padrões estabelecidos. Trata-se de uma análise do esqueleto do processo. Não é avaliada a qualidade dos documentos gerados nem do software desenvolvido pela equipe.

Produto Acabado

Nível mais praticado nas organizações. O processo de qualidade somente é iniciado após o software ter sido transformado em um produto tecnológico. São aplicados todos os testes possíveis no produto acabado.

Parcial

Nível intermediário de qualidade. O processo de teste inicia-se juntamente com o processo de estruturação interna do software e o acompanha ao longo de seu ciclo de desenvolvimento. Os testes são orientados por requerimentos e especificações funcionais.

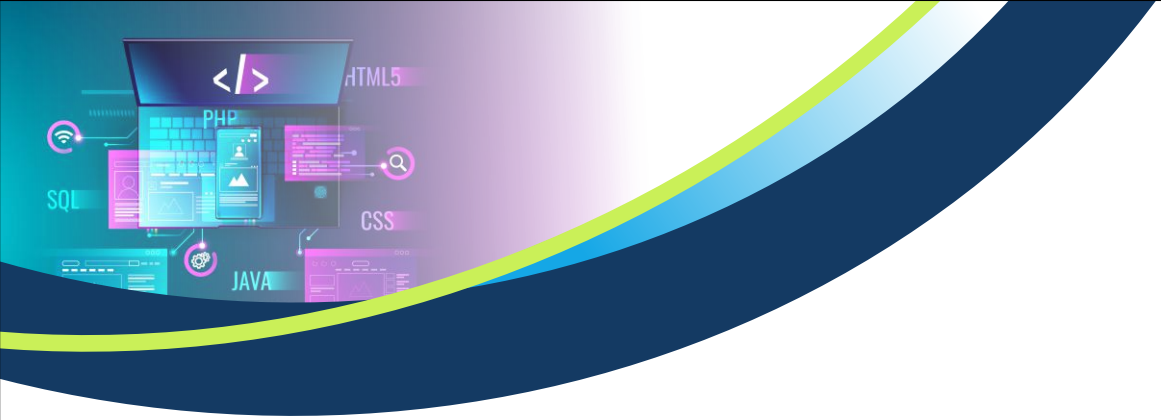
Completo

Trata-se de um efetivo processo de garantia da qualidade do software. O processo inicia-se conjuntamente com o levantamento de requisitos do software. Todas as etapas são completadas segundo um critério de finalização previamente estabelecido, os documentos são todos analisados e revisados pela equipe de qualidade de software e todas as categorias de testes são aplicadas no software desenvolvido.

8

Benefícios de um processo de qualidade de software

Qualquer tipo de erro gera custo financeiro à organização. Enquanto o software não é implantado, os erros identificados ficam restritos ao projeto como retrabalho, sendo necessário contabilizar os custos de identificação do problema, remodelagem, recodificação, teste e nova implantação. Quando esse software já está em produção, os custos de um erro ultrapassam a fronteira do projeto e passam a interferir nos resultados financeiros e operacionais das diversas áreas da organização. Nesse caso, deve-se incluir no custo do erro não somente os aspectos diretamente ligados ao projeto, mas também os prejuízos financeiro e operacional provocados pelo defeito gerado.



Ciclo de desenvolvimento mais confiável

Como os softwares tornaram-se cada dia mais importantes para as organizações, minimizar os riscos de um erro significa poupar a empresa dos custos provenientes desse fato. Mesmo sendo impossível atingir o tão esperado software bug-free, é recomendado criar uma estrutura e processos que proporcionem um ambiente favorável e que permitam a detecção do maior número possível de erros.

8.1

Nenhum processo de engenharia de software, por mais bem idealizado, planejado e executado que for, não poderá garantir um software sem erros. Seguindo esse raciocínio, é preciso entender que em todas as aplicações de software, um conjunto de defeitos está incorporado à aplicação e pronto a se manifestar quando determinado conjunto de situações for combinado, provocando, assim, uma falha no comportamento esperado do software. A missão de descobrir defeitos, quando tratada na dimensão e com a importância que realmente lhe cabe, cria consciência e atitudes direcionadas ao "Zero-defeito". Tal atitude está voltada à não proliferação dos erros, ou seja, uma percepção de que tudo que estamos fazendo pode comprometer a qualidade final do software a ser desenvolvido. Tudo que venha a interferir na qualidade final do produto deve ser repensado no contexto "zero-defeito".

Garantia de ações corretivas no ciclo de desenvolvimento

Em um projeto de desenvolvimento no qual o processo de garantia da qualidade do software não está sendo empregado, os erros são produzidos continuamente, mas ninguém está administrando sua frequência, reincidência ou gravidade. Ninguém está analisando as áreas do software que sofrem maior incidência de problemas de forma a estudar as origens "reais" dos problemas – erros são apenas sintomas, deve-se localizar as reais causas dos problemas de desenvolvimento.

8.2

A equipe de qualidade, através de um levantamento estatístico referente aos erros identificados, pode demonstrar os pontos do projeto que apresentam maior incidência de problemas, possibilitando ações corretivas durante o ciclo de desenvolvimento do software. As ações corretivas têm por objetivo minimizar problemas que são identificados durante o desenvolvimento.

Maiores chances de sucesso do projeto de software

Administrar um projeto de desenvolvimento de software para o sucesso significa eliminar ou minimizar os riscos e conflitos existentes. Existem diversos fatores que podem contribuir com a qualidade final do produto – profissionais experientes e bem treinados, metodologias e ferramentas adequadas, participação constante dos usuários finais, bom entendimento do problema e modelagem da solução flexível em longo prazo.

8.3

Isoladamente, um perfeito processo de garantia da qualidade de software não garante sucesso ao projeto de desenvolvimento de um software, uma vez que este não ataca todos os possíveis riscos inerentes de um projeto. Porém, um bom processo de qualidade minimiza diversos pontos críticos de um projeto de desenvolvimento de um software – identifica prematuramente erros em documentos e análises realizadas, garante que cada fase do desenvolvimento produziu os

documentos obrigatórios e que estes foram adequadamente revisados pelas áreas responsáveis, garante o comportamento do software nas diversas condições existentes, monitora seu comportamento sob condições extremas de acesso, mantém o software em situações de contingência e cenários de exceção.

Torna-se verdadeiro afirmar que o processo de qualidade amplia as chances de sucesso de um projeto de desenvolvimento porque agrega ao software confiabilidade, fator fundamental para o sucesso de um projeto.

8.4 Maior produtividade do desenvolvimento

A primeira impressão é a de que quanto mais pessoas direcionam seus esforços na produção de um software, mais rapidamente teremos uma solução tecnológica disponível e mais cedo estaremos nos beneficiando desse investimento. Portanto, ampliar o número de desenvolvedores significa aumentar a capacidade de produção da equipe, possibilitando encurtar prazos e obter mais rapidamente o resultado esperado. Mas isso não é verdade absoluta.

Algumas pesquisas demonstram que a desorganização se amplia à medida que colocamos mais pessoas para interagir em um ambiente caótico. O que devemos é melhorar a qualidade desse trabalho e, somente assim, acrescentar novos recursos direcionados ao desenvolvimento do projeto.

8.4.1 Fator Desorganização

Todo projeto tecnológico tem seus níveis de desorganização: uns são mais acentuados, outros menos. A desorganização reflete o número de erros gerados e o quanto este se propagou nas fases do projeto. Quanto maior o número de erros e maior a propagação destes, maior será o nível de desorganização do projeto estudado. A desorganização reflete na produtividade da equipe de desenvolvimento e, consequentemente, em retrabalhos do projeto tecnológico.

Fatores que contribuem para a desorganização de um projeto

A entrada de um novo profissional em um projeto altamente desorganizado (altos índices de erros) deve ser encarada de forma positiva caso esse profissional seja direcionado a atividades deficientes do projeto para contribuir com a redução do nível de "desorganização", refletindo imediatamente em menos erros, menor índice de retrabalho e, consequentemente, maior produtividade da equipe como um todo. Do contrário, a entrada desse profissional somente aumentará o nível de desorganização do projeto.

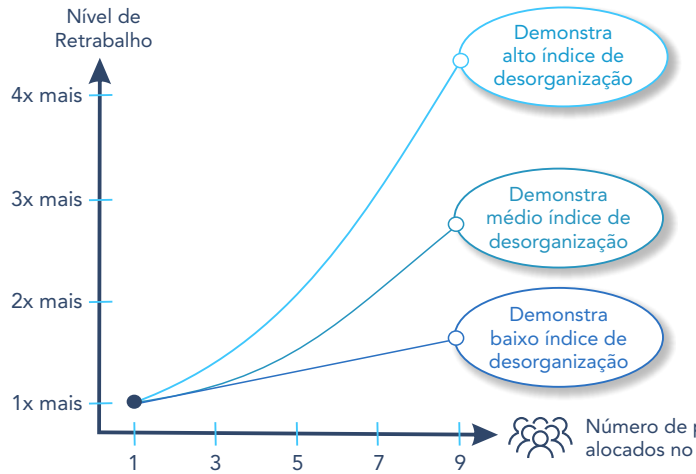
8.4.2 Fator Retrabalho

O fator retrabalho está intimamente ligado ao fator desorganização. É comum encontrarmos projetos de software que parecem nunca conseguir atingir um nível básico de funcionalidade. Os prazos são ampliados, as equipes aumentadas, mais recursos financeiros são direcionados ao projeto, porém não se consegue reverter o quadro caótico.

Trata-se de uma equipe que, apesar de aumentar o número de profissionais dedicados ao projeto, não conseguiu aumentar sua produtividade. Esse cenário não é tão incomum assim, a explicação é que uma equipe tem sua produtividade prejudicada quando o nível de retrabalho é muito alto.

O retrabalho tira os profissionais da atividade de produzir "algo novo" pela atividade de "corrigir algo defeituoso". Cada novo desenvolvedor potencializa o nível de desorganização, trazendo mais "retrabalho" ao projeto. Será mais um profissional gerando erros em seu próprio código e nos códigos de seus colegas. Será mais alguém participando do desenvolvimento, ampliando as dificuldades de comunicação da equipe e direcionamento de um objetivo comum.

Em um determinado momento, é possível perceber que o projeto tem a maior parte de seus recursos direcionados a "fazer o que já foi feito". Para o cliente final, trata-se de um projeto sem fim, sem prazo determinado para acabar.



Influência da desorganização no nível de retrabalho

8.5

Propagação de erros mitigada

Um dos maiores benefícios que os testes agregam a um projeto de software é o fato de existir um processo de qualidade de software que garanta a "regressividade" de uma aplicação, ou seja, garanta que a manutenção ocorrida na aplicação não afete o comportamento das outras funcionalidades. Mesmo um sistema em produção, que roda diariamente de forma segura, pode ter suas funcionalidades comprometidas caso necessite de pequena alteração para se adequar a uma nova condição de negócio. Sem um processo de testes regressivos, os testes se concentrarão nas partes do software que sofreram as mais recentes modificações (testes progressivos), descartando qualquer possibilidade de essas mudanças propagarem erros para outros módulos. Trata-se de uma situação muito comum do dia a dia dos desenvolvedores.

9

Automação de Testes

A automação de testes é a utilização de ferramentas de testes que simulem usuários ou atividades humanas de forma a não empregar procedimentos manuais no processo de execução dos testes. Requer profissionais especializados, exigindo mais detalhes no planejamento e tempo adicional para o desenvolvimento e automação dos testes. A automação dos testes é altamente desejada por diversos fatores, inclusive em termos de custos finais. Como esse processo requer um investimento inicial, a automação é normalmente encarada como mais um trabalho a ser realizado. Mas à medida que reexecutamos os testes, o ganho de tempo, controle, confiabilidade e as diversas possibilidades existentes com essa tecnologia, fica clara a vantagem inerente a esse processo.

Esta tabela, extraída do The Newsletter of the Quality Assurance (1995), demonstra um estudo que envolve 1.750 casos de testes e 700 defeitos existentes. Nesse estudo fica evidente que processos de testes baseados em ferramentas de automação são mais econômicos, rápidos e eficientes do que os processos manuais de testes.

Comparativo entre testes manuais e automatizados

Etapas dos Testes	Teste manuais	Teste automatizado	Melhoria %
Planejamento	32	40	-25%
Definição dos casos de testes	262	117	55%
Execução dos testes	466	23	95%
Conferência dos testes	117	58	50%
Gerenciamento do erro	117	23	80%
Relatórios finais	96	16	83%
Duração total (em horas)	1090	277	75%

Por dentro da automação de testes

A automação de testes nada mais é do que um script criado com a responsabilidade de testar a aplicação de forma literal: validar todas as funcionalidades do produto que necessitam operar como o esperado – aquele famoso caminho feliz! Mas para conseguir chegar a resultados precisos e certeiros é necessário ter uma definição de escopo e metas bem definidas.

Por se tratar de um projeto de desenvolvimento, é preciso ter atenção em cada passo da implementação da automação, sobretudo aos objetivos desejados. Ter metas e escopo bem definidos é a chave para o sucesso em todo processo.

9.1

Algumas metas que podemos alcançar com automação são:

- ✔ Cobrir casos de testes e executar esses mesmos em diferentes versões e dispositivos móveis;
- ✔ Diminuir o tempo do ciclo de regressão;
- ✔ Aumentar a frequência em que são realizados;
- ✔ Um exemplo desse último ponto é rodar esses testes em horários ociosos, onde é possível estender a produtividade.

Contudo não se pode esquecer que para chegar em uma suíte de testes automatizados E2E (end-to-end; aquele que testa o fluxo da aplicação como um todo), é preciso ter testes manuais coesos e fáceis de se interpretar.

Se você tiver esse passo bem definido, a próxima etapa se torna simples: automatizar cenários estáveis com os resultados desejados, uma vez que os scripts serão responsáveis por simular o uso real da aplicação.

Quando e o que eu devo automatizar

Antes de nos aprofundarmos nesse ponto, é muito importante desvencilhar qualidade apenas do ato dos testes. Na verdade, existem muitas técnicas que podem ser aplicadas a fim de se chegar ao que mencionamos anteriormente como cultura da qualidade.
Essencialmente, é importante levar a visão da qualidade para o início do processo de desenvolvimento. Desse modo, consegue-se disseminar aos poucos para todos os profissionais de tecnologia de uma empresa a ótica de que a qualidade é dever de todos e não apenas do QA (Analista de Qualidade).

Isso traz diversos benefícios, como (mas, claro, não limitados a esses):

- ✔ Menos envio de bugs para produção;
- ✔ Correção antecipada de bugs encontrados;
- ✔ Redução de custos de problemas;
- ✔ Produtos com mais qualidade disponibilizados para os usuários.

9.2

É claro que os testes devem entrar nesse planejamento. Porém, o que muitas empresas sedentas por automatizadores não sabem, ou ao menos não se atentam, é que testes manuais e automatizados são complementares, uma vez que a validação humana ainda não pode ser totalmente substituída.

Por isso é muito importante que o time de Qualidade, em parceria com outros times, planeje a automação junto ao desenvolvimento do produto (muitas vezes até mesmo antes do desenvolvimento de algum cenário), mantendo assim um alinhamento a fim de identificar quais funcionalidades devem ser automatizadas e quais devem ser testadas de forma manual.

A partir daí surge a dúvida: Qual cenário automatizar?

Em um primeiro momento, muitos dizem: "Vamos automatizar tudo ", o que pode se tornar um grande equívoco, afinal é necessário investir forças em automatizar o que é crítico para a aplicação, buscando respostas rápidas sobre o funcionamento esperado. Após ter os caminhos críticos cobertos e as manutenções em dia (por motivo de possíveis alterações no projeto), é que se pode pensar em expandir os cenários para caminhos não críticos a fim de aumentar a cobertura de testes no produto.

É necessário entender os caminhos mais críticos do seu produto e priorizá-los na hora de planejar sua estratégia de automação de testes

Implementar automação é desafiador, mas o maior desafio está em entender sua real necessidade e principalmente entender que a automação não é só do profissional de qualidade, mas do time em geral.

Após ter todas as pontas bem definidas, fica claro que para iniciar o desenvolvimento da automação é preciso se atentar ao contexto, entendendo quais os padrões de arquitetura, frameworks e linguagens serão usadas.

Agora que esclarecemos de fato as necessidades e o conceito de automação, é importante lembrar: garantia da qualidade não é só se sentar em frente a um editor de código e criar um script. Se casarmos os processos e os aplicarmos bem, todo o empenho durante uma jornada de qualidade se converte no principal foco de qualquer responsável por um produto: entregar um software de alta qualidade e que ofereça experiências incríveis aos seus usuários.

10

Por que usar ferramentas de captura e repetição em sua estratégia de Automação de Testes

Trabalhar com automatização de testes pode ser interessante pelo ganho de tempo e, consequentemente, em paralelo, pelos resultados assertivos em termos de qualidade de software.

Seja pelo valor do investimento que essa iniciativa muitas vezes requer, tanto em dinheiro quanto em tempo, muitos gestores ainda parecem receosos frente a qual caminho tomar quando pensam em automação de testes. É nesse cenário que muitos deles se deparam com ferramentas de captura e repetição (ou record and play).

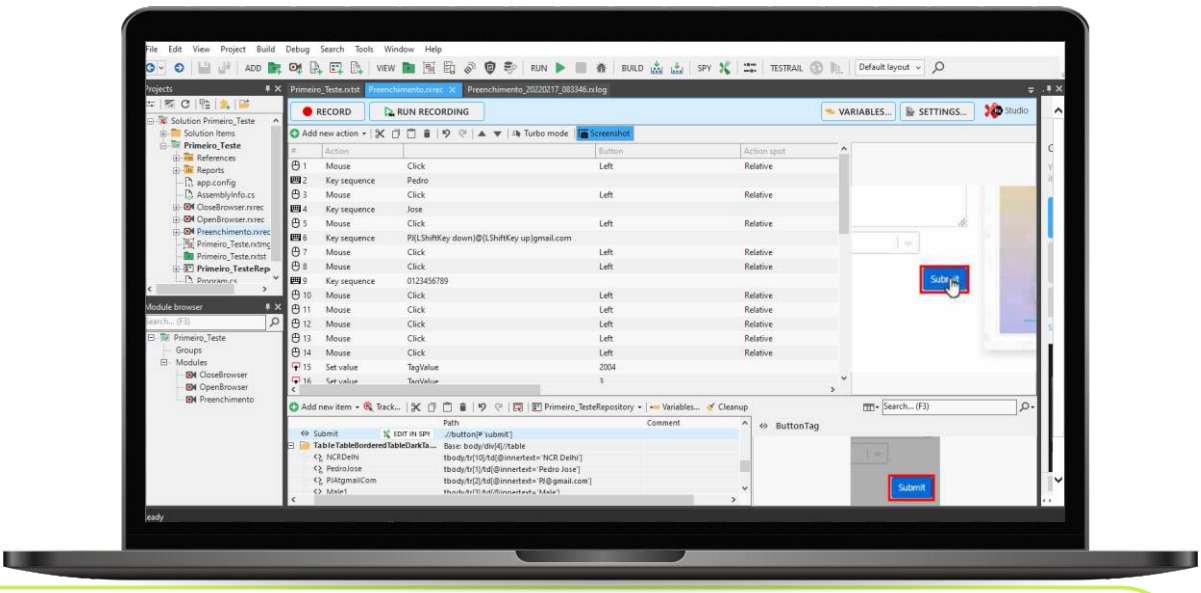
O que ferramentas de captura e repetição oferecem?

O que você acha de poder configurar cenários de testes de seu site ou app apenas percorrendo os fluxos relevantes, enquanto uma ferramenta memoriza o caminho e as ações tomadas, permitindo que, posteriormente, você consiga testar se o fluxo está funcionando como deveria?

Essas ferramentas ganham espaço na área de Qualidade de Software por conta de alguns fatores, principalmente a facilidade e a rapidez na gravação dos scripts: elas permitem gravar um script de teste de maneira muito simplificada, inclusive inserindo validações customizadas em cada etapa. Em muitos casos, não é necessário conhecimentos em programação para realizar essas gravações.

A questão da redução dos riscos de retrabalho e a facilidade na manutenção dos scripts gerados também são pontos positivos para muitos profissionais, uma vez que hoje essas ferramentas oferecem drivers e funções que não só diminuem o tempo que levaria para realizar a análise de código, quanto para realizar as alterações que podem ser feitas com alguns cliques. É possível até mesmo reaproveitar os scripts já criados pelo time e editar alguns passos para que seja possível executar as automações em cenários de testes semelhantes.

Exemplo de script gerado em uma suite de automação Record 'n' Play



Considerações Finais

Para chegar até aqui você passou pelos principais conceitos envolvidos na área de Qualidade de Software e conheceu algumas dicas práticas para a realização desses conceitos. Ao longo deste e-book, apresentamos frameworks e ilustrações para ajudar todos aqueles que buscam se aventurar no mundo de QA em suas jornadas – estudantes, pesquisadores e profissionais que procuram avançar para explorar as fronteiras do teste de software. Esperamos que todo esse conteúdo seja útil para jogar luz sobre os seus principais desafios.

A DevTrends é especialista em Qualidade de Software e reúne o que há de mais relevante e atual em tecnologia para engenharia de software. Do gerenciamento e arquitetura de banco de dados ao planejamento e automação de testes. Seu portfólio conta com marcas mundialmente conhecidas para uma abordagem Devops, como Armory, HeadSpin, Idera, Ranorex e TestRail, cobrindo as principais fases do seu desenvolvimento de software.

Entendemos que a explosão de novos sistemas e questões como usabilidade, desempenho, privacidade desafiam cada vez mais times multidisciplinares: desenvolvedores, DBAs, arquitetos de dados, testadores, engenheiros de software, segurança etc. Por isso, nossa missão é integrar as equipes, instrumentalizá-las, otimizar e adequar seus processos para um mundo em constante transformação.

Para saber mais sobre as nossas soluções, acesse o nosso site.

Fale conosco, será um prazer entender o seu cenário e contribuir com suas iniciativas de Qualidade de Software!

DevTrends

www.dtrends.com.br
contato@dtrends.com.br