

EAD
UNISANTA

QUALIDADE DE SOFTWARE

Luis Fernando Bueno Mauá

**GUIA DA
DISCIPLINA**

1. INTRODUÇÃO A QUALIDADE DE SOFTWARE

Objetivo

Nesta aula serão apresentados conceitos de controle, garantia e custo de qualidade, sendo o principal objetivo introduzir conceitos da produção de software de qualidade.

Introdução

Conceitos de qualidade são imprecisos e difíceis de serem aceitos por todas as pessoas, no entanto, métricas de qualidade de software surgem desde a década de 70 e vêm se desenvolvendo de forma a ajudar no processo de desenvolvimento de software. A garantia de controle de qualidade de software está intimamente relacionada a atividades de verificação e validação e estão presentes em todo o ciclo de vida do software. Definir qualidade de software é uma tarefa difícil. Muitas definições têm sido propostas e uma definição decisiva poderia ser debatida interminavelmente.

1.1. Conceitos de Qualidade

Como um atributo de um item, a qualidade se refere a coisas que podem ser medidas, ou seja, comparadas com padrões conhecidos, tais como, tamanho, cor, propriedades elétricas, maleabilidade, etc. Entretanto, é mais difícil categorizarmos qualidade em software, que é uma entidade intelectual, do que em objetos físicos. Ao se examinar um item baseado em suas características mensuráveis, dois tipos de qualidade podem ser encontrados: qualidade de projeto e qualidade de conformidade. Qualidade de projeto se refere a características que projetistas especificam para um item (performance, tolerância, etc.).



Entendendo

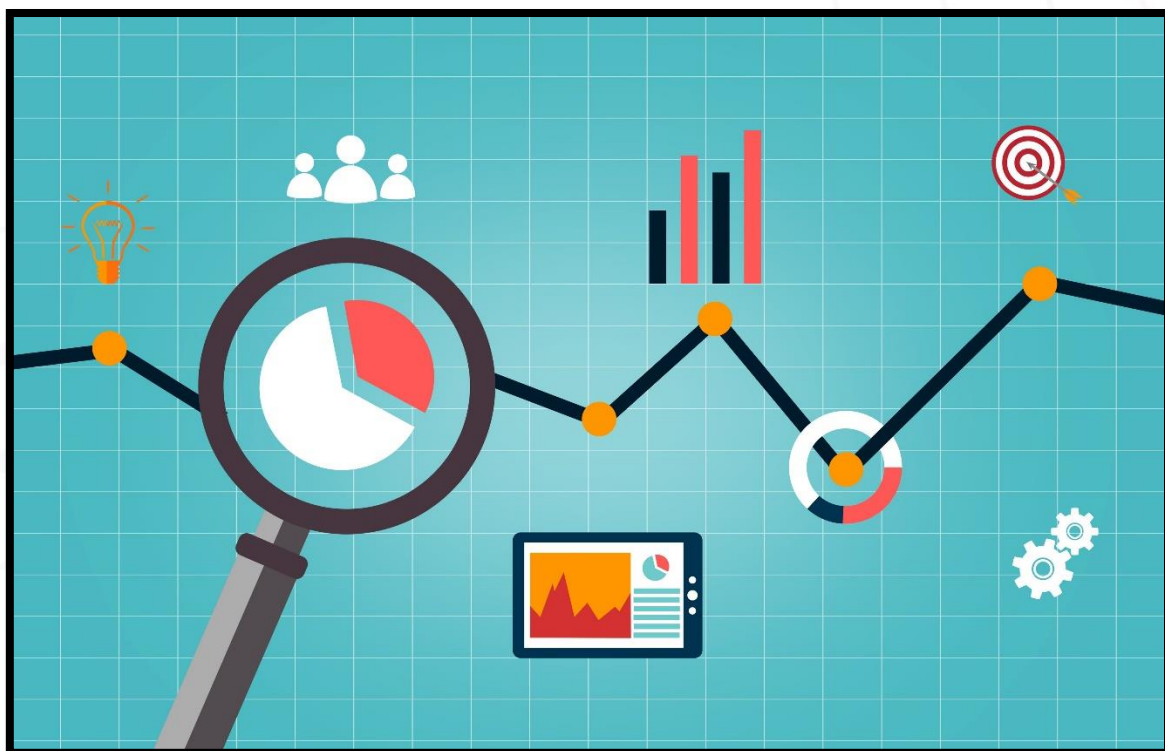
O Dicionário Aurélio define qualidade como: “propriedade, atributo ou condição das coisas ou das pessoas capaz de distingui-las das outras e de lhes determinar a natureza”

O enfoque maior é nos requerimentos, na especificação e no projeto do sistema. Qualidade de conformidade é o grau no qual as especificações do projeto são seguidas durante o processo de desenvolvimento. O enfoque maior é na implementação.

Uma definição de qualidade de software é: “conformidade a requisitos funcionais e de desempenho explicitamente declarados, a padrões de desenvolvimento claramente documentados e a características implícitas que são esperadas de todo software profissionalmente desenvolvido”

1.2. Controle de Qualidade

Pela definição da ISO, controle de qualidade é “a atividade e técnica operacional que é utilizada para satisfazer os requisitos de qualidade”. O controle de qualidade é feito através de uma série de inspeções, revisões e testes, usados através do ciclo de desenvolvimento para garantir que cada trabalho produzido está de acordo com sua especificação/requerimento. Portanto, o controle de qualidade é parte do processo de desenvolvimento e, como é um processo de feedback, ele é essencial para minimizar os defeitos produzidos.



1.3. Garantia de Qualidade

A Garantia de Qualidade de Software ou SQA (Software Quality Assurance) é uma atividade que é aplicada ao longo de todo o processo de engenharia de software.



Ela abrange:

- Métodos e ferramentas de análise, projeto, codificação e teste;
- Revisões técnicas formais que são aplicadas durante cada fase da engenharia de software;
- Uma estratégia de teste de múltiplas fases;
- Controle da documentação do software e das mudanças feitas nela;
- Um procedimento para garantir a adequação aos padrões de desenvolvimento de software, se eles forem aplicados;
- Mecanismos de medição e divulgação.



• Entendendo

A garantia de qualidade de software não é algo com o qual se começa a pensar depois que o código é gerado.

Geralmente, a garantia de qualidade consiste daqueles procedimentos, técnicas e ferramentas aplicadas por profissionais para assegurar que um produto atinge ou excede

padrões pré-especificados durante o ciclo de desenvolvimento do produto; se tais padrões não são aplicados, a garantia de qualidade assegura que um produto atinge ou excede um nível de excelência (industrial ou comercial) mínimo aceitável.

1.4. Custo de Qualidade

Como seria utópico alcançar a perfeição em um sistema de computação, então o mais importante passa a ser definir qual nível de checagem (de qualidade) seria suficiente para o sistema em questão e os custos associados. O custo de qualidade inclui todos os gastos financeiros relacionados às atividades de qualidade, os quais podem ser divididos em: custos de prevenção, custos de avaliação e custos de falhas.



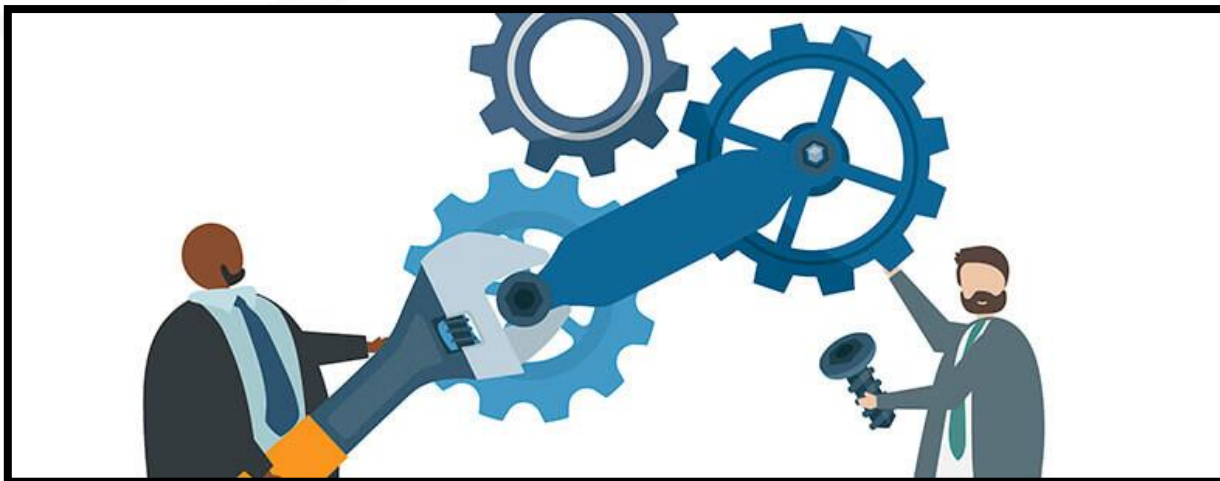
Os custos de prevenção incluem:

- Planejamento da qualidade;
- Revisões técnicas formais;
- Teste de equipamentos;
- Treinamento.

Os custos de avaliação incluem:

- Inspeções dos processos e relações entre eles;
- Manutenção dos equipamentos;
- Testes.

Os custos de falhas poderiam desaparecer se nenhum defeito ocorresse antes da entrega do produto para o cliente. Os custos de falhas podem ser divididos em: custos de falhas internas e custos de falhas externas.



Os custos de falhas internas incluem:

- Retrabalho;
- Conserto de bugs;
- Análise de falhas.

Os custos de falhas externas incluem:

- Resolução de queixas;
- Troca/devolução do produto;
- Suporte em help on-line;
- Trabalhos de segurança.



• Importante

O custo da qualidade é fator preponderante para a competitividade das organizações. A eficácia dos programas de qualidade deve contemplar bons sistemas de custos para viabilizar os processos produtivos.

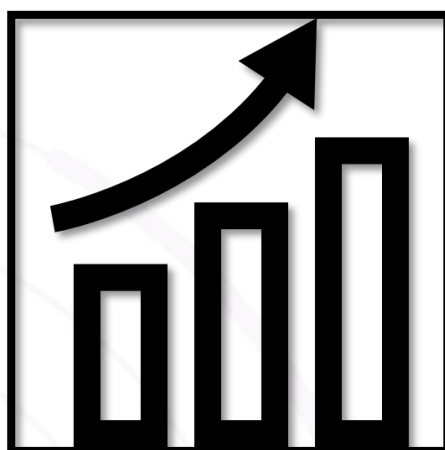
2. EVOLUÇÃO DOS CONCEITOS DE QUALIDADE

Objetivo

Esta aula irá abordar os conceitos da evolução da qualidade ao longo da história com destaque no Ocidente. Compreende a evolução pela busca da qualidade desde a revolução industrial com enfoque no produto, processo e sistema.

Introdução

Ao longo da história o homem sempre procurou o que mais se adequasse às suas necessidades, fossem estas de ordem material, intelectual, social ou espiritual. A relação cliente-fornecedor sempre se manifestou dentro das famílias, entre amigos, nas organizações de trabalho, nas escolas e na sociedade em geral. No final do 2º milênio vivemos o cenário da busca da Qualidade Total nas empresas como fator de sobrevivência e competitividade. Hoje, nos primeiros passos do 3º Milênio, para melhor compreender a evolução do conceito Qualidade é importante analisarmos "de onde viemos", a fim de entendermos "onde estamos", para então sabermos, "para onde estamos indo" na trilha de evolução da Qualidade no mundo. Neste sentido, a arte de se obter Qualidade experimentou uma grande evolução no século XX, partindo da mera inspeção de produtos acabados à visão estratégica de negócios. Esta evolução pode ser analisada conforme seu contexto no Ocidente, no Japão e no mundo como um todo.

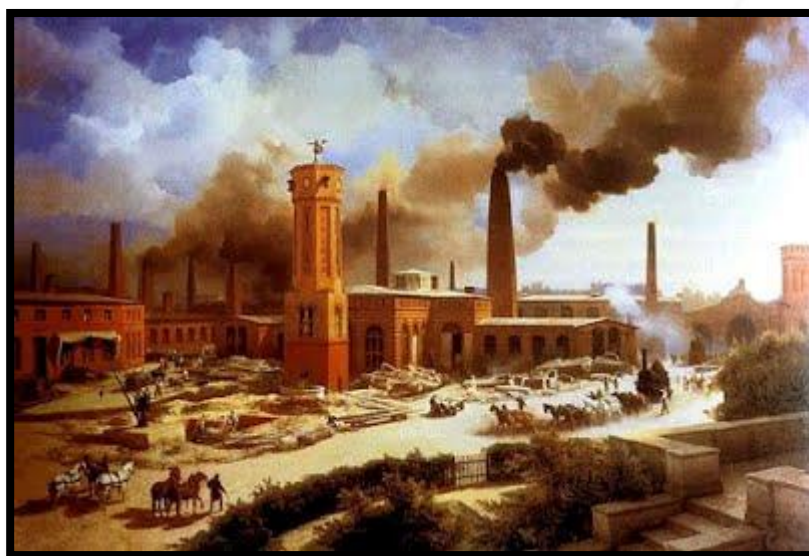


2.1. A Evolução da Qualidade no Ocidente

Voltando no tempo, especialmente a partir da Revolução Industrial, com o desenvolvimento das ferramentas de trabalho e dos sistemas de unidades de medidas,

tanto na Inglaterra quanto nos Estados Unidos da América, a Qualidade evoluiu até nossos dias essencialmente através de quatro Eras, dentro das quais a arte de obter Qualidade assumiu formas distintas:

- Era da Inspeção – Qualidade com foco no produto
- Era do Controle Estatístico da Qualidade – Qualidade com foco no processo
- Era da Garantia da Qualidade – Qualidade com foco no sistema
- Era da Gestão da Qualidade Total ("Total Quality Management - TQM") – Qualidade com foco no negócio



2.2. A Era da Inspeção - Qualidade com Foco no Produto

No final do século XVIII e início do século XIX a Qualidade era obtida de uma forma bem diferente da obtida hoje em dia. A atividade produtiva era artesanal e em pequena escala. Os artesãos e artífices eram os responsáveis pela construção de qualquer produto e por sua Qualidade final. Com o desenvolvimento da industrialização, e o advento da produção em massa, tornou-se necessário um sistema baseado em inspeções, onde um ou mais atributos de um produto eram examinados, medidos ou testados, a fim de assegurar a sua Qualidade. No início do século XX, o engenheiro e executivo Frederick W. Taylor estabeleceu os Princípios da Administração Científica, e G.S. Radford, com a publicação do seu livro *The Control of Quality in Manufacturing*, legitimaram a função do inspetor conferindo a ele a responsabilidade pela Qualidade dos produtos.

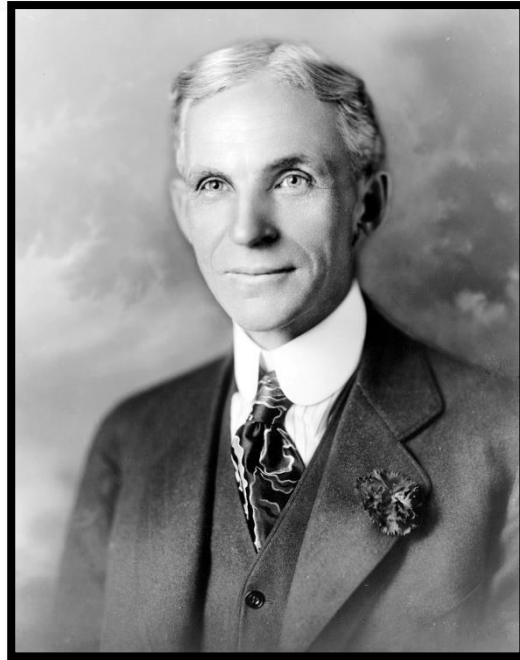


Entendendo

Frederick Winslow Taylor foi um engenheiro norte-americano que introduziu o conceito da chamada Administração científica, revolucionando todo o sistema produtivo no começo do século XX e criando a base sobre a qual se desenvolveu a atual Teoria Geral da Administração.

A força-motriz do “Século da Produtividade” foram os conceitos adotados por Taylor que atribuiu ao inspetor a responsabilidade pela qualidade do trabalho. A consequência imediata deste estudo sobre os métodos gerenciais foi a separação do planejamento da produção, baseada na concepção de que os operários e os supervisores não estavam preparados para colaborar com o planejamento. Taylor atribuiu a responsabilidade do planejamento a gerentes e engenheiros, deixando aos supervisores e aos operários a execução das tarefas. Até meados do século XX era raro uma empresa apresentar em seu organograma um departamento dirigido à qualidade. Essa função era realizada por inspetores específicos, mas eles estavam espalhados pelos diversos departamentos de produção. Apenas algumas grandes organizações exibiam departamentos de inspeção final e testes, que se reportavam, normalmente, ao superintendente da produção ou ao gerente da fábrica. O sistema criado por Taylor obteve resultados surpreendentes no que diz respeito ao aumento de produtividade e passou a ser o referencial de produção para muitas

empresas norte-americanas, espalhando-se pelo mundo. Apesar da significativa contribuição de Henry Ford, o pioneiro da produção em massa, essa talvez tenha sido, a principal razão da liderança mundial dos EUA, em termos de produtividade.



Entendendo

Henry Ford foi um empresário e inventor americano que revolucionou a indústria automobilística. Foi fundador da Ford Motor Company, criador do modelo Ford T e do sistema de produção em série, conhecido como Fordismo.

O trabalho de Taylor é fruto de todo pensamento cartesiano que dominou o mundo ocidental. A princípio Taylor aplicou seu método aos departamentos de produção das fábricas. Posteriormente seus seguidores estenderam o conceito para serviços. Aos poucos o sistema taylorista gerou alguns efeitos indesejáveis devido à ênfase dada pela alta gerência à produtividade. Supervisores e operários priorizaram a produtividade e relegaram a qualidade a segundo plano. Nesta época, a inspeção estava centrada no produto, o primeiro nível de complexidade.

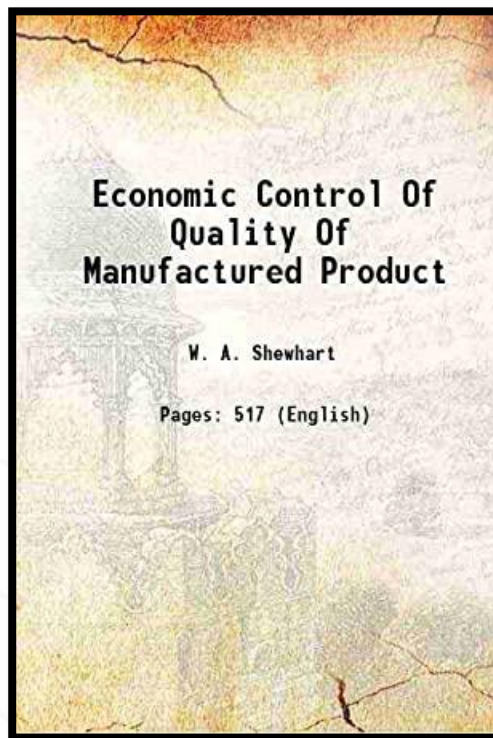


Durante a primeira grande guerra, com o aumento da atividade industrial, surge a figura do inspetor em tempo integral. Em 1922, a atividade de inspeção é formalmente incorporada ao Controle da Qualidade sendo, pela primeira vez, a qualidade vista como responsabilidade gerencial e independente. Mas, ainda, a função do Controle da Qualidade permanece limitada a inspeção. A atividade dos inspetores se restringia a identificação e quantificação dos produtos defeituosos, que, muitas vezes, resultava em medidas punitivas. Os fabricantes removiam as peças defeituosas sem que fosse feito um estudo prévio sobre as causas. A criação dos departamentos da qualidade gerou dois problemas básicos:

- A Alta Direção das empresas concluiu que qualidade era responsabilidade exclusiva do departamento da qualidade.
- A Alta Direção se distanciou da função qualidade, delegando-a aos gerentes, ficando cada vez menos informada sobre o assunto. Quando havia uma situação de crise, não tinham os conhecimentos necessários, com presteza e confiabilidade, para adotar as ações adequadas. É a típica postura dos “apagadores de incêndio”. Desta forma, nunca são removidas as causas reais das não-conformidades. A solução é, apenas, momentânea, por ser baixo o nível de aprendizado adquirido nesta prática. Esta abordagem prevaleceu por muitos anos, durante os quais a Qualidade era obtida através de inspeção, controle e separação dos “bons” e dos “maus” produtos. Aos inspetores cabia a tarefa de identificação e quantificação das peças defeituosas; estas eram removidas e trocadas sem que se fizesse uma avaliação das causas reais do problema para prevenir sua repetição. O objetivo principal era obter Qualidade igual e uniforme em todos os produtos. Ênfase à conformidade.

2.3. Era do Controle Estatístico da Qualidade – Qualidade com Foco no Processo

Houve significativos desenvolvimentos na década de 1930, entre eles o trabalho de pesquisadores para resolver problemas referentes à Qualidade dos produtos da Bell Telephone, nos E.U.A... Este grupo, composto por nomes como W. A. Shewart, Harold Dodge, Harry Romig, G. D. Edwards e posteriormente Joseph Juran, dedicou boa parte de seus esforços em pesquisas que levaram ao surgimento do Controle Estatístico de Processos. Com o crescimento da produção, o modelo baseado na inspeção 100% torna-se caro e ineficaz. Em 16 de maio de 1924, Shewhart, aplicando conhecimentos estatísticos, desenvolve poderosa técnica com a finalidade de solucionar problemas de controle da qualidade da Bell Telephone Laboratories: o Gráfico de Controle de Processo, até hoje utilizado na indústria. Ao publicar um livro, em 1931, sob o título "Economic Control of Quality of Manufactured Product", forneceu um método preciso e mensurável para definição do controle do processo, estabelecendo princípios para monitorar e avaliar a produção.



Shewart, o mestre de W. E. Deming, foi o primeiro a reconhecer a variabilidade, segundo nível de complexidade, como inerente aos processos industriais e a utilizar técnicas estatísticas para o controle de processos.



Entendendo

A contribuição mais importante de Shewhart, tanto para a Estatística quanto para a indústria, foi o desenvolvimento do Controle Estatístico de Qualidade. A ideia era incorporar o uso de variáveis aleatórias independentes e identicamente distribuídas. O princípio geral por trás da ideia é que quando um processo está em estado de controle e seguindo uma distribuição particular com certos parâmetros

Começa a se estruturar o estilo de gestão corretiva: identificar as causas reais e agir sobre elas. Matéria prima, operador e equipamento são algumas das fontes de variabilidade (causas) que podem apresentar variações no seu desempenho e característica e, portanto, afetar o produto (efeito). O conhecimento destas variações permite que a partir da sua quantificação e do estabelecimento de limites estatísticos seja possível manter o processo sob estado de controle. Através dos gráficos de controle de processo é possível identificar, minimizar e, algumas vezes, remover as causas especiais de variação. O advento da Segunda Grande Guerra Mundial exigiu que outras técnicas também fossem desenvolvidas para combater a ineficiência e impraticabilidade apresentada pela inspeção 100% na produção em massa de armamentos e munições. Neste sentido, Dodge e H. Romig desenvolveram técnicas de amostragem, nos E.U.A., que tiveram grande aceitação. Programas de capacitação começaram a ser oferecidos em larga escala nos E.U.A. e

Europa Ocidental, tanto referente ao controle de processo quanto às técnicas de amostragem. O objetivo principal era controlar a Qualidade através de métodos estatísticos. Ênfase ao controle da variabilidade. Nesta época várias associações em prol da Qualidade começaram a ser formadas e em julho de 1944 era lançado o primeiro jornal especializado na área da Qualidade, Industrial Quality Control, que deu origem mais tarde à revista hoje mundialmente conhecida, Quality Progress editada pela American Society for Quality Control (ASQC). A ASQC foi fundada em 1946 a partir da formação, em outubro de 1945, da Society of Quality Engineers, tornando-se a "locomotiva" da disseminação dos conceitos e técnicas da Qualidade no Ocidente de então até os dias de hoje.



2.4. Era da Garantia da Qualidade - Qualidade com Foco no Sistema

Durante a Segunda Grande Guerra os produtos destinados a uso militar tiveram prioridade no que dizia respeito a instalações, material, mão-de-obra habilitada e serviços de toda ordem. A produção de bens de consumo foi diminuída, incluídos os automóveis e eletrodomésticos. Enquanto isso, os operários que trabalhavam na produção militar, em muitas horas extras, fizeram aumentar o poder aquisitivo de várias famílias. No fim da guerra, em 1945, os bens para a população civil eram escassos. A prioridade máxima das empresas passou a ser, então, o cumprimento dos prazos de entrega para garantir uma fatia maior do mercado, e a qualidade dos produtos foi se deteriorando de forma escandalosa - um fenômeno que sempre se repete em tempos de escassez. A falta de produtos atraiu para o mercado novos competidores, cuja inexperiência contribuiu ainda mais para o declínio da qualidade. Nos anos que se sucederam após a segunda grande guerra, ocorre grande desenvolvimento tecnológico e industrial. Foram lançados no mercado, novos materiais e novas fontes de energia principalmente a fornecida pelas centrais nucleares, com seus requisitos tecnológicos bastante exigentes. Todos estes fatores tecnológicos, associados ao aumento das pressões provocadas pela concorrência,

provocaram profundas revisões dos conceitos adotados e grande reviravolta administrativa e econômica nos meios empresariais, bem como em toda a sociedade. Entre 1950 e 1960 vários trabalhos foram publicados ampliando o campo de abrangência da Qualidade. A prevenção passa a ser adotada na gestão dos processos produtivos tendo implicações positivas no nível qualidade resultante, mensurado pela redução de desperdícios. Os quatro principais movimentos que compõem esta era são:

- A. A quantificação dos custos da Qualidade;
- B. O controle total da Qualidade;
- C. As técnicas de confiabilidade;
- D. O programa Zero Defeitos



3. TESTE DE SOFTWARE

Objetivo

A fase de teste de software é a última etapa do desenvolvimento de um software. A fase de testes é de fundamental importância, pois através dela é possível detectar e solucionar erros no software.

Introdução

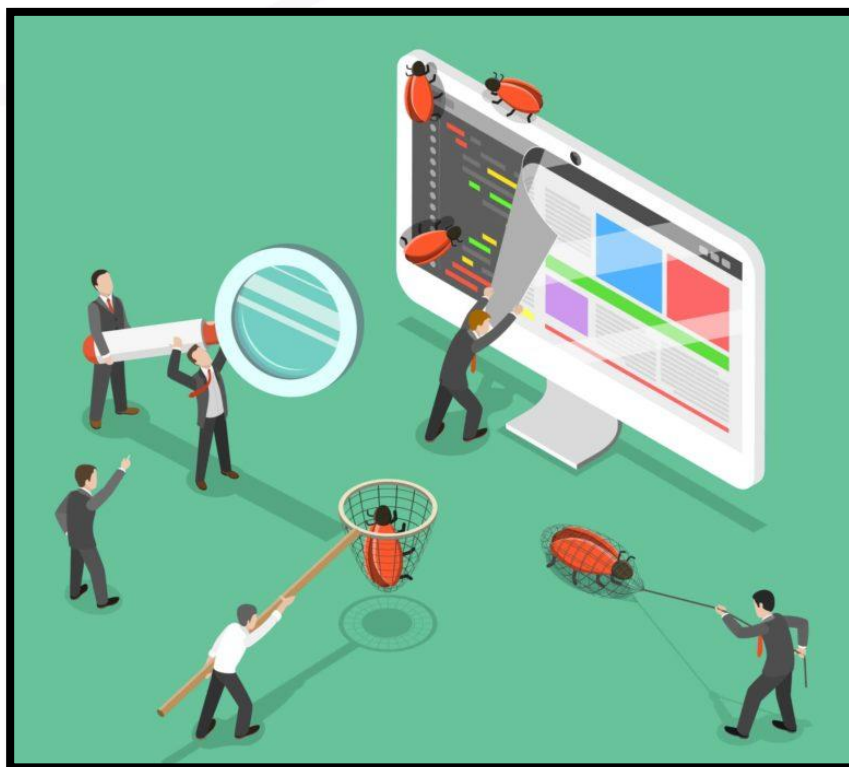
No decorrer de todo o desenvolvimento, atividades de garantia de qualidade do software são executadas, porém, a possibilidade de continuar encontrando erros é enorme, mesmo depois de concluído o desenvolvimento do software. Na fase de testes, executam uma série de atividades que consistem em varrer o software criado a procura de qualquer tipo de erro ou falha na codificação, que possam vir a interferir no correto funcionamento do aplicativo.



3.1. Falhas, erros e Defeitos

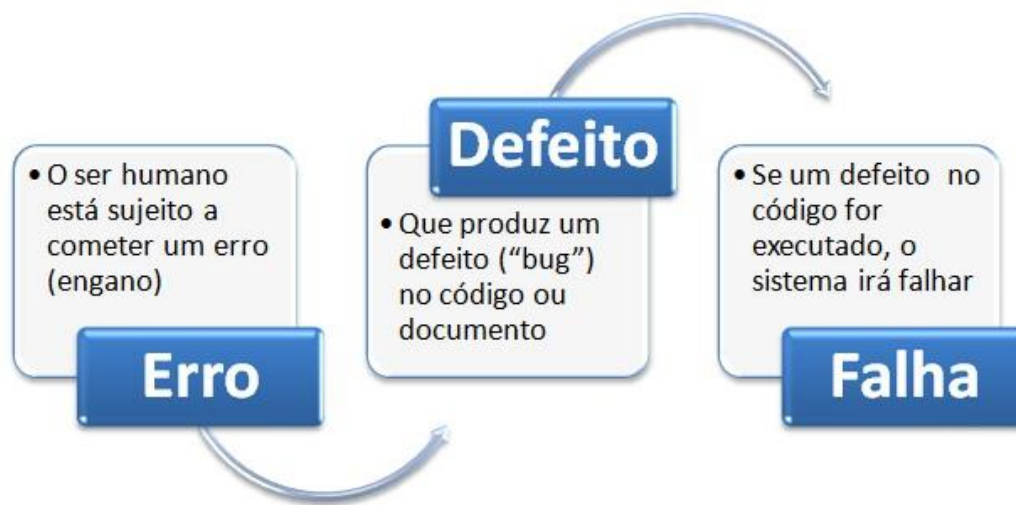
O teste é responsável por detectar o maior número possível de erros, pois encontrar todos é praticamente impossível. Os testes de software são responsáveis por executar o software utilizando valores reais, com a finalidade de encontrar erros. Caso os testes sejam realizados e nenhuma falha seja descoberta, provavelmente, os casos de testes não foram

bem elaborados. Os casos de testes devem elevar a probabilidade de detecção de erros, caso contrário, os testes não serão bem-sucedidos.



Os testes são realizados com a inserção de dados atuais para que se possam obter os resultados esperados de acordo com os requisitos e especificações previamente estabelecidos. Se os dados de saída são os esperados, significa que as informações geradas pelo software são legítimas e totalmente confiáveis. Apesar de todos os esforços exigidos na fase de teste, é comum que o software não funcione completamente sem a ausência de erros. Falhas fazem parte do código de programação, isso muitas vezes acontece devido à complexidade do desenvolvimento de software e da negligência dos desenvolvedores nas fases iniciais da elaboração de um sistema. Mesmo que os testes, por mais eficazes que sejam, não consigam detectar e remover todos os erros contidos no software, estes são indispensáveis pois aumentam o grau de segurança, e consequentemente, a confiança no produto pelo desenvolvedor e pelo cliente. Os testes são importantes, também, para evitar surpresas desagradáveis no futuro. Além disso, os testes são responsáveis por mostrar que o software possui erros e que os mesmos devem ser corrigidos, mas isso não significa que a partir das correções, o software estará imune a possíveis falhas posteriores. Novos erros podem ser inseridos ao software no momento em que outros erros são corrigidos, através de alterações realizadas no código a fim de corrigir

os erros descobertos até o momento. Os testes bem-sucedidos conseguem, em média, corrigir aproximadamente 60% das falhas contidas no software.



3.2. Custos dos Testes

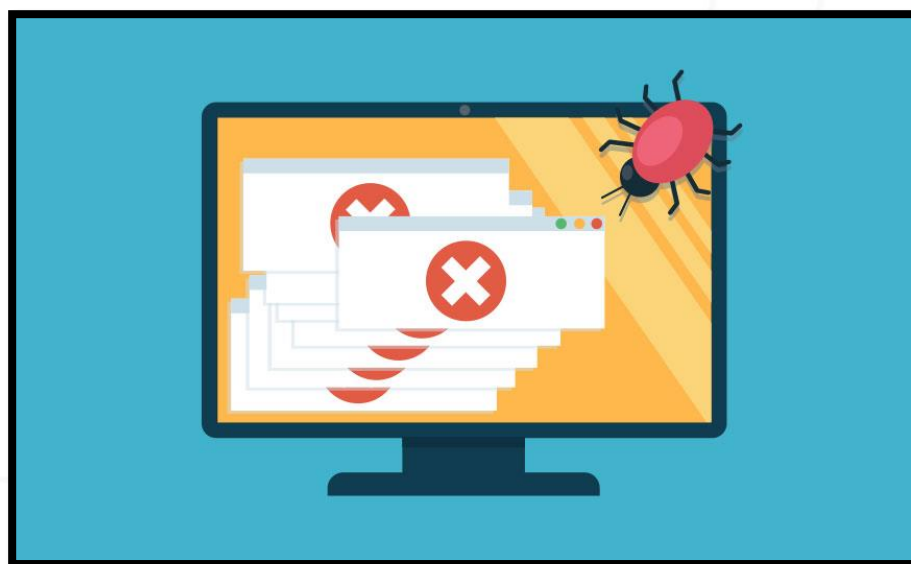
O custo dos testes realizados dentro do processo de desenvolvimento é relativamente pequeno se comparado ao gasto com manutenção corretiva do software pronto, sem mencionar que para, um software ter qualidade, é necessário que haja uma rotina exaustiva de testes, que se estende por todo o desenvolvimento do projeto. Jamais a atividade de teste deve trabalhar de maneira isolada, mas sim de maneira conjunta com as demais atividades e de forma iterativa. Os casos de teste devem testar as partes produzidas em cada etapa do projeto e o produto completo ao final do projeto. Só assim a qualidade do produto final é garantida. A atividade de testes é constituída das atividades chaves, planos de teste, projetos de teste, casos de teste, procedimentos de teste, execução de teste e relatório de teste. Podendo ser dividida também como técnicas de teste de software e estratégias de teste de software.

3.2.1. Custo efetivo dos bugs no desenvolvimento de software

Quanto custa a correção de bugs no desenvolvimento de sistemas? Depende de quão tarde você irá descobri-los!

A prevenção é melhor do que a cura não vale só na medicina! A frase do erudito holandês DESIDERIUSERASMUS é bem oportuna também na indústria de

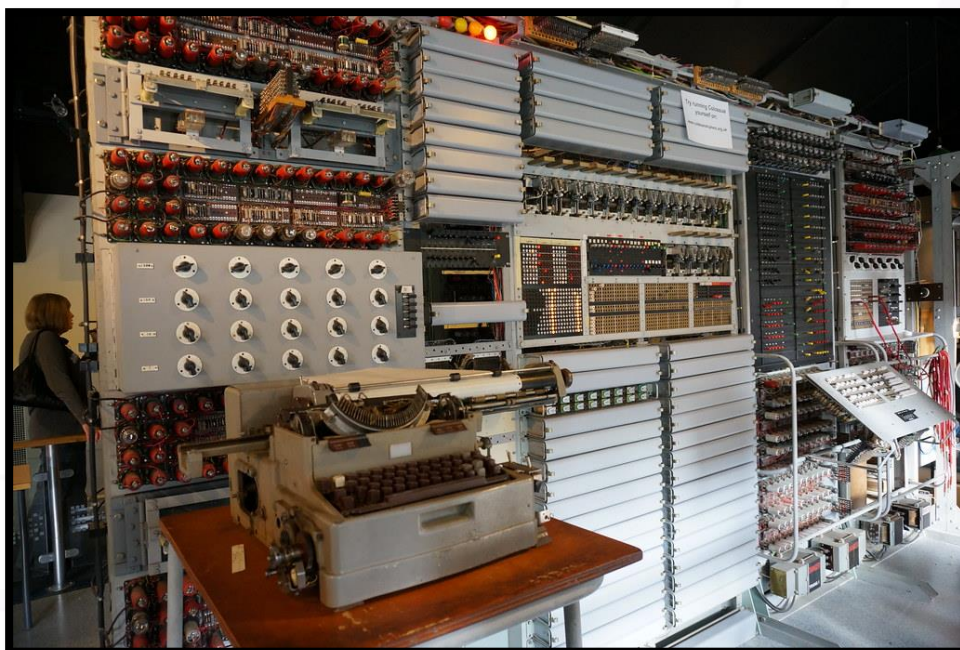
desenvolvimento de sistemas. Corrigir um bug é muito maior do que criar mecanismos para evitá-los. Entenda como bug tudo que é um defeito ou problema, desde a concepção até o pleno uso de um sistema. Programadores por aí também o chamam de probleminha, usuário, negocinho...É provável que esteja se lembrando das frases clássicas de um desenvolvedor quando você encontra um problema e ouve: “no meu computador funciona”, “você está fazendo alguma coisa errada” ou “aperta F5”...O problema é que a evidência tem mostrado que defeitos em softwares é um fator inerente na indústria de tecnologia. Um artigo da Coralogix relata que 75% do tempo dos desenvolvedores estão voltados em debugar código para descobrir e consertar problemas. A estimativa é de que a cada 1000 linhas de código criadas, 70 novos bugs são criados e 15 serão descobertos pelo seu cliente. E se isso de fato acontecer, cada bug corrigido em sistemas que estão em produção custará 30 vezes mais (tempo e dinheiro) para serem corrigidos do que se tivessem sido descobertos na fase de desenvolvimento.



3.2.2. *Origem da palavra Bugs no desenvolvimento de software*

Bug é um jargão da informática que se refere às temidas falhas inesperadas que ocorrem ao executar algum software ou usar um hardware. Esses erros imprevisíveis que prejudicam o funcionamento correto de alguma tecnologia podem desencadear problemas incômodos, como travamentos e roubar informações sigilosas. As falhas inesperadas são uma porta de entrada para os crimes cibernéticos.

Contudo, há bugs mais graves que necessitam de reparação rápida, pois podem colocar em risco o sistema de segurança de um aparelho e, conseqüentemente, comprometer dados como senhas e contas bancárias. Por isso, muitos hackers se aproveitam dos bugs para espalhar vírus, malwares e roubar informações sigilosas. As falhas inesperadas são uma porta de entrada para os crimes cibernéticos. A palavra tem origem inglesa e, em português, o significado de bug é “inseto”. Uma das teorias afirma que o termo teria sido utilizado pela primeira vez por Grace Hopper, programadora da marinha dos EUA, em 1947. Em um diário de bordo, Hopper usou a palavra para explicar o mau funcionamento do computador Mark II, da Universidade de Harvard. A falha teria sido causada por um inseto que ficou preso nos contatos de um relê, daí a associação com o bichinho e com as falhas em computadores. Apesar do registro de Hopper, há também a teoria de que Thomas Edison tenha escolhido o termo primeiro para descrever uma falha mecânica, causada por um inseto que gerou problemas na leitura do seu fonógrafo.



Os Bugs mais famosos

Bug do milênio

No final do século XX ocorreu o **bug do milênio**, que passou de um problema simples de informática para uma preocupação em todo o mundo. Isso porque os sistemas antigos que foram desenvolvidos nesse período guardavam e interpretavam as datas com

dois dígitos no ano. Sendo assim, quando houve a virada do milênio, surgiu a preocupação de o ano 2000 ser reconhecido como 1900. Isso faria com que as instituições financeiras tivessem suas aplicações financeiras com juros negativos, dando muito prejuízo para os investidores.

Após esse evento, houve grandes esforços para renovar os sistemas e corrigir o problema a tempo, antes que algum prejuízo ocorresse. Esse foi um dos fatores que resultou no crescimento de empresas do segmento de informática.



O bug de mais de 300 milhões de dólares

O lançamento do foguete Ariane 5 em 4 de junho de 1996, na França, marcava a data de um bug milionário. Naquele ano havia sido investido um alto capital para construção do foguete e de todo projeto consolidado da corrida espacial europeia. No entanto, no momento do lançamento, o foguete Ariane 5 desviou a rota e, logo em seguida, ligou o modo de autodestruição, conquistando um dos maiores prejuízos da história, levando mais de 300 milhões de dólares às chamas.

A causa do problema foi o reaproveitamento do software do Ariane 4. Acontece que o código estava recebendo o valor de 64 bits em float point. Porém, esperava-se receber um valor inteiro de apenas 16 bits. Assim, a aplicação do Ariane 5 foi incapaz de lidar com uma carga tão grande de processamento, consolidando o desastre do lançamento.

A lição dita neste acontecimento é: nunca esqueça de debugar o código.



4. NÍVEIS DE TESTES

Objetivo

Entender os níveis de Teste na sequência de um desenvolvimento de um sistema.

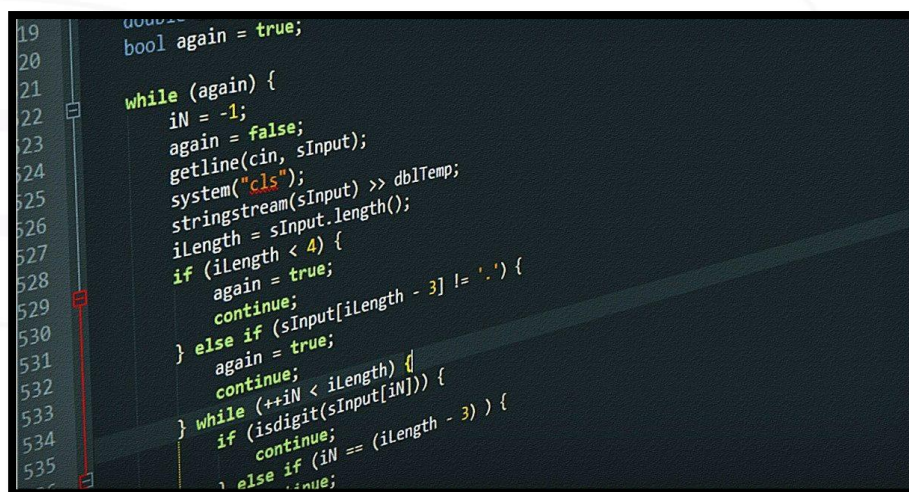
Introdução

Na última década assistiu-se a uma evolução muito significativa da indústria de testes de software, que deu lugar a múltiplas novas oportunidades, aumentando a relevância desta área para as organizações. É cada vez mais importante garantir o desempenho eficaz das aplicações e dos testes de software, certificando-nos que essas aplicações ou programas são executados com o menor número possível de falhas.

Estas garantias assentam na definição de uma série de atividades que conduzem à execução dos Testes: o Ciclo de Vida de Testes de Software (STLC – Software Testing Life Cycle). Este procedimento identifica quais as diversas fases dos testes de software e quando devem ser realizadas e concluídas.

4.1. Teste de Unidade

É responsável por testar os menores trechos de código de um sistema que possui um comportamento definido e nomeado de entrada e saída, podendo ser funções e procedimentos para linguagens procedurais e métodos em linguagens orientadas a objetos.



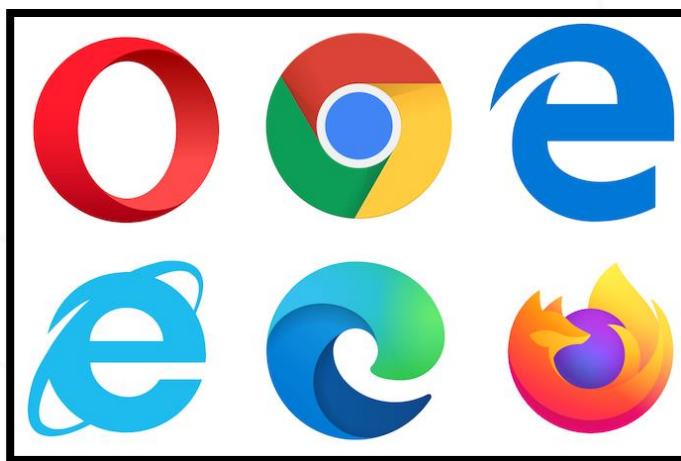
```
19  double
20  bool again = true;
21
22  while (again) {
23      iN = -1;
24      again = false;
25      getline(cin, sInput);
26      system("cls");
27      stringstream(sInput) >> dblTemp;
28      stringstream(sInput) >> iN;
29      iN = sInput.length();
30      if (iN < 4) {
31          again = true;
32          continue;
33      } else if (sInput[iN - 3] != '.') {
34          again = true;
35          continue;
36      } while (++iN < iN) {
37          if (isdigit(sInput[iN])) {
38              continue;
39          } else if (iN == (iN - 3)) {
40              continue;
41          }
42      }
```

4.2. Teste de Integração

Testa a integração de unidades. Qualquer teste que não é isolado de outras dependências pode ser entendido como um teste de integração, isto é, um teste que pode identificar erros da comunicação entre módulos do sistema, por exemplo, um teste de desempenho pode falhar devido a um erro no sistema. No entanto, vale ressaltar que para buscar erros de integração é recomendado que uma bateria especializada deste tipo de teste seja criada, ou seja, uma bateria exclusivamente para buscar erros de integração, sem outras finalidades.

4.3. Teste de Aceitação

São testes idealmente especificados por clientes ou usuário finais do sistema que definem os comportamentos esperados pelo sistema. Este tipo de teste verifica se o sistema atende ao comportamento desejado, o que o torna útil para identificar quando o projeto foi finalizado, já que quando o sistema passa em todos os testes de aceitação, então o sistema foi concluído. Os testes verificam se os resultados esperados não estão fora de um intervalo esperado, ou então verificam que os resultados obtidos atendem a certas propriedades relacionadas ao algoritmo.



4.4. Teste de Sistema

São testes realizados após a implantação do sistema em um ambiente idêntico ao de produção, incluindo o hardware, sistema operacional e outras dependências, visando buscar limitações de carga e incompatibilidade dos softwares. A automação de cada tipo de teste possui um conjunto de benefícios e um grau de dificuldade específico, então cada projeto precisa avaliar o custo-benefício para tomar as decisões corretas, já que é normal

ter que priorizar e optar por tarefas específicas em um projeto. Outro ponto a avaliar é o conhecimento do time em automatizar determinado tipo de teste, pois a automação pode trazer mais problemas caso os testes não sejam bem implementados.

4.5. Teste Alfa

Alfa é um teste de disponibilidade limitada realizado antes que as versões sejam liberadas para uso mais difundido. Nosso objetivo com o teste Alfa é verificar a funcionalidade e coletar feedback de um conjunto limitado de clientes. Normalmente a participação é por convite e está sujeita a termos de disponibilidade pré-geral. As versões Alfa podem não ter todos os recursos, não têm contrato de nível de serviço (SLA) e não há obrigatoriedade de suporte técnico. No entanto, geralmente elas são adequadas para uso em ambientes de teste.

4.6. Teste Beta

Na versão Beta, os produtos ou recursos estão prontos para testes e usos mais amplos pelos clientes. É comum as versões Beta serem anunciadas publicamente. Não há SLA em uma versão Beta, a menos que especificado de outra forma nos termos de produtos ou de um programa Beta específico. No entanto, o suporte técnico é limitado. Em média, a fase Beta dura cerca de seis meses.



4.7. Teste de Regressão

É uma destas atividades e uma das formas mais eficazes para se analisar a relação entre funcionalidades novas e antigas de sistemas sendo essencial para garantia a Confiabilidade da entrega final.

Para executar uma bateria de testes de regressão, antes de qualquer coisa, é sugerido que procedimentos de testes tenham sido criados previamente. Geralmente esses conjuntos de informações fazem parte das chamadas suítes de regressão, as quais armazenam comumente os casos de teste que devem ser executados.

Em seguida, é primordial que seja feita uma análise para selecionar os casos de teste candidatos para validar determinada versão do sistema ou funcionalidades de um ciclo de desenvolvimento, pois nem sempre todos os casos de teste precisam necessariamente ser utilizados.

Uma forma de fazer isso é listar aqueles que possuem maior risco, com maior possibilidade de revelar a presença de falhas e preferencialmente que estejam relacionados com as funcionalidades alteradas. Sendo assim, a empresa deve investir tempo e recursos onde os defeitos são mais prováveis e perigosos.

4.8. Normas ISO / IEEE Importantes no Teste de Software

Normas técnicas são documentos publicados por organizações profissionais responsáveis por **padronizar** determinadas atividades, processos, dispositivos, produtos, etc. Nessas normas são descritas as etapas de processos, entradas e saídas esperadas de cada etapa, as formas e conteúdos de documentações, bem como outras informações necessárias a atividade de teste.

O motivo da existência de normas é para que se tenha uma padronização nas formas com que são realizados processos, produzidos documentos, e para que cada termo tenha o mesmo significado e seja entendido em diferentes situações facilitando a comunicação tanto entre as equipes internas de uma organização, quanto entre organizações diferentes.

A utilização de normas garante que todas as organizações que realizam determinada ação a façam de maneira similar, e que a comunicação entre essas organizações seja

facilitada devido a essa similaridade. O uso de padrões também facilita a obtenção de um parâmetro de comparação para determinado atributo.

Nesse sentido, no teste de software e na computação em geral, utilizamos normas provenientes de duas instituições:

ISO: “International Organization for Standardization” ou em português, “Organização Internacional para Padronização”.



IEEE: “Institute of Electrical and Electronics Engineers”, pronuncia-se I-3-e, ou em português “Instituto de Engenheiros Eletricistas e Eletrônicos”.



As seguintes normas são fundamentais para o desenvolvimento de bons processos de teste de software:

ISO 9126-1-Software Quality Characteristics (2003):

Esse padrão define um modelo de qualidade o qual pode ser aplicado a qualquer tipo de software sem fazer especificações sobre os requisitos desse produto. O objetivo desse padrão é proporcionar um framework para avaliação da qualidade dos produtos de software. Assim o modelo de qualidade definido por ele é baseado em seis características:

1. Funcionalidade: é a capacidade que o software tem de prover funções que atendam aos requisitos implícitos e explícitos. Pode ser entendido como “o que o software faz”.

2. Portabilidade: capacidade da transferência de um produto de software de um ambiente para outro.

3. Confiabilidade: capacidade que o produto de software tem de repetir sua funcionalidade dadas as mesmas condições antes aplicadas. Ou ainda, é a capacidade que um produto tem de executar determinada função sem sofrer desgaste ou envelhecimento.

4. Manutenibilidade: capacidade que o produto de software possui de ser modificado. As modificações incluem correções, melhorias ou adaptações.

5. Usabilidade: capacidade que o produto tem de ser compreendido por diversos tipos de usuários. O quão fácil é para que o usuário aprenda e possa utilizar esse produto.

6. Eficiência: capacidade do produto de apresentar um desempenho satisfatório quando lhe é proporcionado recursos suficientes.

Atualmente, essa norma foi substituída pela Norma ISO/IEC 25000. Na família de normas 25000, essas capacidades do software foram unidas a outras que vieram de outras normas.

5. TÉCNICAS DE TESTE

Objetivo

Ensinar os testes de Caixa Branca ou Teste de Aberta e o Teste de Caixa Preta. Teste de software não se trata apenas de encontrar bug, mas de investigar, analisar e garantir que a sua entrega será com qualidade em todos os aspectos possíveis (qualidade do código desenvolvido, dos requisitos acordados, dos padrões adotados e afins).

Introdução

A grande diferença entre as técnicas é o fato de olhar para o código e poder codificar, por isso tem o nome de caixa aberta e caixa preta. A busca por uma entrega cada vez mais ágil, é oportuno falarmos que nada adianta entregarmos algo que está em desacordo com o que foi combinado com os stakeholders.

5.1. Teste de caixa branca

O teste de caixa branca possui esse nome porque o testador tem acesso à estrutura interna da aplicação. Logo, seu foco é garantir que os componentes do software estejam concisos. Nesse sentido, esse tipo de teste também é conhecido como **teste estrutural** ou **caixa de vidro**, já que busca garantir a qualidade na implementação do sistema. Logo, ele tem por objetivo validar, apenas, a lógica do produto.

A partir disso, ele é comumente realizado utilizando-se o código-fonte. Portanto, exige mais conhecimento técnico por parte do testador, sem contar o maior custo. Por ser baseado na implementação, quando a alteramos, também precisamos alterar o teste.

Tendo isso em mente, existem algumas práticas que visam amplificar a efetividade do teste de caixa branca. Desse modo, veremos duas, a saber: teste de condição e teste de ciclo.

Teste de condição

Essa técnica é simples, pois sua proposta é avaliar se os operadores/variáveis lógicos (booleanos – true/false) estão consistentes.

Teste de ciclo

Utiliza a estrutura de repetição (for/while)? De modo bem simples, é justamente isso que essa técnica faz: valida estruturas de repetição. Para isso, ela divide os ciclos em 4 tipos: desestruturado, simples, aninhado e concatenado. O ciclo desestruturado nada mais é do que o conjunto de blocos de repetição utilizados de maneira desordenada. Por conta disso, ao ser identificado, deve ser reestruturado, já que aumenta consideravelmente o custo dos testes e da manutenção do sistema. Já o ciclo simples, como o próprio nome diz, é apenas uma estrutura de repetição sendo testada. Ciclos aninhados são ciclos dentro de ciclos. E, por último, mas não menos importante, os ciclos concatenados no teste de caixa branca são estruturas de repetição dependentes, ou seja, para testar o bloco 2, eu preciso garantir que o bloco 1 é coerente.

5.2. Teste de caixa preta

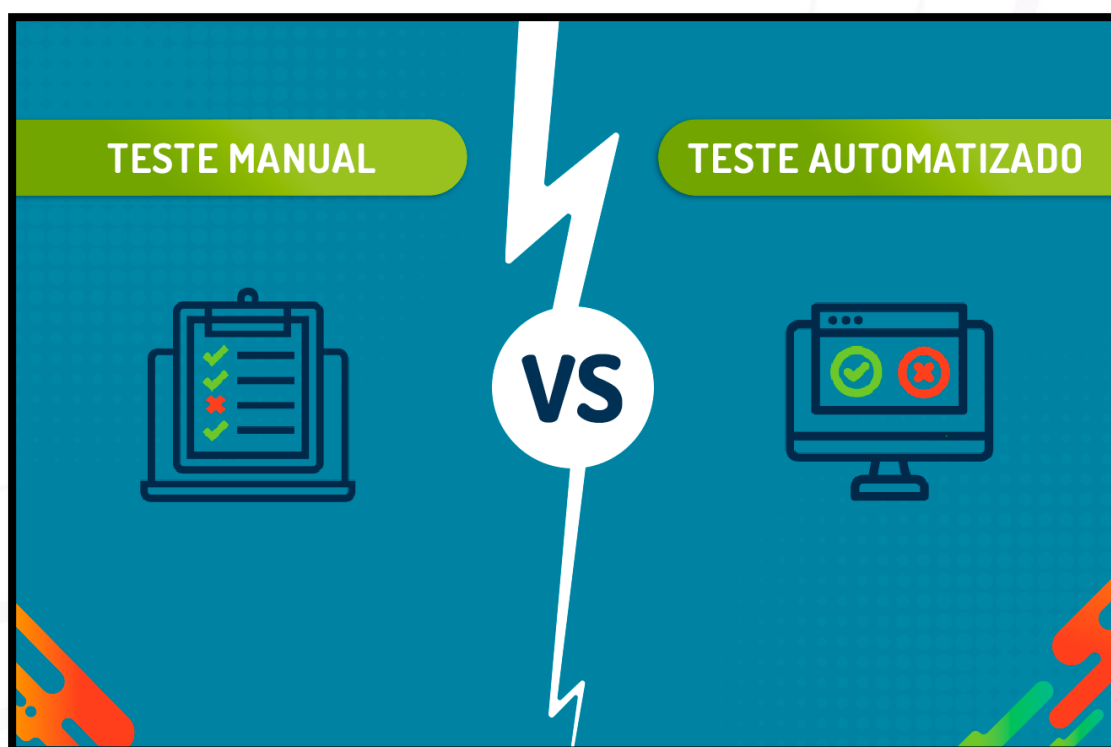
Diferente do teste de caixa branca, ele possui esse nome porque o código-fonte é ignorado no teste. Assim, ao se utilizar dessa técnica, o Tester não está preocupado com os elementos constitutivos do software, mas em como ele funciona. Nesse sentido, esse tipo de teste também é conhecido como teste funcional, já que busca garantir que os requisitos funcionais do produto estão consistentes. A partir disso, ele é comumente realizado utilizando-se da experiência do usuário, ou seja, através da interface do produto. Com isso, para aumentarmos a qualidade e, consequentemente, blindarmos o software de falhas, entendemos que todas as entradas/saídas possíveis precisam ser testadas. Contudo, sabemos que isso — na grande parte dos casos — é humanamente impossível. Ademais, a falta de clareza dos requisitos pode (e vai) impactar nas entradas e saídas aceitas para o teste.

Isso quer dizer que, além da volumetria de dados que teremos que validar, eles podem não ser adotados nos testes. Por exemplo: imagine que você utiliza apenas números para testar um campo de CPF, mas o desenvolvedor utiliza o campo como string (caracteres alfanuméricos).

5.3. Testes Automatizados

Os testes automatizados são definidos como utilização de ferramentas de testes que possibilitem simular usuários ou atividades humanas de forma a não requerer

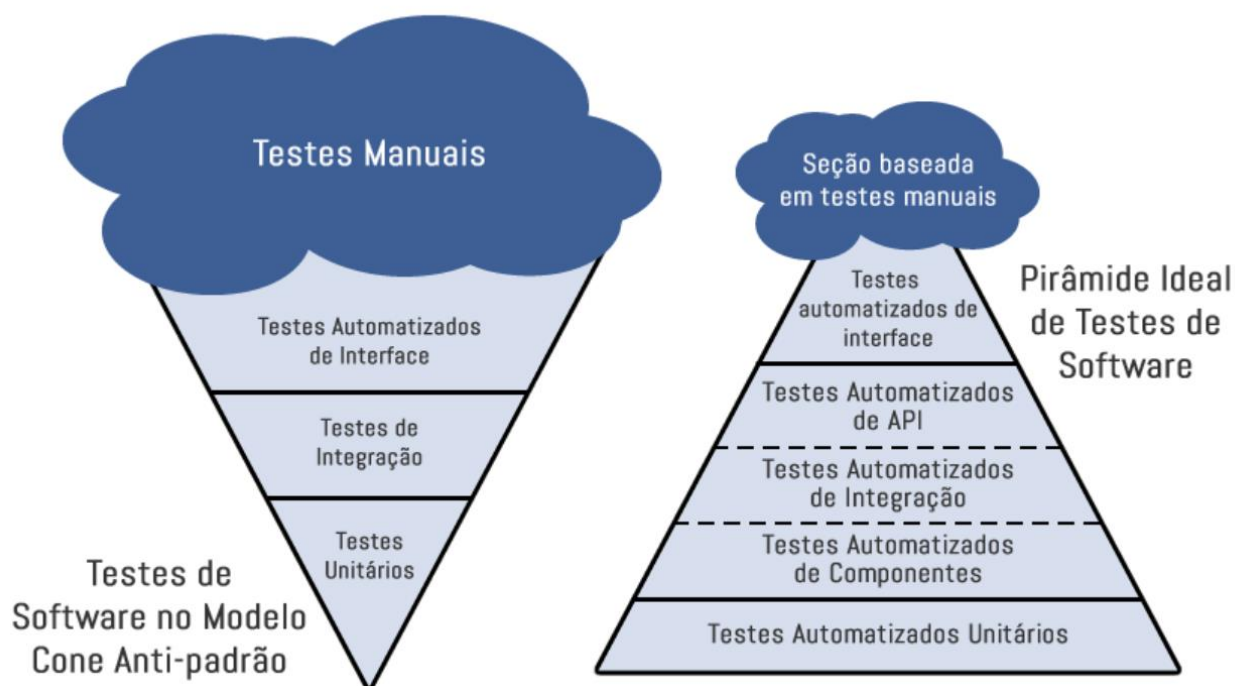
procedimentos manuais no processo de execução dos testes. Sendo um dos motivos para sua utilização é que a automação exige um esforço inicial de criação, porém possibilita uma incomparável eficiência e confiabilidade, impossível de ser atingida com procedimentos manuais. A automação de testes não faz parte de um nível específico, sendo que o mesmo garante que testes de regressão sejam feitos. Controlar a qualidade de sistemas de software é um grande desafio devido à alta complexidade dos produtos e às inúmeras dificuldades relacionadas ao processo de desenvolvimento, que envolve questões humanas, técnicas, burocráticas, de negócio e políticas. Idealmente, os sistemas de software devem não só fazer corretamente o que o cliente precisa, mas também fazê-lo de forma segura, eficiente e escalável e serem flexíveis, de fácil manutenção e evolução. Salvo honrosas exceções, na indústria de software brasileira, estas características são muitas vezes asseguradas através de testes manuais do sistema após o término de módulos específicos ou até mesmo do sistema inteiro.



5.3.1. História

A ideia de testar manualmente sempre existiu, desde da época dos cartões perfurados até os softwares de milhões de linhas de código, pois é uma prática básica e natural das ciências exatas. A necessidade de criar scripts ou programas para testar cenários específicos também não é nova, no entanto, a ideia de criar testes automatizados

como um módulo adicional do sistema a ser testado é relativamente nova, com início em meados da década de 90. Muitas práticas de desenvolvedores já evidenciavam a necessidade da criação destes testes, tais como trechos de código utilizados para imprimir valores de variáveis e métodos main espalhados em trechos internos do código fonte para fazer execuções pontuais do programa. A prática de testes automatizados se popularizou devido a linguagens de programações que facilitam a escrita e execução dos testes, além de não poluir o código fonte e não interferir no comportamento original do sistema. Muito tem se estudado, novos conceitos e técnicas de desenvolvimento foram criadas, e as ferramentas estão cada vez mais poderosas, facilitando não apenas a escrita de testes para trechos específicos de código, mas também para a integração de módulos, interfaces gráficas e bancos de dados.



6. TIPOS DE TESTES

Objetivo

Nesta aula você irá entender sobre o que testar em um software e qual será o foco do teste. O objetivo principal do processo de teste de software é detectar a presença de erros no sistema testado.

Introdução

Um tipo de teste é um grupo de atividades, destinado a testar características específicas de um sistema de software, ou parte, com base em objetivos de teste específicos.

6.1. Teste de Funcionalidade

Tem como função avaliar as funções do sistema observando se estão funcionando corretamente. Envolvendo testes anteriores como por exemplo, os testes de unidade, de integração, de sistema e etc. Testes de funcionalidade dão prioridade a navegação e as interações.

6.2. Teste de Usabilidade

São aspectos que envolvem a experiência do usuário ao utilizar o sistema, ou seja, qual será a facilidade de usabilidade do sistema. O produto, que pode ser um site, uma aplicação web, um produto físico, não precisa estar completamente desenvolvido para ser testado.

6.3. Teste de Desempenho

Os testes de desempenho mostram o que determinada solução precisa para ser melhorada, antes de ser disponibilizada para o público. Sem a aplicação desses testes, o software pode apresentar diversos problemas, como:

- Ficar lento com uma grande quantidade de usuários acessando ao mesmo tempo;
- Apresentar inconsistências entre plataformas diferentes;
- Possuir falhas e funcionalidades faltando ou inoperantes.

Aplicações que chegam ao mercado sem ter realizado testes de desempenho bem embasados podem gerar péssimos resultados, prejudicar a marca da empresa e ainda impactar negativamente na receita.

6.3.1. *Tipos de testes de desempenho*

Dentro dos testes de desempenho, existem divisões de acordo com o objetivo. Confira:

Testes de Carga

Testes de Carga são usados para avaliar a capacidade da aplicação de manter a qualidade de desempenho diante de quantidades diversas de usuários.

Testes de Stress

Com testes de stress, o objetivo é definir como a solução irá desempenhar diante de uma quantidade elevada de tráfego ou processamento de dados. O objetivo é encontrar o limite da aplicação.

Testes de Volume

Os testes de volume avaliam a quantidade de dados que o sistema pode gerenciar. O objetivo é verificar a capacidade da solução de acordo com seus requisitos e encontrar gargalos.

Testes de Escalabilidade

O objetivo dos testes de escalabilidade é determinar a eficácia da aplicação software diante de um aumento na “ampliação” para suportar um aumento na carga do usuário. Isso ajuda a planejar a adição de capacidade ao seu sistema de software.

Testes de Performance

Para identificar o tempo mínimo de resposta, número de usuários simultâneos e quantidade de transações por segundo, esses testes são aplicados.

6.4. **Teste de segurança**

Um dos mais importantes tipos de teste da atualidade. Não é à toa que nos últimos anos foram criadas legislações como a **Lei Geral de Proteção de Dados (LGPD)**, no Brasil,

e a **General Data Protection Regulation (GDPR)**, na União Europeia. É essencial se preocupar com segurança de dados e informações!



O teste de segurança avalia se o sistema e os dados são acessados de maneira segura e apenas pelo autor das ações. O objetivo é evitar que algum fraudador intercepte a informação que está sendo trafegada ou que colete dados sensíveis.

6.5. Teste de Portabilidade

Avalia o funcionamento do sistema em diferentes plataformas e dispositivos nas quais o sistema está proposto a funcionar. É importante estar atento ao desempenho do sistema em todas as plataformas de pesquisa. Nos dias de hoje, com a automação de testes ficou mais fácil. Porém, no Brasil, ainda é pouco explorada.

6.6. Teste de Stress

Observa o comportamento do sistema em condições extremas. Testando os limites do software analisando seu desempenho verificando até onde o software pode ser exigido e quais as falhas decorrentes do teste.

Os testes de estresse são fundamentais em aplicações em que a eficiência seja uma característica importante. Por exemplo:

- Servidores de arquivos e servidores web, que devem atender a solicitações de um
- Grande número de clientes;
- Aplicações industriais, tais como o controle de uma refinaria de petróleo;
- Jogos de computador, que precisam de um desempenho aceitável para serem viáveis comercialmente.

6.7. Selenium

Selenium é um conjunto de ferramentas de código aberto multiplataforma, usado para testar aplicações web pelo browser de forma automatizada. Ele executa testes de funcionalidades da aplicação web e testes de compatibilidade entre browser e plataformas diferentes. O Selenium suporta diversas linguagens de programação e vários navegadores web como o Chrome e o Firefox.

Selenium IDE

Selenium IDE é um ambiente integrado de desenvolvimento para scripts de testes automatizados, onde permite o usuário criar testes de forma muito rápida. Essa ferramenta te permite gravar os scripts, funcionando como um recorder, gravando as ações do usuário.

Esse script gerado você pode parametrizar e executar quantas vezes você quiser. O Selenium IDE inclui o Selenium Core, permitindo que você facilmente possa gravar e reproduzir os testes no ambiente real que será executado. O IDE é voltado para testes rápidos, com rápidos feedbacks.

É importante ressaltar que o Selenium IDE funcionava até a versão 55 do Firefox - e somente no Firefox. Porém, em 2018 eles voltaram com o projeto, ainda em “alpha”, mas agora disponível para Firefox e Chrome.

Selenium WebDriver

O Selenium WebDriver usa o próprio driver do navegador para a automação. É a forma mais moderna de interação atualmente, pois cada browser possui o seu respectivo driver, permitindo a interação entre o script de teste e o respectivo browser.

Esta versão é indicada para testes mais elaborados e por usuários familiarizados com a ferramenta. Geralmente usa-se o Selenium IDE para testes básicos, exporta-se o script e depois edita-se o script para realizar testes mais elaborados.

Selenium Grid

No Selenium Grid, você lança seu script sobre diferentes navegadores. O Grid é voltado para clusterização, permitindo você realizar os testes em máquinas diferentes de forma remota.



Referências

Bibliografia

- PRESSMAN, Roger S. Engenharia de Software. 7. ed. McGraw-Hill – Artmed, 2011.
- SOMMERVILLE, Ian. Engenharia de Software. 9. ed. Pearson Education do Brasil, 2011.
- ENGHOLM JR., Hélio. Engenharia de software na prática - São Paulo, SP: Novatec, 2010
- SCHACH, Stephen R. Engenharia de Software: os paradigmas classic e orientado a objetos. 7. ed. McGraw-Hill - Artmed, 2009.
- KOSCIANSKI, André; SOARES, Michel dos Santos. Qualidade de Software - Aprenda as Metodologias e Técnicas Mais Modernas para o Desenvolvimento. Novatec, 2007.
- HIRAMA, Kechi. Engenharia de Software: qualidade e produtividade com tecnologia. Campus, 2011.