



TESTE E QUALIDADE DE SOFTWARE

Professora Esp. Janaina Aparecida de Freitas

REITOR	Prof. Ms. Gilmar de Oliveira
DIRETOR DE ENSINO PRESENCIAL	Prof. Ms. Daniel de Lima
DIRETORA DE ENSINO EAD	Prof. Dra. Geani Andrea Linde Colauto
DIRETOR FINANCEIRO EAD	Prof. Eduardo Luiz Campano Santini
DIRETOR ADMINISTRATIVO	Guilherme Esquivel
SECRETÁRIO ACADÊMICO	Tiago Pereira da Silva
COORDENAÇÃO DE ENSINO, PESQUISA E EXTENSÃO	Prof. Dr. Hudson Sérgio de Souza
COORDENAÇÃO ADJUNTA DE ENSINO	Prof. Dra. Nelma Sgarbosa Roman de Araújo
COORDENAÇÃO ADJUNTA DE PESQUISA	Prof. Ms. Luciana Moraes
COORDENAÇÃO ADJUNTA DE EXTENSÃO	Prof. Ms. Jeferson de Souza Sá
COORDENAÇÃO DO NÚCLEO DE EDUCAÇÃO A DISTÂNCIA	Prof. Me. Jorge Luiz Garcia Van Dal
COORDENAÇÃO DOS CURSOS - ÁREAS DE GESTÃO E CIÊNCIAS SOCIAIS	Prof. Dra. Ariane Maria Machado de Oliveira
COORDENAÇÃO DOS CURSOS - ÁREAS DE T.I E ENGENHARIAS	Prof. Me. Arthur Rosinski do Nascimento
COORDENAÇÃO DOS CURSOS - ÁREAS DE SAÚDE E LICENCIATURAS	Prof. Dra. Katiúscia Kelli Montanari Coelho
COORDENAÇÃO DO DEPTO. DE PRODUÇÃO DE MATERIAIS	Luiz Fernando Freitas
REVISÃO ORTOGRÁFICA E NORMATIVA	Beatriz Longen Rohling Caroline da Silva Marques Carolayne Beatriz da Silva Cavalcante Marcelino Fernando Rodrigues Santos Eduardo Alves de Oliveira Jéssica Eugênio Azevedo Kauê Berto André Dudatt Carlos Firmino de Oliveira Vitor Amaral Poltronieri Carlos Eduardo da Silva Carlos Henrique Moraes dos Anjos Yan Allef
PROJETO GRÁFICO E DIAGRAMAÇÃO	
ESTÚDIO, PRODUÇÃO E EDIÇÃO DE VÍDEO	

FICHA CATALOGRÁFICA

Dados Internacionais de Catalogação na Publicação - CIP

F862t Freitas, Janaina Aparecida de
Teste e qualidade de software / Janaina Aparecida de Freitas.
Paraná: EduFatec, 2023.
92 p. : il. Color.

1. Software - Testes. 2. Software - Controle de qualidade. I.
Centro Universitário Unifatec. II. Núcleo de Educação a
Distância. III. Título.

CDD: 23 ed. 005.14

Catalogação na publicação: Zineide Pereira dos Santos - CRB 9/1577

**As imagens utilizadas neste material didático
são oriundas dos bancos de imagens
[Shutterstock](#).**

AUTORA

Professora Esp. Janaina Aparecida de Freitas

Graduada em Informática pela Universidade Estadual de Maringá (UEM/2010), e depois fiz uma Especialização MBA em Teste de Software pela UniCeuma de Brasília, pois na época fiz parte da equipe de qualidade da empresa de desenvolvimento de sistemas e depois trabalhei como Analista de Sistemas. Atue no mercado gráfico trabalhando com gráficas como Design e em jornal como diagramadora. Depois, do mercado de trabalho ingressei na carreira docente, cursando a Pós em Docência no Ensino Superior: Tecnologias Educacionais e Inovação pela UniCesumar. e mais uma Pós na área de Gerenciamento de Banco de Dados pela UniCesumar. Atualmente sou graduanda em Letras pela UniCesumar e sou aluna não regular do programa de Mestrado em Ciências da Computação pela Universidade Estadual de Maringá (UEM). Tenho experiência na área de educação desde 1995 sendo estas em escolas profissionalizantes de informática e atualmente atuando no EAD-Unicesumar desde junho de 2015 como professora mediadora. Atuando também como formadora e conteudista de disciplinas dos cursos de Engenharia de Software, Análise e Desenvolvimento de Sistemas, Sistemas para Internet e Gestão da Tecnologia.

CURRÍCULO LATTES: <https://lattes.cnpq.br/4906244382612830>

APRESENTAÇÃO DO MATERIAL

Seja muito bem-vindo (a)!

Prezado (a) aluno (a), se você se interessou pelo assunto desta disciplina, isso já é o início de uma grande jornada que vamos trilhar juntos a partir de agora. Proponho, junto com você, construir nosso conhecimento sobre os principais conceitos a respeito da qualidade de software e teste de software.

É importante entender as definições teóricas sobre o que é qualidade de software e porque ela é importante para a empresa e para os usuários e como está presente no nosso cotidiano e também entender a sua importância na Engenharia de Software.

Na unidade I começaremos a nossa jornada pelos conceitos sobre qualidade, e entenderemos como ela é o resultado de atividades que foram realizadas no processo de desenvolvimento, pois quando o cliente solicita um software, faz-se necessário garantir que o software atenda a essa necessidade. Também falaremos sobre os conceitos de qualidade de processo x qualidade de produto x qualidade de software e processos de gerência da qualidade de software.

Já na unidade II vamos ampliar nossos conhecimentos básicos sobre as métricas de software e conhecer os principais frameworks usados para as métricas de produto. Também iremos conhecer as métricas usadas para o modelo de requisitos e para o modelo projeto de software e entender sobre os fundamentos da revisão de software, além de conceitos básicos usados na garantia de qualidade de software e seus elementos.

Nas unidades III e IV, vamos tratar sobre teste de software e todo o seu processo. Começamos com os conceitos básicos sobre Teste de software e seus objetivos para entender como funciona o processo do Ciclo de Vida do Teste de Software. E ampliando nosso conhecimento, vamos entender as técnicas de teste de software que podem ser aplicadas neste processo e quais as principais ferramentas para automatizar os testes de software.

E para finalizar, conhecer os conceitos sobre o processo de teste de software e como funciona um ambiente de teste de software e como é o trabalho em equipe. Além de aprofundar os conhecimentos sobre as métricas e medição que são usadas no teste de software e como é a gerência de risco em teste de software e a sua importância para as empresas.

Aproveito para reforçar o convite a você, para junto conosco percorrer esta jornada de conhecimento e multiplicar os conhecimentos sobre tantos assuntos abordados em nosso material. Esperamos contribuir para seu crescimento pessoal e profissional.

Muito obrigado e bom estudo!

SUMÁRIO





UNIDADE

QUALIDADE DE SOFTWARE

Professora Esp. Janaina Aparecida de Freitas

Plano de Estudos

- Conceitos fundamentais sobre a Qualidade de Software.
- Importância da Qualidade de Software na Engenharia de Software.
- Conceitos sobre Qualidade, Qualidade de Processos e Qualidade de Produto.
- Processos de Gerência da Qualidade de Software.

Objetivos da Aprendizagem

- Conceituar e contextualizar sobre a Qualidade de Software;
- Compreender a importância da Qualidade de Software na Engenharia de Software;
- Entender os conceitos que envolvem a Qualidade, a Qualidade de Processos e a Qualidade de Produto;
- Compreender sobre os processos da Gerência da qualidade de software.

INTRODUÇÃO

Olá, aluno (a), seja bem-vindo(a) ao estudo sobre qualidade de software.

Você já pensou em quantos produtos ou serviços fazem parte do seu dia a dia? Quando você adquire esses produtos ou serviços, espera que tenham qualidade e não apresentem defeitos. É neste momento que te convido a pensar sobre o significado da palavra “qualidade” e como ela está presente em nossas vidas.

Nessa primeira unidade pretendo que você comprehenda os conceitos sobre qualidade, o porquê ela é importante para a empresa e para os usuários e como está presente no nosso cotidiano.

No primeiro tópico da nossa unidade vamos conceituar e contextualizar sobre a Qualidade de Software. Ela é o resultado de atividades que foram realizadas no processo de desenvolvimento, pois quando o cliente solicita um software, faz-se necessário garantir que o software atenda a essa necessidade.

No segundo tópico vamos compreender a importância da Qualidade de Software na Engenharia de Software. Vamos entender que a qualidade está relacionada com as necessidades de cada cliente, podendo ser influenciada por fatores como cultura, tipo de produto ou de serviço prestado, percepções e necessidades de cada usuário.

No terceiro tópico vamos entender os conceitos que envolvem a Qualidade, a Qualidade de Processos e a Qualidade de Produto. Aprenderemos que o produto desenvolvido deverá ser avaliado, analisando se o que foi produzido está de acordo com as necessidades explícitas e implícitas do cliente.

No quarto e último tópico vamos compreender sobre os processos da Gerência da qualidade de software. O gerenciamento da qualidade busca assegurar que o produto de software satisfaça as necessidades do cliente, envolvendo todas as atividades do desenvolvimento por todo o seu ciclo de vida e para implementar se usa políticas e procedimentos com atividades de melhoria contínua de processos realizadas durante todo o projeto.

Está preparado?! Bons estudos!

QUALITY

CONCEITOS FUNDAMENTAIS SOBRE A QUALIDADE DE SOFTWARE



A qualidade é hoje considerada muito importante em todas as áreas. Todos querem produtos e serviços com qualidade. A qualidade de software é o resultado de atividades que foram realizadas no processo de desenvolvimento deste software. Quando se fala em qualidade de software é necessário lembrar que o projeto do software, o processo de desenvolvimento e o produto final têm que ter qualidade também.

Quando se adquire um produto ou serviço, deseja-se que esse produto esteja em pleno funcionamento, de acordo com aquilo que foi solicitado e livre de falhas. E no caso de um software, a realidade deve ser a mesma, pois quando o cliente solicita um software, ele cria a expectativa e, por isso, faz-se necessário garantir que o software atenda a essa necessidade (ZANIN *et al*, 2018).

Quando se usa a palavra qualidade, parece que ela tem um significado bem claro, contudo, trata-se de algo complexo e de difícil mensuração, principalmente quando nos referimos a qualidade de software. A razão disso é que a qualidade é relativa e pode assumir diversos valores e sentido, de acordo com o usuário.

A qualidade vai depender do ponto de vista de quem avalia o software, e temos: usuários, desenvolvedores e empresas que podem ter necessidades diferentes (quadro 1):

QUADRO 1 - PONTO DE VISTA DE QUEM AVALIA O SOFTWARE

Usuário	Procura avaliar o software sem conhecer seus aspectos internos. Seu interesse está apenas na facilidade de uso, no desempenho, na confiabilidade dos resultados e principalmente no preço
Desenvolvedores	Procuram avaliar os aspectos de conformidade em relação aos requisitos dos clientes que foram coletados e nos aspectos internos do software
Empresa	Procura avaliar os aspectos de conformidade em relação aos requisitos coletados dos clientes e dos desenvolvedores e também aspectos relacionados ao custo e ao cronograma de entrega.

Fonte: Adaptado de Pressman e Maxim (2016, p. 415).

Nesse sentido, a qualidade do software pode ser medida de acordo com o quanto ela está em conformidade com o que o cliente solicitou.

E os desenvolvedores de software concordam que um software de alta qualidade é um objetivo importante para as empresas desenvolvedoras. E como definir a qualidade de software? Em linhas gerais, para Pressman e Maxim (2016, p. 415) a "qualidade de software pode ser definida de como uma gestão de qualidade efetiva aplicada de modo a criar um produto útil que forneça valor mensurável para aqueles que o produzem e para aqueles que o utilizam".

Entretanto, o termo qualidade de software possui várias definições na literatura. A seguir algumas que são bem utilizadas:

- “Qualidade de Software é um conjunto de propriedades a serem satisfeitas em um determinado grau de modo que o software satisfaça às necessidades de seus usuários” (GUERRA; COLOMBO, 2000).
- “Qualidade de Software é a conformidade a requisitos funcionais e de desempenho explicitamente estabelecidos, a padrões de desenvolvimento explicitamente documentados, e a características implícitas que são esperadas de todo software desenvolvido profissionalmente” (PRESSMAN; MAXIM, 2016).

O objetivo de qualquer empresa que desenvolve sistemas é produzir software com qualidade, dentro dos prazos estabelecidos e que obtenha a satisfação do cliente. Para isso, devemos produzir o software de acordo com os requisitos que foram coletados e desenvolvê-lo nos prazos estabelecidos e com um nível mínimo de defeitos para a satisfação do cliente (PRESSMAN; MAXIM, 2016).

De acordo com Zanin et al (2018, p. 12) os "projetos de software contam com vários desafios para entregar o software em perfeito funcionamento, devido a uma série de fatores — em especial, a complexidade". E essa complexidade é porque construir um sistema envolve muitas habilidades, entre elas a comunicação e interpretação, e tudo para conseguir entender o que o cliente deseja, considerada uma das fases mais críticas, além de habilidades específicas para as fases de implementação, entre outras.

E como garantir a qualidade de um software diante das complexidades? É um desafio enorme, pois precisamos estabelecer uma cultura de não tolerância a erros, por meio de processos que possam ajudar a inibir ou impedir falhas.

Temos basicamente dois tipos de qualidade: interna e externa. A **qualidade interna** de um software é uma avaliação de seus componentes estruturais, arquitetura, estilo de codificação, adequação a padrões e métricas estabelecidas. A **qualidade externa** já está mais ligada à funcionalidade do software, facilidade de uso, tempo de resposta, precisão de resultados. Normalmente, uma boa qualidade interna leva a uma boa qualidade externa. Mas nem sempre isso ocorre (PRESSMAN; MAXIM, 2016).

Segundo (ZANIN et al, 2018) a preocupação com a qualidade deve estar voltada para a melhoria do processo que envolve o desenvolvimento do software. Pois, se garantirmos a qualidade do processo, podemos também garantir a qualidade do software.

Dessa forma, programas de melhorias de qualidade, modelos e normas propiciam aumento de produtividade, redução no número de defeitos no software, maior previsão e visibilidade dos processos definidos, além do cumprimento das metas de custo, prazo, funcionalidades e um aumento na motivação da equipe desenvolvedora.

Como toda norma e qualquer modelo, o mesmo não é perfeito e completo, pois são idealizados com base em diversas considerações e práticas. As diversas normas e modelos existentes possuem vantagens e desvantagens na sua implantação e isto também deve ser considerado na decisão da escolha do mais apropriado a cada empresa (SOMMERVILLE, 2018).

Atualmente, as principais normas e padrões para melhoria da qualidade de software utilizadas pelas empresas que desenvolvem ou adquirem produtos de software são o modelo

CMMI, as normas ISO/IEC, PSP e MPSBr. Entre as ISO, temos o modelo de qualidade ISO 25010 que é a mais nova norma (criada em 2011 e revisada em 2017).

De acordo com Pressman e Maxim (2018) esta norma define dois modelos de qualidade:

- I. modelo da qualidade em uso que descreve cinco características apropriadas quando consideramos utilizar o produto em um determinado contexto.
- II. modelo da qualidade de produto que descreve oito características que enfocam a natureza dinâmica e a estática dos sistemas de computador.

O que podemos considerar para dizer que um software é “bom o suficiente”? De acordo com Pressman e Maxim (2016, p. 449) software bom o suficiente “fornece funções e características de alta qualidade que os usuários desejam, mas, ao mesmo tempo, fornece outras funções e características”. É o que se espera de um software e que os usuários ignorem os erros pelo fato de estarem muito satisfeitos com as outras funcionalidades oferecidas pela aplicação. Mas é difícil ter um software bom o suficiente, pois isso pode funcionar em poucos casos e para algumas empresas de software e em um conjunto limitado de domínios de aplicação (PRESSMAN; MAXIM, 2016).

IMPORTÂNCIA DA QUALIDADE DE SOFTWARE NA ENGENHARIA DE SOFTWARE

Quando você usa um produto ou serviço, espera que ele tenha qualidade e não apresente defeitos. É neste momento que devemos pensar sobre o significado da palavra qualidade e como ela está presente no nosso cotidiano.

E para as empresas, com a crescente competitividade no mercado, as empresas de TI se obrigam cada vez mais a desenvolver produtos com qualidade para oferecer a seus clientes. E a qualidade hoje deixou de ser um diferencial e passou a ser um pré-requisito básico para qualquer produto ou serviço. E pensando sobre essa informação, qual a importância da qualidade de software na Engenharia de Software?

Para responder essa pergunta vamos pensar em um software com qualidade, onde tem a satisfação total do cliente, é de fácil manutenção e não tem possíveis falhas. Esse cenário é o ideal e, por isso, a qualidade de software mostra a sua grande importância no processo de desenvolvimento, já que a sua função é garantir que o software seja entregue ao cliente atendendo a todas as suas expectativas e necessidades e que mesmo sem saber, o cliente receberá um produto com qualidade (SOMMERVILLE, 2018).

E outro ponto para responder a pergunta é que a disseminação do uso do software em todas as áreas de negócio, aumentou a importância da qualidade de software. E na medida em que ocorre essa disseminação do uso de software, por exemplo, por sistemas complexos, a qualidade se torna um fator essencial no desenvolvimento dele. Existem algumas razões que devemos considerar, com relação a qualidade de software.

QUADRO 2 - RAZÕES PARA CONSIDERAR EM RELAÇÃO À QUALIDADE

Razão	Descrição
Competitividade	Forma do software com qualidade se destacar no mercado.
Sobrevivência	Clientes buscam software com qualidade, e a empresa precisa buscar formas de sobreviver no mercado desenvolvendo software com qualidade.
Essencial	Essencial ter qualidade principalmente para o mercado internacional.
Custo/benefício:	Sistema de qualidade direciona para o aumento de produtividade e tem redução de custos.
Retenção de Consumidores	A preocupação com a qualidade aumenta a satisfação e assegura os consumidores por mais tempo.

Fonte: Adaptado de Pressman e Maxim (2016, p. 451-452).

Mas será que temos mais razões ou pontos a considerar na importância da qualidade de software na Engenharia de Software? Sim, uma das razões é que precisamos pensar na qualidade de software na visão de quem usa, o usuário e o papel dele no desenvolvimento do software é importantíssimo para o sucesso do produto.

Você já parou para pensar que cada usuário tem desejos e necessidades diferentes em relação ao mesmo tipo de produto de software? E, neste momento, temos que pensar qual a melhor forma de atender a esses desejos e interesses. Portanto, é importante que você considere que o usuário tem o direito de participar e opinar durante o processo de desenvolvimento do software. E isso tem permitido que as empresas desenvolvedoras de software diminuam custos, ampliem as equipes de teste, garantam melhor qualidade para terem maiores lucros (PRESSMAN; MAXIM, 2016).

Outra razão a considerar é a importância dos requisitos na qualidade de software. Você já pensou por que é tão difícil entender, claramente, o que o cliente necessita? O que ele deseja? E para responder a esta pergunta precisamos ter em mente que entender os requisitos de um cliente é uma das tarefas mais difíceis enfrentadas por um engenheiro de software. Além das muitas dificuldades para fazer o levantamento das necessidades e desejos do cliente e para entender as informações por ele transmitidas, o sistema muda ao longo do projeto. É importante que o levantamento de requisitos seja realizado de forma

consistente e de acordo com o que o cliente solicitou, para evitar problemas e retrabalhos pela equipe de desenvolvimento, e, consequentemente, um aumento de custo em fases posteriores do desenvolvimento (ZANIN *et al*, 2018).

CONCEITOS SOBRE QUALIDADE, QUALIDADE DE PROCESSOS E QUALIDADE DE PRODUTO



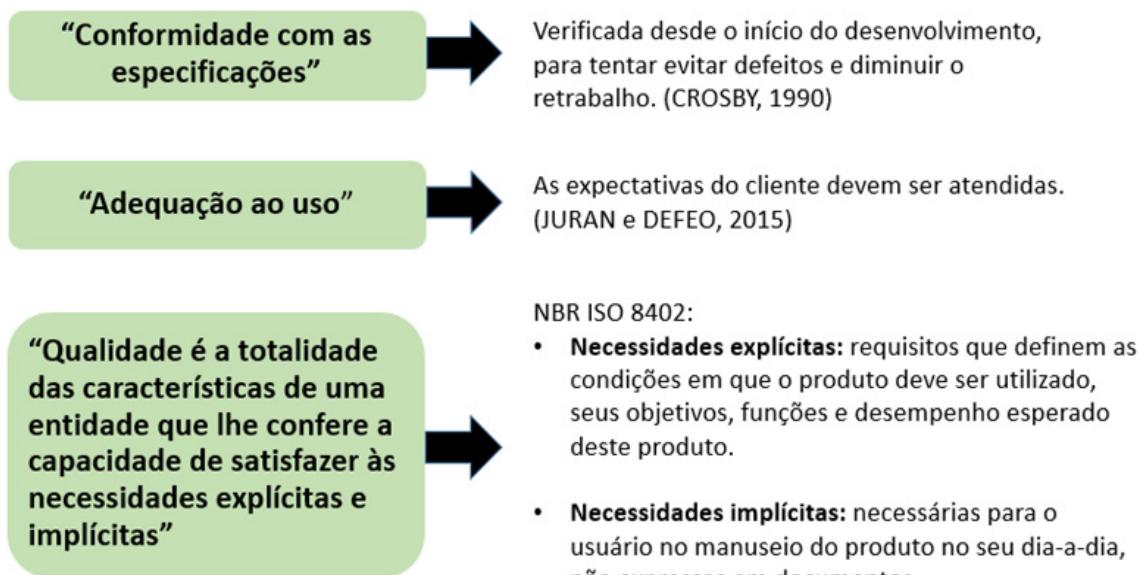
Hoje, a qualidade tem uma importância fundamental para alavancar a competitividade entre as empresas. Ela não é apenas um diferencial de mercado para que as empresas vendam e lucrem mais. Ela é um requisito para que as empresas consigam inserir seus produtos no mercado.

E para você entender mais sobre a qualidade, precisa entender os conceitos e as diferenças sobre qualidade, sobre a qualidade de processos, sobre a qualidade de produto e sobre a qualidade de software.

Vamos começar pelo termo “qualidade”, usado em expressões do tipo: “boa qualidade”, “má qualidade”. Segundo Guerra e Colombo (2000, p. 18) a “qualidade” é conformidade com requisitos, e estes devem estar definidos para permitir que sejam gerenciados com o uso de medidas, de forma a reduzir o retrabalho e aumentar a produtividade”.

A seguir, a figura 1 mostra algumas definições encontradas na literatura sobre o termo qualidade:

FIGURA 1- DEFINIÇÕES ENCONTRADAS NA LITERATURA SOBRE O TERMO QUALIDADE



Fonte: a autora.

Para Pressman e Maxim (2016) a qualidade vem sendo cada vez mais difundida nas empresas e a sua busca faz com que o mercado fique cada vez mais competitivo. E de acordo com a NBR ISO 9000 (2005) temos que a qualidade é “o grau no qual um conjunto de características inerentes satisfaz aos requisitos” e que o produto ou serviço atenderá a esses requisitos especificados.

Mas como reconhecer se o produto ou serviço tem qualidade? Responder a esta pergunta não é algo tão simples quanto se imagina. Você sabe o que é qualidade ao ver e, mesmo assim, é algo difícil de ser definido ou de mensurar. Entretanto, temos algumas maneira de identificar e definir, se algo tem qualidade (quadro 3):

QUADRO 3 - MANEIRAS DE IDENTIFICAR E DEFINIR A QUALIDADE

Visões	Descrição
Visão do usuário	O produto atende às metas específicas se ele atender às preferências do usuário.
Visão do fabricante	Conformidade com as especificações pré-definidas no início do projeto, se atende às especificações originais do produto.
Visão do Produto	A qualidade deve estar ligada às funções e recursos de um produto.
Visão baseada em valor	A base é o quanto um cliente estaria disposto a pagar por um produto (valor do produto)
Transcendental	A qualidade é algo que se reconhece imediatamente, mas não se consegue definir explicitamente. A qualidade não pode ser medida de maneira precisa, sendo reconhecida somente por meio do contato que o cliente terá com o produto.

Fonte: Pressman e Maxim (2016, p. 457).

E a qualidade do processo? Para falar sobre ela, primeiro vamos definir o que é um processo. Podemos definir processo de software como um conjunto de atividades, ações e tarefas que são realizadas na construção de um produto.

De acordo com Pressman e Maxim (2016, p. 458), é uma “série de atividades, práticas, eventos, ferramentas e métodos que garantem técnica e administrativamente que o software pode ser desenvolvido com qualidade e de maneira organizada, disciplinada e previsível”. E quando se trabalha em um projeto, é importante seguir alguns passos que auxilie a criar um produto ou serviço com qualidade dentro do prazo estabelecido.

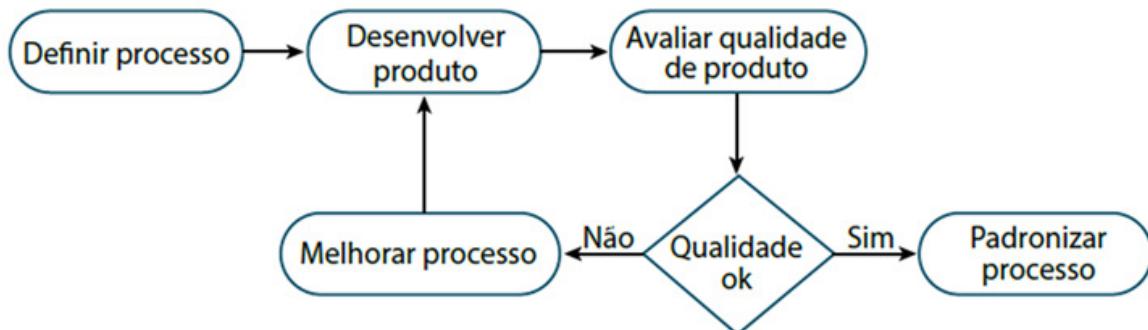
E quando se aplica a qualidade no processo de software, temos um aumento da qualidade do produto ou serviço e também uma redução de retrabalho, levando a uma diminuição do tempo de desenvolvimento e obtemos mais produtividade (SOMMERVILLE, 2018).

Conforme Sommerville (2011, p.472), “ameliorar o processo de software é importante para que defeitos no produto possam ser evitados ao máximo possível, aumentando a produtividade e facilitando a manutenção”. É importante que sejam implantados processos bem definidos e orientados por normas; as práticas para a qualidade devem ser usadas não somente nos processos, mas nos produtos de software, pois os mesmos podem ser avaliados segundo as normas internacionais de qualidade (SOMMERVILLE, 2011).

Qualidade de processo é uma área relacionada diretamente à Engenharia de Software, sendo que o estudo de uma área complementa o entendimento da outra. Nas duas áreas, são estudados os modelos do processo de desenvolvimento de software. E os modelos são uma forma de explicar, com mais detalhes, as etapas envolvidas no desenvolvimento de um software (PRESSMAN; MAXIM, 2016).

A figura a seguir, mostra a abordagem da qualidade baseada em processos. Onde se mede a qualidade e passa a alterar o processo até atingir o nível de qualidade requerida ou desejada.

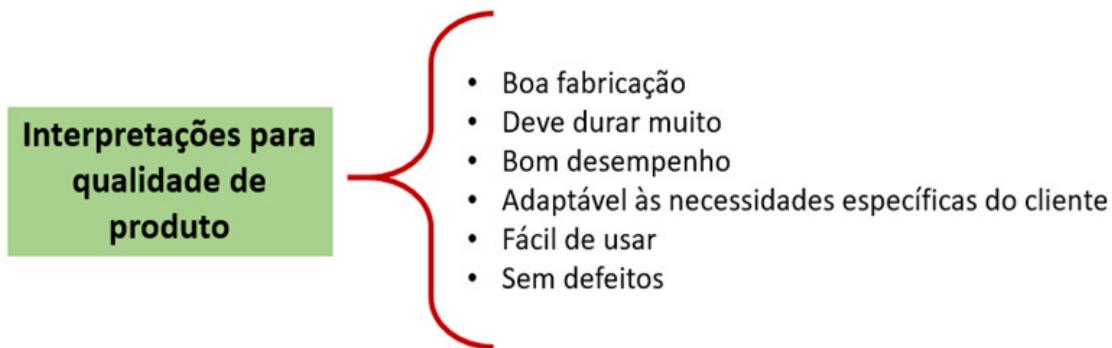
FIGURA 2 - QUALIDADE BASEADA EM PROCESSOS



Fonte: Sommerville (2018, p. 458).

E a qualidade do produto? A especificação deve ser mais precisa e detalhada e deve ser pensando durante todo o processo utilizado para seu desenvolvimento. A qualidade do produto deve estar em conformidade com requisitos que foram solicitados pelo cliente e bem definidos para que seja possível gerenciá-los, e com isso, reduzir o retrabalho e aumentar a produtividade da empresa desenvolvedora de software (GUERRA; COLOMBO, 2000).

FIGURA 3 - INTERPRETAÇÕES PARA QUALIDADE DE PRODUTO



Fonte: Adaptada de Pressman e Maxim (2016).

O Padrão ISO/IEC 9126 foi desenvolvido como uma tentativa de identificar os atributos fundamentais da qualidade do produto de software. E de acordo com o padrão, para que um produto de software seja considerado com qualidade deve possuir os seguintes atributos fundamentais de qualidade:

QUADRO 4 - ATRIBUTOS FUNDAMENTAIS DA QUALIDADE DE PRODUTO.

Atributo	Descrição
Funcionalidade	Capacidade de prover funcionalidades que satisfaçam o usuário em suas necessidades explícitas e implícitas, dentro de um determinado contexto de uso.
Confiabilidade	Se mantém no nível de desempenho nas condições estabelecidas, imune às falhas e sem defeitos.
Usabilidade	Capacidade de ser compreendido, de fácil aprendizado e uso atraente ao usuário.
Eficiência	Tempo de execução e os recursos envolvidos são compatíveis com o nível de desempenho do software.
Manutenibilidade	Capacidade de ser modificado, incluir as melhorias ou extensões de funcionalidade, fácil de corrigir defeitos, falhas ou erros.
Portabilidade	Capacidade de ser transferido de um ambiente para outro, de ser utilizado em várias plataformas.

Fonte: Pressman e Maxim (2016, p. 363).

O nível de qualidade de produto que se deseja alcançar deve estar de acordo com o que o mercado e o cliente buscam, ou seja, ir ao encontro dos requerimentos. Com essa definição para qualidade de produto, os requerimentos devem ser mensuráveis e os requerimentos do produto devem se encontrar. Neste caso, a qualidade existe ou não, pois podemos ter requerimentos extremamente completos e complexos ou bem simples.

FIGURA 4 - QUALIDADE DE PRODUTO.



Do **ponto de vista** de quem produz, qualidade é o encontro com os requerimentos e especificações é o fim, ou meta. (GUERRA e COLOMBO, 2000).

E a melhor forma de se conseguir alcançar a qualidade de produto é introduzir a qualidade ao longo do processo produtivo, desde sua concepção até a entrega do produto final (SOMMERVILLE, 2018).

Fonte: a autora.

Concluindo, a qualidade de um produto de software é resultante das atividades realizadas durante seu processo de desenvolvimento. A qualidade do produto de software é necessária para a empresa, e a qualidade do processo de desenvolvimento desse produto de software é ainda mais necessária e importante.

1

PROCESSOS DE GERÊNCIAS DA QUALIDADE DE SOFTWARE

Para Pressman e Maxim (2016, p. 427) a “qualidade de software não aparece simplesmente do nada. Ela é o resultado de um bom gerenciamento de projeto e uma prática consistente de engenharia de software”.

O gerenciamento da qualidade de projetos de software busca assegurar que o produto satisfaça as necessidades do cliente, envolvendo todas as atividades do desenvolvimento por todo o seu ciclo de vida (PMBOK - Project Management Body of Knowledge).

E para implementar o gerenciamento da qualidade se usa políticas e procedimentos com atividades de melhoria contínua de processos realizadas durante todo o projeto.

O gerenciamento e a prática devem ser aplicados no contexto das atividades que são: métodos de engenharia de software, técnicas de gerenciamento de projeto, ações de controle de qualidade e garantia da qualidade de software.

QUADRO 5 - ATIVIDADES DA GERÊNCIA DE SOFTWARE.

Atividades	Descrição
Métodos de engenharia de software	Onde temos a expectativa de construir software de alta qualidade e onde temos de entender o problema a ser resolvido.
Técnicas de gerenciamento de software	Técnicas que ajudam no impacto de decisões de gerenciamento inadequadas sobre a qualidade de software.
Controle de qualidade	Temos um conjunto de ações de engenharia de software que ajudam a garantir que cada produto resultante atinja suas metas de qualidade.
Garantia da qualidade	Estabelece a infraestrutura necessária para suportar métodos sólidos de engenharia de software, gerenciamento racional de projeto e ações de controle de qualidade.

Fonte: Pressman e Maxim (2016, p. 427).

Em controle de qualidade temos que ter modelos que são revistos de modo a garantir que sejam completos e consistentes. Além de ter o código inspecionado de modo a revelar e corrigir erros antes dos testes começarem.

Para Pressman e Maxim (2016, p. 428) pode-se aplicar uma “série de etapas de teste para descobrir erros na lógica de processamento, na manipulação de dados e na comunicação da interface”. Também pode-se usar uma combinação de medições e feedback que permite que se ajuste o processo quando qualquer um desses artefatos deixa de atender às metas estabelecidas para a qualidade.

E sobre a garantia da qualidade, ela tem como objetivo fornecer ao pessoal técnico e administrativo os dados necessários sobre a qualidade do produto e, com isso, ganhando entendimento e confiança de que as ações para atingir a qualidade desejada do produto estão funcionando. Esses dados fornecidos são de responsabilidade do gerenciamento, que trata dos problemas e aplica os recursos necessários para resolver os problemas de qualidade (PRESSMAN; MAXIM , 2016).

Podemos concluir que a qualidade é atingida por meio da aplicação de métodos de engenharia de software, técnicas, práticas administrativas consistentes e controle de qualidade completo. E todos suportados por uma infraestrutura que consiga garantir a qualidade de software.



Se produzirmos um sistema de software de péssima qualidade, perdemos porque ninguém vai querer comprá-lo. Se, por outro lado, depositamos um tempo infinito, um esforço extremamente grande e grandes somas de dinheiro para construir um software absolutamente perfeito, então, isso levará muito tempo para ser completado, e o custo de produção será tão alto que iremos à falência – ou perdemos a oportunidade de mercado, ou então, simplesmente esgotamos todos os nossos recursos. Dessa maneira, os profissionais desta área tentam encontrar aquele meio-termo mágico, onde o produto é suficientemente bom para não ser rejeitado logo de cara, como, por exemplo, durante uma avaliação, mas também não é o objeto de tamanho perfeccionismo e trabalho que levaria muito tempo ou que custaria demasiadamente para ser finalizado.

Fonte: Pressman, Roger, S. e Bruce R. Maxim. Engenharia de software. (9^a edição). Grupo A, 2021. (p. 365).



Entendemos que a qualidade de software não implica apenas se a funcionalidade de software foi corretamente implementada, mas também depende dos atributos não funcionais do sistema.

Fonte: Pressman, Roger, S. e Bruce R. Maxim. Engenharia de software. (9^a edição). Grupo A, 2021.

CONSIDERAÇÕES FINAIS

Chegamos ao final da nossa primeira unidade! Espero que você tenha aproveitado ao máximo os conhecimentos apresentados aqui, onde aprendemos sobre qualidade de software e seus conceitos.

Foram apresentados, inicialmente, os conceitos da qualidade e suas definições e por que o termo “qualidade” é importante e qual sua relação com a empresa e com os clientes.

Aprendemos sobre o termo “qualidade”, o que ele significa e como ele é usado para indicar que um produto ou serviço tem excelência e que está de acordo com o que foi definido em suas especificações iniciais.

A qualidade de um produto ou serviço está relacionada às necessidades de cada usuário e temos muitos fatores que podem influenciar de forma diferente, entre elas o tipo de produto ou serviço prestado e as necessidades de cada usuário. O que é qualidade para você, pode ser diferente para outra pessoa.

Você já pensou em quantos produtos ou serviços fazem parte do seu dia a dia? Quando você adquire esses produtos ou serviços, espera que tenham qualidade e não apresentem defeitos. É neste momento, que te convido a pensar sobre o significado da palavra qualidade e como ela está presente em nossas vidas.

Espero que tenha entendido a importância da qualidade e te convido a continuar seus estudos na próxima unidade!

Obrigado pela companhia.

Até mais!

LEITURA COMPLEMENTAR

Artigo: A importância da qualidade e testes no desenvolvimento de software

Autores: SALIM, Brandon Panace e ALVES, Gabriel Simões.

Link de acesso: <http://ric.cps.sp.gov.br/handle/123456789/9699>

Resenha: o artigo mostra como o desenvolvimento de software é um processo único onde cada novo projeto tem seu impacto diante do mercado alvo ou da empresa que o encomendou, a individualidade de cada projeto traz desafios únicos também, diante deste cenário a atuação da qualidade de software se mostra muito necessária, criando processos melhores para o desenvolvimento com base nos resultados obtidos do projeto atual e de projetos passados. Mostra a importância da atuação da qualidade de software, sendo utilizadas pesquisas bibliográficas para criar um maior entendimento sobre as metodologias de qualidade de software já existentes no mercado. E também foi desenvolvidos testes que demonstrem o funcionamento das metodologias e seu funcionamento em meio ao desenvolvimento.

Fonte: SALIM, Brandon Panace; ALVES, Gabriel Simões. A importância da qualidade e testes no desenvolvimento de software, 2022. Trabalho de conclusão de curso (Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) - Fatec Taubaté, Taubaté, 2022.

Artigo: Gestão da Qualidade no Desenvolvimento de Produto De Software

Autor: Jéssica Bruna da Costa Aguiar e Fernando Cesar Graciano

Link de acesso: <https://revista.fatectq.edu.br/interfacetecnologica/article/view/1314/737>

Resenha: artigo que fala sobre a exigência e a competitividade que estão se tornando um desafio para a produção de produtos de boa qualidade nos tempos atuais. Sendo assim a alta em criatividade e investimento faz com que seus produtores tenham que ter qualidade e eficiência. Para o processo de um produto de software existem requisitos básicos para uma boa qualidade, como por exemplo: bom desempenho, adaptável às necessidades específicas, fácil de usar e sem defeitos. O artigo tem como objetivo mostrar que com um bom gerenciamento de projeto com prazo e cronogramas desde o começo até o final da produção tende a ter qualidade, pois é fundamental que o software seja eficaz, confiável e siga as exigências.

MATERIAL COMPLEMENTAR

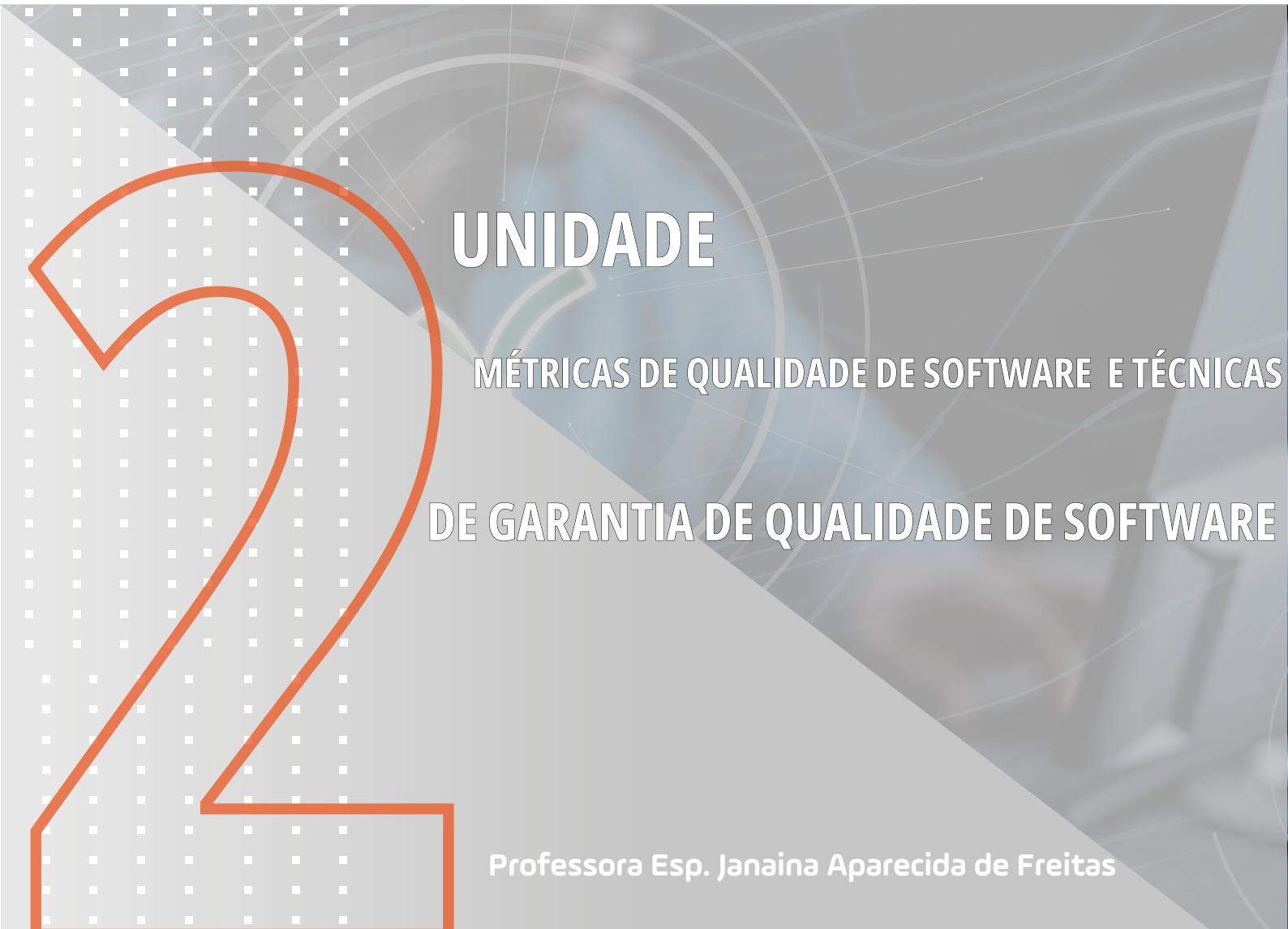


Livro: Qualidade de Software

Autores: Aline Zanin, Breno Cristóvão Rocha e Paulo A Pasqual Júnior

Editora: Grupo A

Sinopse: A qualidade é obtida a partir da sistematização de um processo de avaliação de conformidade em todas as etapas do desenvolvimento. Para alcançar a qualidade é necessário definir modelos, padrões, procedimentos, técnicas e métricas. E ainda,



UNIDADE

MÉTRICAS DE QUALIDADE DE SOFTWARE E TÉCNICAS

DE GARANTIA DE QUALIDADE DE SOFTWARE

Professora Esp. Janaina Aparecida de Freitas

Plano de Estudos

- Conceitos básicos sobre Métricas de Qualidade de Software e framework para Métricas de Produtos de Software.
- Métricas para o modelo de Requisitos e para o modelo projeto de software e Revisão de Software.
- Conceitos básicos da Garantia de Qualidade de Software.
- Elementos da Garantia da Qualidade de Software.

Objetivos da Aprendizagem

- Entender os conceitos básicos sobre as métricas de software e expor os frameworks para métricas de produto;
- Compreender as métricas para o modelo de requisitos e para o modelo projeto de software e entender os fundamentos da revisão de Software;
- Entender os conceitos básicos usados na garantia de qualidade de software;
- Conhecer os elementos trabalhados na garantia da qualidade de software.

INTRODUÇÃO

Olá aluno (a), seja bem-vindo (a) ao estudo sobre métricas de qualidade de software e técnicas de garantia de qualidade de software.

Nesta unidade iremos começar a entender a importância das métricas de software para a qualidade do produto. Quando usamos produtos ou serviços, espera-se que tenham qualidade e não apresentem defeitos. E podemos melhorar essa qualidade medindo o software. E por que medir? Para obter o controle do projeto e, com isso, poder gerenciá-lo. Medimos e avaliamos o software para estimar se estamos perto ou longe dos objetivos que foram definidos no projeto, como cronograma, custos, compatibilidade com os requisitos, qualidade etc.

Durante o ciclo de vida de um software são produzidos muitos elementos qualitativos. E as métricas podem ser usadas na análise, projeto, codificação e teste para serem conduzidas de forma mais objetiva e avaliadas de maneira mais quantitativa.

No primeiro tópico da nossa unidade vamos entender os conceitos básicos sobre as métricas de software e depois expor os frameworks para métricas de produto. As métricas de software ajudam a equipe de software a visualizar o projeto e seus artefatos criados, assim como na implementação do software, focando nos atributos específicos e no código para criar software com mais qualidade.

No segundo tópico vamos compreender as métricas para o modelo de requisitos e para o modelo projeto de software e entender os fundamentos da revisão de Software. Na fase de levantamento de requisitos, as métricas de produto proporcionam informações sobre a qualidade do modelo de análise e temos um modelo de requisitos que é onde os requisitos são formulados e onde se estabelece a base para o projeto de software. As métricas de projeto da arquitetura possuem foco na arquitetura do software e na eficácia dos módulos e componentes presentes dentro da arquitetura.

No terceiro tópico vamos entender os conceitos básicos usados na garantia de qualidade de software. A garantia e controle da qualidade são atividades essenciais para qualquer empresa que desenvolve produtos e com o tempo passando, o controle de qualidade tornou-se uma atividade importante de ser realizada e, muitas vezes, por outras pessoas e não por aquelas que constroem o produto.

No quarto e último tópico vamos conhecer os elementos trabalhados na garantia da qualidade de software. Ela possui várias preocupações e atividades que se concentram na

gestão da qualidade de software. Esses elementos são os padrões essenciais na construção do software e, normalmente, são construídos por organizações de padronização, como a ISO (Organização Internacional de Padronização) e o IEEE (Instituto de Engenheiros Eletrotécnicos e Eletrônicos).

Preparado (a) para continuar? Então, vamos seguir em frente.

Boa leitura e bons estudos!



CONCEITOS BÁSICOS SOBRE MÉTRICAS DE QUALIDADE DE SOFTWARE E FRAMEWORK PARA MÉTRICAS DE PRODUTOS DE SOFTWARE



Para prover melhorias em um produto de software é preciso entender sobre métricas e medição. E uma métrica de software é algo que poderá ser medido no software que foi desenvolvido, como o tamanho de um sistema em linhas de códigos, o número de defeitos encontrados durante o desenvolvimento ou durante o seu uso pelo cliente, entre outros (SOMMERVILLE, 2018).

Para Pressman e Maxim (2016, p. 653), “um elemento-chave de qualquer processo de engenharia é a medição. Você pode usar medidas para melhorar o entendimento dos atributos dos modelos criados e para avaliar a qualidade dos produtos ou sistemas construídos”. As métricas auxiliam os engenheiros de software a visualizar o projeto e seus artefatos criados, assim como na implementação do software, focando nos atributos específicos e no código, para criar software com mais qualidade.

Você deve estar pensando: Por que precisamos medir e avaliar o software? Medimos ele para obter controle do projeto e, portanto, poder gerenciá-lo. Medimos e avaliamos o software para estimar se estamos perto ou longe dos objetivos que foram definidos no projeto, como cronograma, custos, compatibilidade com os requisitos, qualidade etc.

Durante a criação de um software, sempre haverá elementos qualitativos. E as métricas proporcionam uma base por meio da qual análise, projeto, codificação e teste podem ser conduzidos de forma mais objetiva e avaliados de maneira mais quantitativa

O primeiro passo no processo de medição é derivar as medições de software e as métricas adequadas para a representação do software. Depois, os dados necessários são coletados para derivar as métricas formuladas. Uma vez calculadas, as métricas apropriadas são analisadas com base em

diretrizes preestabelecidas e dados do passado. Os resultados das análises são interpretados para obter informações sobre a qualidade do software e os dados da interpretação levam à modificação dos requisitos e modelos de projeto, código-fonte ou casos de teste. Em alguns casos, pode também levar à modificação do próprio processo de software (PRESSMAN; MAXIM, 2016).

Para você entender melhor sobre métricas, vou apresentar alguns conceitos que serão fundamentais para se entender o conteúdo a ser tratado. No contexto da Engenharia de Software, poderemos encontrar os termos medidas, métricas e indicadores:

FIGURA 1 - DESCRIÇÃO DOS TERMOS.

MEDIDA: proporciona uma indicação de **quantidade, capacidade ou tamanho de algum atributo ou processo**

MÉTRICA: está relacionada a medidas individuais, como o número de erros encontrados em um teste de unidade

INDICADOR: é uma métrica, ou mesmo uma combinação de métricas, que possuem informações sobre um processo do software

Fonte: a autora.

Segundo Sommerville (2018), a medição tem um objetivo a longo prazo que é o de ser usada para revisões e fazer julgamento sobre a qualidade de software. Para o autor, ao usar a medição, o sistema poderá ser parcialmente avaliado e com isso podemos deduzir um valor para a qualidade do software, onde: se ele atingir o valor estipulado, é aprovado. Outro ponto é que a medição pode ser usada para realçar partes do software que podem ser melhoradas.

Temos algumas métricas que auxiliam em todo o processo de desenvolvimento do software, como as métricas de controle e métricas de previsão.

FIGURA 2 - TIPOS DE MÉTRICAS

Métrica de Controle

- **Supora processos relacionados ao gerenciamento, ao processo de software em si.**
- **Exemplo: tempo necessário para reparar os defeitos que foram encontrados.**

Métrica de Previsão

- **Ajuda a prever as características do software e são associadas com o software em si.**
- **Conhecidas como Métricas de Produto.**
- **Exemplo: o número de atributos e operações de uma classe de objeto.**

Fonte: a autora.

As medidas são utilizadas como indicadores de qualidade independentes dos modelos de requisitos e projeto ou estágios do processo de software, elas sempre fornecem algo consistente e objetivo para ser usado na avaliação da qualidade do software.

É importante que você entenda que as métricas só serão úteis se forem verificadas e validadas, e que demonstram ser importantes para o processo de desenvolvimento do software. Assim, é indispensável que uma métrica tenha propriedades matemáticas, com intervalos numéricos de avaliação, por exemplo. Pois para cada métrica ser analisada e validada precisam ter um valor real de interesse, independentemente de qualquer outro fator que possa vir a surgir depois.

Existem diversas métricas que já foram elaboradas e aplicadas em software de computadores, mas nem todas possuem suporte para que o engenheiro de software consiga aplicá-las. De modo geral, há um conjunto de atributos que devem ser abrangidos por métricas, de acordo com o quadro a seguir:

QUADRO 1 - CONJUNTO DE ATRIBUTOS QUE DEVEM SER ABRANGIDOS POR MÉTRICAS.

Atributos	Descrição
Simples e computáveis	As métricas devem ser relativamente fáceis de aprender e derivar, sua computação não deve demandar esforço ou tempo fora do normal.
Empiricamente e intuitivamente persuasiva.	A métrica deve satisfazer as ideias do engenheiro de software sobre o atributo do produto considerado (por exemplo, uma métrica que mede coesão de módulo deverá crescer em valor na medida em que aumenta o nível de coesão).
Consistente e objetiva.	A métrica deverá sempre produzir resultados que não sejam ambíguos. E quem usar ela, deverá ser capaz de derivar o mesmo valor da métrica usando as mesmas informações sobre o software.
Consistente no seu uso das unidades e dimensões.	A computação matemática da métrica deverá usar medidas que não resultem em combinações bizarras de unidades. Por exemplo, multiplicar o número de pessoas nas equipes de projeto pelas variáveis da linguagem de programação no programa resulta em uma mistura duvidosa de unidades que não é claramente convincente.
Independente da linguagem de programação.	As métricas devem ser baseadas no modelo de requisitos, modelo de projeto ou na própria estrutura do programa. Elas não deverão ser dependentes dos caprichos da sintaxe ou semânticas das linguagens de programação.
Um mecanismo efetivo para feedback de alta qualidade.	A métrica deverá fornecer informações que podem levar a um produto final de melhor qualidade.

Fonte: Adaptado de Pressman e Maxim (2016, p. 656).

Muitas métricas utilizadas satisfazem os atributos citados, mas uma ou outra podem não satisfazer, e isso não é um sinal para a rejeição dela.

As métricas fornecem muitas informações úteis e com valores distintos, mesmo que não satisfaça um atributo perfeitamente (PRESSMAN; MAXIM, 2016).

As métricas de produto quantificam atributos internos do software, como por exemplo, tamanho, acoplamento entre componentes, coesão de um componente etc. E são divididas em dois tipos de métricas:

- **Métricas Dinâmicas:** são as métricas coletadas por medições realizadas durante a execução do programa. Exemplo: tempo de execução. As métricas dinâmicas ajudam a avaliar os atributos de qualidade como a eficiência e a confiabilidade e elas são medidas após o sistema ter sido implementado.
- **Métricas Estáticas:** são as métricas que são coletadas por medições realizadas na documentação de projeto ou código fonte do programa. Exemplo: linhas de código. As métricas estáticas ajudam a avaliar atributos como a complexidade do produto e a facilidade de manutenção e podem ser medidas na fase de projeto.

A seguir, um quadro com algumas métricas estáticas:

QUADRO 2 – ALGUMAS MÉTRICAS ESTÁTICAS.

Fan-in / Fan-out	Fan-in conta o número de funções que chamam uma determinada função, onde o valor alto significa grande impacto em mudanças (propagação) Fan-out conta o número de funções chamadas pela função, onde o valor alto significa grande complexidade da função.
Tamanho do código	É uma das métricas que tem se mostrado como as mais confiáveis e úteis. Visto que em geral, quanto maior o componente, mais complexo e propenso a erros ele será.
Complexidade Ciclomática	Métrica que mede a complexidade de controle do programa (if, while, for etc.), e está relacionada à facilidade de compreensão.
Tamanho do Vocabulário	Métrica que conta a quantidade de identificadores (exemplo, nome de classes) do programa. Sendo que muitos identificadores podem indicar que eles são mais significativos.
Profundidade de Aninhamento	Métrica que conta as estruturas internas como if e while aninhados. Sendo que as estruturas aninhadas são mais difíceis de se compreender.

Fonte: Adaptado de Pressman e Maxim (2016, p. 656).

Vamos conhecer um framework que está sendo muito utilizado e que foi desenvolvido pela Google e pela Digital Telepathy chamado HEART. Ele utiliza a métrica de experiência com o usuário, ou seja, fornece várias métricas que são centradas no usuário e que tem a finalidade de mediar a experiência do usuário com o software. E para o sucesso do software é importante entender que o usuário é a parte mais importante do processo e que entendendo ele, poderá trazer benefícios.

As siglas da palavra HEART significam: Happiness, Engagement, Adoption, Retention e Task Success. Muitas empresas sabem que para o negócio dar certo e ter sucesso, é necessário levar em consideração o cliente como sendo o coração do negócio e, por isso, esse framework recebe esse nome.

Essas cinco palavras formam cinco itens fundamentais que resultam numa métrica que auxiliam a equipe a verificar o desempenho do sistema, o crescimento e o sucesso.

- **Happiness:** significa felicidade. Métrica usada para medir a satisfação do usuário em relação ao software que está sendo utilizado, ou seja, informa o quanto ele gosta ou não do produto. Alguns exemplos de métricas: analisar como os usuários se sentem, a satisfação, a percepção de facilidade do uso do sistema etc..

- **Engagement:** significa engajamento. Essa métrica mede o engajamento do usuário em utilizar e interagir com o software. Mede-se realmente se o usuário está utilizando ou não as funções do software, qual a frequência que os usuários voltam a utilizar o seu sistema, analisar o número de visitas de usuários na semana, número de ações realizadas no sistema, número de relatórios gerados etc.

- **Adoption:** significa adoção. Verifica o uso de novas funcionalidades criadas no sistema e quantos novos usuários começaram a utilizar a função criada. Verificar também o quanto de usuários antigos estão aceitando o sistema, e o porquê de alguns não estarem aceitando, verificar quantos usuários concluem suas tarefas e como é a regularidade da execução destas tarefas.

- **Retention:** significa retenção. Métrica que verifica o comportamento de todos os usuários do sistema frente às mudanças que sempre acontecem no software, verificar qual a quantidade de usuários que voltaram a utilizar o sistema, verificar o número de usuários do sistema, quantos deixaram de utilizar etc.

- **Task Success:** significa sucesso da tarefa. Métrica auxilia na verificação de quanto é difícil um usuário realizar e terminar uma determinada tarefa no sistema. Verifica se a solução encontrada para determinada função, está de acordo com a necessidade do usuário, verifica o sucesso de resultados obtidos, ou métricas relacionadas a erros existentes ao finalizar uma ação no sistema.

A seguir temos um quadro que mostra um exemplo de HEART:

QUADRO 3 - EXEMPLO DO FRAMEWORK HEART

	Objetivos	Sinais	Métricas
Happiness	Fazer os usuários acharem o sistema útil e fácil de usar	Respondendo pesquisas. Avaliando o serviço Deixando feedbacks.	Ranking de satisfação dos usuários. Número de avaliações realizadas.
Engagement	Fazer os usuários continuam a executar e utilizar as atividades	Gastando tempo no sistema.	Tempo de utilização Frequência média de sessões ou telas abertas.
Adoption	Fazer com que novos usuários utilizem o sistema	Criando novos usuários Usando features	Número de registros Adoção de features
Retention	Fazer com que usuários voltem para utilizar o software	Ativos no aplicativo	Taxa de erros
Task Success	Fazer com que os usuários completem os objetivos de forma rápida e fácil	Completando tarefas	Número de quedas

Fonte: a autora.

Analizando o exemplo do HEART é possível observar que as métricas são importantes, úteis e que facilitam a qualidade do software e a qualidade de experiência com o usuário.

MÉTRICAS PARA O MODELO DE REQUISITOS E PARA O MODELO PROJETO DE SOFTWARE E REVISÃO DE SOFTWARE

Podemos trabalhar com as métricas em todas as fases do ciclo de desenvolvimento do software. Algumas atingem níveis fáceis de usar e compreender e outras de níveis mais difíceis, dependendo de como são aplicadas. Na fase de levantamento de requisitos, as métricas de produto proporcionam informações sobre a qualidade do modelo de análise (PRESSMAN; MAXIM, 2016).

O modelo de requisitos é onde os requisitos são formulados e onde se estabelece a base para o projeto de software.

Uma das métricas usadas é a baseada em função. Ela é utilizada para medir a funcionalidade que o sistema irá fornecer, podendo ser empregadas para estimar um custo ou trabalho, para projetar, para codificar e testar um determinado software. Elas ajudam a prever o número de erros que podem ser encontrados durante um teste e a prever o número de componentes ou linhas projetadas do sistema (PRESSMAN; MAXIM, 2016).

Outra métrica usada é por Pontos de função, que são derivadas de medidas que são calculadas por valores de domínio de informações, que são definidos em diversas formas, como mostradas nos quadros abaixo:

QUADRO 4 – MÉTRICAS PARA O MODELO DE REQUISITOS.

Número de entradas externas (EEs): que são originadas de um usuário transmitidas de outra aplicação.

Número de saídas externas (EOs): onde cada saída é formada por dados de uma aplicação que fornece informações para usuários.

Número de consultas externas (EQs): são consultas externas definidas como entradas que resultam em respostas imediatas de um software

Número de arquivos de interfaces externos (EIFs): onde cada arquivo de interface reside fora da aplicação, mas fornece informações que são usadas na aplicação.

Número de arquivos lógicos internos (ILFs): cada arquivo lógico de dados de um aplicativo é mantido através de entradas externas

Fonte: Adaptado de Pressman e Maxim (2016, p. 659).

Uma vez entendidos os dados, podemos utilizar a fórmula FP para calcular o ponto de função:

$$FP = \text{contagem total} \times [0,65 + 0,01 \times \sum (Fi)]$$

Onde temos: os Fi que são os fatores de ajustes de valores que são baseados em respostas para algumas questões.

A seguir, uma imagem que exemplifica como computar os pontos de função.

FIGURA 3 - COMPUTANDO PONTOS DE FUNÇÃO.

Valor do Domínio de Informação	Contagem	Fator de peso			
		Simples	Médio	Complexo	
Entradas externas (EIs)	3	×	3	4	6 = 9
Saídas Externas (EOs)	2	×	4	5	7 = 8
Consultas Externas (EQs)	2	×	3	4	6 = 6
Arquivos Lógicos Internos (ILFs)	1	×	7	10	15 = 7
Arquivos de Interface Externos (EIFs)	4	×	5	7	10 = 20
Contagem total					50

Fonte: Pressman e Maxim (2016, p. 662).

Temos as métricas para qualidade de especificação, onde são determinadas características para serem usadas para avaliar a qualidade do modelo de requisitos, como: totalidade a peculiaridade (ausência de ambiguidade), totalidade, exatidão, entendimento, repetibilidade, consistência interna e externa, disponibilidade, brevidade, rastreabilidade, modificação, precisão e reutilização (PRESSMAN; MAXIM, 2016).

Agora vamos discutir sobre as Métricas para o Modelo de Projetos. Para a construção de um projeto de software é importante que se tenha métricas ou padrões que proporcionam melhor visualização e podem ajudar a evolução do projeto com mais qualidade.

A Métrica de projeto da arquitetura possui foco na arquitetura do software e na eficácia dos módulos e componentes presentes dentro da arquitetura. Para Pressman e Maxim (2016), essas métricas são uma “caixa-preta”, pois não exigem qualquer conhecimento do funcionamento interno de determinado módulo ou componente de software.

Ela define três medidas de complexidade de projeto de software: complexidade estrutural, complexidade de dados e complexidade de sistemas. À medida que esses valores de complexidade aumentam, a complexidade global da arquitetura do sistema também aumenta. Isso leva a uma maior probabilidade de que o trabalho de integração e teste também aumente (PRESSMAN; MAXIM, 2016, p. 664)

A métrica para projeto orientado a objeto é importante, pois um projetista experiente difere de um novato, ele “sabe” como caracterizar um sistema de modo que implemente os requisitos do cliente. E a medida que aumenta o tamanho e a complexidade destes sistemas temos uma visão mais objetiva das características do projeto, que pode favorecer tanto o projetista experiente (que obtém uma visão adicional) quanto o novato (que obtém uma indicação da qualidade que, de outra forma, não estaria disponível) (PRESSMAN; MAXIM, 2016).

E devido a esse motivo, há características importantes que um projeto orientado a objetos deve ter e que são citados no quadro a seguir:

QUADRO 5 - CARACTERÍSTICAS IMPORTANTES DE UM PROJETO ORIENTADO A OBJETOS.

Características	Descrição
Tamanho	Definida em termos de população, volume, comprimento e funcionalidade. Em população, conta-se classes ou operações; que também são medidas em relação ao volume. Em comprimento, é medida a cadeia de elementos de projetos interconectados. Em funcionalidade, é o valor fornecido ao cliente para a aplicação.
Complexidade	Medida em relação a características estruturais, verificando como as classes se comportam e se relacionam com outras classes.
Acoplamento	Onde são analisadas as conexões, ou seja, as colaborações entre classes e mensagens trocadas entre os objetos.
Totalidade	Considera vários pontos de vista sobre algo, se há uma implicação indireta no grau com que a abstração ou componente pode ser reutilizado.
Suficiência	Analisa se um componente do projeto é suficiente para expor todas as propriedades do objeto de domínio da aplicação.
Coesão	Analisa se todas as operações estão funcionando em conjunto, que foram projetadas de maneira correta, a fim de atingir o objetivo final.
Coerência	Analisa se faz parte do domínio do problema ou projeto.
Similaridade	Analisa o grau de similaridade entre duas ou mais classes, em relação à estrutura, função, comportamento ou finalidade.
Originalidade	Analisa se uma operação não pode ser construída por meio de uma sequência de outras operações que estão contidas em uma classe. Uma classe, por exemplo, é tida como original se suas operações forem primitivas.
Volatilidade	Medido a possibilidade de que uma alteração venha a ocorrer, já que em um projeto pode ocorrer diversas alterações durante seu ciclo.

Fonte: Adaptado de Pressman e Maxim (2016, p. 668).

As métricas de software do tipo orientado a objetos podem ser aplicadas para o modelo de projeto e também para o modelo de requisitos.

Agora vamos falar sobre a Revisão de Software. De acordo com Pressman e Maxim (2016, p. 431), as revisões de software são “como um filtro” para a gestão de qualidade. Isso significa que elas são aplicadas em várias etapas durante o processo de engenharia de software e servem para revelar erros e defeitos que podem ser eliminados”. Os autores ainda comentam que as revisões de software ajudam a purificar os artefatos que são criados durante o desenvolvimento de um software, desde os modelos de requisitos e de projeto, dados de teste e código.

O principal objetivo das revisões técnicas é encontrar erros durante o processo, para que não se tornem defeitos depois da liberação do software. A vantagem evidente das revisões técnicas é a descoberta precoce de erros, a fim de que não sejam propagados para a próxima etapa na gestão de qualidade. (PRESSMAN; MAXIM, 2016, p. 433).

Revisar um software é algo importante a se realizar já que estamos analisando criticamente algo. O processo de revisão pode ser estabelecido como uma avaliação crítica de um software, de um processo, de um objeto ou classe. As revisões são consideradas o mecanismo mais eficaz para descobrir erros e defeitos logo no início do gerenciamento da qualidade (PRESSMAN; MAXIM, 2016).

Assim, visam a qualidade de um processo ou de um produto e são considerados como filtros de problemas e inconsistências que podem estar presentes em todo o processo de programação e desenvolvimento do software.

As revisões de software que podemos realizar são: Inspeções, Walkthroughs (passo a passo) e Pair programming (programação em pares).

- **Inspeções:** é um tipo de revisão aplicada a todos os artefatos presentes e que analisa praticamente tudo, sendo considerado rigoroso na sua inspeção, onde devem ser analisados erros, defeitos e falhas possíveis de acontecer. O objetivo da inspeção é melhorar a qualidade dos componentes do software, removendo o que há erros e pensando nas próximas fases de implementação, resultando em ganhos ao final de cada projeto.
- **Walkthroughs (passo a passo):** é como uma reunião entre a equipe com a finalidade de analisar a qualidade do produto em relação ao desenvolvimento do software. Ajuda na prevenção de possíveis erros e cada membro da equipe uma saberá o que deve ser feito.
- **Pair programming (programação em pares):** o código fonte é compartilhado entre dois programadores e ambos trabalham programando e analisando o código criado, ao mesmo tempo e no mesmo computador. Os programadores se intercalam nas ações (um programador codifica e outro analisa o trabalho e, depois, realizam a troca de papéis), analisam as sintaxes, métodos e tudo que pode ser otimizado e simplificado.

Quando se realiza uma revisão de softwares, deve-se planejar o que fazer, o que deve ser revisado e quais serão os resultados esperados. Pode-se usar checklists para a organização e execução das análises, para que tudo seja controlado e, com isso, gerar documentos que informem o que foi revisado e alterado.

CONCEITOS BÁSICOS DA GARANTIA DE QUALIDADE DE SOFTWARE



É muito comum imaginar que é possível trocar o prazo e as vezes, o custo, por qualidade. E em muitos casos, é possível reduzir prazos e custos através da redução dos requisitos de um software. E quanto à qualidade? Ela é consequência de muitos processos e pessoas envolvidas nestes processos e das tecnologias usadas. A qualidade do produto é complexa quando pensamos nesses fatores, e por isso é mais difícil controlar o nível de qualidade do produto do que controlar os requisitos deste produto.

Para Pressman e Maxim (2016), a garantia da qualidade e o controle da qualidade são atividades essenciais para qualquer empresa que desenvolve produtos. Conforme o tempo foi passando, muitas técnicas de produção em massa tornaram-se comuns, e o controle de qualidade tornou-se uma atividade importante de ser realizada, muitas vezes por outras pessoas e não por aquelas que constroem o produto.

De acordo com ZANIN et al (2018), a garantia da qualidade de software (Quality Assurance) são as atividades que dão suporte aos processos estabelecidos de forma contínua e que visam fornecer confiabilidade a esses processos. Eles estão em contínua revisão, melhoria e adaptação para desenvolver softwares que atendam às necessidades e requisitos estipulados pelo cliente.

SGA - Software Quality Assurance (garantia de qualidade de software) é o nome dado ao processo para garantir a qualidade de um software e que possui uma série planejada de atividades de apoio que auxiliam na confiança ao software.

A garantia da qualidade de software (SQA) conforme Pressman e Maxim (2016, p. 449) abrange:

1. Um processo de SQA;
2. Tarefas específicas de garantia e controle da qualidade (revisões técnicas e uma estratégia de testes multicamadas);
3. Uma prática efetiva de engenharia de software (métodos e ferramentas);
4. O controle de todos os artefatos de software e as mudanças feitas nesses produtos;
5. Um procedimento para garantir a conformidade com os padrões de desenvolvimento de software (quando aplicáveis);
6. Mecanismos de medição e de geração de relatórios.

A garantia de qualidade de software está associada com todas as atividades que formam o ciclo de desenvolvimento de um software, desde o primeiro contato com o cliente até a programação do software em si. Dessa forma, a área de garantia da qualidade se preocupa com a verificação e a validação do software, a verificação quanto aos processos e às definições (garantia da qualidade de processos) e a validação quanto ao produto (garantia da qualidade de produto). Para esse fim, diversas são as técnicas, os modelos e as atividades empregadas (ZANIN et al, 2018, p.21).

O objetivo da garantia da qualidade de software é fazer com que o sistema seja entregue ao cliente com qualidade e com valor de negócio, e que o processo de verificação dessa qualidade seja conduzido de forma organizada e bem documentada.

É considerado um bom processo de garantia de qualidade quando existe uma relação, onde realiza-se uma etapa de garantia da qualidade de software para cada etapa de desenvolvimento.

Há diversos modelos que podem ser utilizados para a garantia da qualidade. Eles podem ser usados para a garantia da qualidade de processos e de produtos para a verificação e validação.

A garantia da qualidade tem alguns papéis fundamentais que devem ser considerados:

- Ajuda a estabelecer processos;
- Determina programas de medida para avaliar processos;
- Procura identificar as fraquezas de medida para avaliar processos;
- É uma responsabilidade de gerenciamento;
- Está relacionada com todos os produtos que serão gerados por um processo;
- Avalia se o controle de qualidade está funcionando.

O processo de garantia da qualidade considera uma análise mais detalhada para a qualidade do processo e do produto. E neste processo, podemos usar as métricas para quantificar a sua qualidade e para os produtos, podemos usar as técnicas de verificação e validação (ZANIN et al, 2018).



ELEMENTOS DA GARANTIA DA QUALIDADE DE SOFTWARE

A garantia da qualidade de software possui várias preocupações e atividades que se concentram na gestão da qualidade de software.

O quadro a seguir lista algumas dessas preocupações e atividades:

QUADRO 6 – ALGUMAS PREOCUPAÇÕES E ATIVIDADES PARA A GARANTIA DA QUALIDADE

Padrões	O IEEE, a ISO e outras organizações de padronização produzem uma ampla lista de padrões para engenharia de software e documentos relacionados à área. Esses padrões podem ser adotados pela empresa de desenvolvimento ou impostos pelo cliente ou outros envolvidos. A garantia da qualidade tem o papel de garantir que os padrões adotados sejam seguidos e que todos os produtos resultantes estejam em conformidade.
Revisões e auditorias	As revisões técnicas são uma atividade de controle de qualidade e seu objetivo é revelar erros. As auditorias são um tipo de revisão que tem o intuito de assegurar que as diretrizes de qualidade estejam sendo seguidas no trabalho de engenharia de software.
Testes	O teste de software tem a função de controle de qualidade com um objetivo principal: encontrar erros. O papel da garantia da qualidade é garantir que os testes sejam planejados e conduzidos de modo que se tenha a maior probabilidade possível de alcançar seu objetivo primário.
Coleta e análise de erros/defeitos	A coleta e a análise são formas de melhorar e medir o desempenho. A garantia da qualidade reúne e analisa dados de erros e defeitos para melhor compreender como eles são introduzidos e quais atividades são as mais adequadas para sua eliminação.
Gerenciamento de mudanças	As mudanças ocorrem em qualquer projeto de software. A garantia da qualidade garante que práticas adequadas de gerenciamento de mudanças sejam utilizadas.
Educação	A garantia da qualidade assume a liderança no processo de aperfeiçoamento do software e é um patrocinador de programas educacionais dos engenheiros de software.
Gerência dos fornecedores	O papel da garantia da qualidade é garantir software de alta qualidade por meio da sugestão que o fornecedor deve seguir, e incorporar exigências de qualidade como parte de qualquer contrato com um fornecedor externo.
Administração da segurança	A garantia da qualidade garante o emprego de processos e tecnologias apropriados para se ter a segurança de software desejada.
Proteção	A garantia da qualidade pode avaliar o impacto de falhas de software e por iniciar as etapas necessárias para redução de riscos.
Gestão de riscos	Análise e a redução de riscos são preocupações dos engenheiros de software, e a garantia da qualidade garante que as atividades de gestão de riscos sejam conduzidas apropriadamente e que planos de contingência relacionados a riscos tenham sido estabelecidos.

Fonte: Adaptado de Pressman e Maxim (2016, p. 450).

Além de cada uma dessas preocupações e atividades, a garantia da qualidade trabalha para garantir que atividades de suporte ao software, como: manutenção, suporte on-line, documentação e manuais sejam realizadas ou produzidas pensando na qualidade.

E ainda temos que pensar na qualidade do projeto e do código. Na qualidade do projeto, a garantia da dela deve fazer com que os requisitos e o próprio projeto estejam de acordo com o que foi solicitado. Na qualidade do código, a garantia da qualidade deve fazer com que os códigos estejam em conformidade com os padrões de codificação, o que irá facilitar a manutenção futura.



Um processo de modelagem e análise é efetuado como parte de proteção do software. Inicialmente, os problemas são identificados e classificados por criticalidade e risco. Por exemplo, problemas associados a um controle computadorizado de um automóvel podem: (1) provocar uma aceleração descontrolada que não pode ser interrompida, (2) não responder ao acionamento do pedal do freio (através de uma desativação), (3) não operar quando a chave é ativada e (4) perder ou ganhar velocidade lentamente. Uma vez identificados esses perigos no nível de sistema, técnicas de análise são utilizadas para atribuir gravidade e probabilidade de ocorrência. Para ser efetivo, o software deve ser analisado no contexto de todo o sistema.

Pressman, Roger, S. e Bruce R. Maxim. Engenharia de software. (9^a edição). Grupo A, 2021.



Garantia de qualidade e controle de qualidade são áreas diferentes da engenharia de software. A área de controle de qualidade age diretamente sobre o produto e realiza a execução dos processos, enquanto a área de garantia da qualidade estrutura tanto o controle de processos quanto o controle de produtos e garante que estes estejam sendo executados.

Fonte: ZANIN, Aline; JÚNIOR, Paulo A P.; ROCHA, Breno C.; et al. Qualidade de software. Grupo A, 2018.

CONSIDERAÇÕES FINAIS

Chegamos ao final da nossa segunda unidade! Espero que você tenha aproveitado ao máximo os conhecimentos apresentados nesta etapa, onde aprendemos sobre métricas de qualidade de software e técnicas de garantia de qualidade de software.

Foram apresentados a importância das métricas de software para a qualidade do produto. Aprendemos que para obter o controle do projeto precisamos medir e avaliar o software, para que possamos gerenciá-lo. Aprendemos também que medimos e avaliamos o software para estimar se estamos no caminho e objetivos definidos no projeto e que as métricas podem ser usadas na análise, projeto, codificação e teste para serem conduzidas de forma mais objetiva e avaliados de maneira mais quantitativa, para criar software com mais qualidade.

Compreendemos ao longo da unidade o que são as métricas para o modelo de requisitos e para o modelo projeto de software e passamos a entender os fundamentos da revisão de Software. Entendemos que a garantia da qualidade e o controle da qualidade são atividades essenciais para qualquer empresa que desenvolve produtos sobre a garantia de qualidade de software.

Espero que tenha entendido as métricas de qualidade de software e técnicas de garantia de qualidade de software e te convido a continuar seus estudos na próxima unidade!

Obrigado pela companhia.

Até mais!

LEITURA COMPLEMENTAR

Artigo: Uma Análise de Métricas de Software Orientados à função e sua Aplicação ao Desenvolvimento Orientado a Objetos

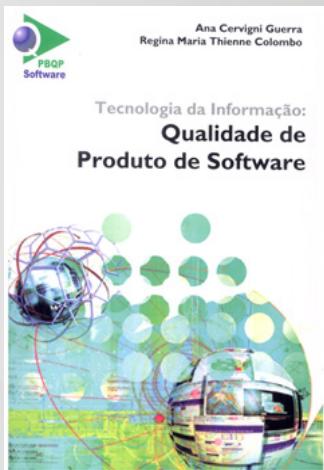
Autores: Everton Alves Miranda

Link de acesso: https://www.researchgate.net/publication/272906572_Uma_analise_de_metricas_de_software_orientadas_a_funcao_e_sua_aplicacao_no_desenvolvimento_orientado_a_objetos

Resenha: o artigo visa mostrar uma abordagem da mensuração de softwares através da metodologia de Pontos por Função, partindo de aspectos históricos, demonstrando seu funcionamento e analisando sua possível aplicação em softwares desenvolvidos dentro do paradigma da Orientação a Objetos.

Fonte: Miranda, Everton. (2002). Uma análise de métricas de software orientadas à função e sua aplicação ao desenvolvimento orientado a objetos. Revista Vértices. 4. 37-41. 10.5935/1809-2667.20020007.

MATERIAL COMPLEMENTAR



Título: Tecnologia da Informação: Qualidade de Produto de Software

Autor: Ana Cervigni Guerra; Regina Maria Thienne Colombo

Editora: PBPQ Software

Sinopse: O livro trata de assuntos relacionados à Engenharia de Software, em especial sobre a qualidade de software. Apresenta conceitos fundamentais sobre a qualidade de software, categorias de produtos de software e formas de avaliação, modelos de qualidade, requisitos e avaliação da qualidade de software, metodologia do processo de avaliação e como é realizado a avaliação de um software. Estes conceitos são importantes para quem pretende entrar no mercado de trabalho.



FILME/VÍDEO

Título: A Teoria de tudo

Ano: 2014

Sinopse: Baseado na biografia de Stephen Hawking, o filme mostra como o jovem astrofísico (Eddie Redmayne) fez descobertas importantes sobre o tempo, além de retratar o seu romance com a aluna de Cambridge Jane Wide (Felicity Jones) e a descoberta de uma doença motora degenerativa quando tinha apenas 21 anos.

UNIDADE INTRODUÇÃO AO TESTE DE SOFTWARE

Professora Esp. Janaina Aparecida de Freitas

Plano de Estudos

- Conceitos básicos sobre Teste de Software.
- Ciclo de Vida do Teste de Software.
- Técnicas de Teste de Software.
- Ferramentas para automatização de Teste de Software.

Objetivos da Aprendizagem

- Entender os conceitos básicos sobre Teste de software e seus objetivos;
- Compreender o processo do Ciclo de Vida do Teste de Software;
- Entender as Técnicas de Teste de Software que podem ser aplicadas;
- Conhecer as Ferramentas para automatizar os Teste de Software.

INTRODUÇÃO

Olá aluno (a), seja bem-vindo (a) ao estudo sobre teste de software.

Nesta unidade vamos começar a entender a importância de se realizar os testes e entender o porquê ele é importante e essencial para garantir a qualidade do software e ter um produto com o mínimo de erros possível. Para realizarmos esta tarefa, faz-se necessário o uso de ferramentas que auxiliem no processo de teste de software.

Imagino que você tenha acesso a diferentes softwares diariamente, por exemplo, aplicações bancárias ou aplicativos, sistemas usados nas empresas em que trabalha. Você se sente seguro em realizar uma transação bancária pela web ou pelo aplicativo? Com certeza sim, apesar que às vezes podem ocorrer algumas falhas. Esses exemplos mostram que sistemas que não funcionam corretamente e que não foram testados podem trazer diversos prejuízos às empresas e ter grandes impactos para as pessoas que os utilizam.

No primeiro tópico da nossa unidade vamos entender os conceitos básicos sobre os testes de software e seus objetivos. O teste é considerado uma atividade de verificação e validação do sistema onde é realizada uma análise dinâmica na execução, com o objetivo de verificar a presença de erros e aumentar a confiança de que o mesmo está de acordo com o que o cliente requisitou.

No segundo tópico vamos compreender o processo do Ciclo de Vida do Teste de Software, pois quanto antes os testes iniciarem, mais barato será corrigir os erros encontrados.

No terceiro tópico vamos entender as Técnicas de Teste de Software que podem ser aplicadas. Elas são procedimentos técnicos e gerenciais que ajudam na avaliação e melhoria do processo. Ao realizar testes, eles devem refletir as ações tomadas para descobrir erros introduzidos na codificação das funcionalidades definidas nas especificações dos programas. E temos uma grande variedade de técnicas de teste, aplicáveis a diferentes circunstâncias, podendo ser utilizadas para classificar diferentes conceitos de testes, técnicas de design de situação de testes, técnicas de execução de teste e organizações de testes.

No quarto e último tópico vamos conhecer as ferramentas para automatizar os Testes de Software. Ao automatizar um teste, colocamos a máquina testar, ou seja, escrevemos um programa que teste o sistema de forma automática e uma máquina, com certeza executaria o teste muito mais rápido do testador. Um teste automatizado é muito parecido com um teste manual, só que é a máquina, através de um script que fará os testes.

Preparado (a) para continuar? Então, vamos seguir em frente.

Boa leitura e bons estudos!

CONCEITOS BÁSICOS SOBRE TESTE DE SOFTWARE

TEST

Hoje temos um aumento pela necessidade de melhorias no software, percebida pelas próprias empresas, seja por exigência do mercado ou pelos clientes. E uma das formas das empresas melhorarem a qualidade do software por elas desenvolvido é através da implantação de equipes de testes com o objetivo de testar os sistemas produzidos e atingir um nível de qualidade aceitável em um espaço de tempo cada vez menor.

O teste de software é considerado uma atividade de verificação e validação do software que foi desenvolvido, onde é realizada uma análise dinâmica na execução dele, com o objetivo de verificar a presença de defeitos ou falhas e aumentar a confiança de que ele está correto e de acordo com o que o cliente solicitou.

No processo de desenvolvimento do software, o teste é a última fase e é de suma importância, pois por meio dela é possível ser verificado e validado o sistema e caso sejam encontrados defeitos e falhas, estes podem ser resolvidos. O objetivo do teste de software é encontrar erros e produzir softwares com qualidade e que sejam seguros e confiáveis para uso (GONÇALVES; BARRETO; ZENKER, 2019).

Para Sommerville (2018, p. 203) “o teste é destinado a mostrar que um programa faz o que é proposto a fazer e para descobrir os defeitos do programa antes do uso”. Quando o software é testado, ele é executado, onde são incluídos dados fictícios, ou seja, é simulado como se o usuário estivesse usando o sistema. A partir dos resultados do teste, é verificado possíveis defeitos e falhas, ou anomalias ou se algum atributo não está funcionando adequadamente.

Para Pressman e Maxim (2016, 467) o teste de software “exige mais trabalho de projeto do que qualquer outra ação da engenharia de software. Se for feito casualmente, perde-se tempo, fazem-se esforços desnecessários e, ainda pior, erros passam sem ser detectados”. O sensato é estabelecer uma estratégia sistemática para teste de software.

O teste de software possui dois objetivos principais: (I) eles devem encontrar defeitos no software, para que possam ser corrigidos ou minimizados e (II) fornecem uma avaliação geral da qualidade do produto e uma estimativa das possíveis falhas encontradas (GONÇALVES; BARRETO; ZENKER, 2019).

Os testes de software não conseguem provar que um sistema funciona. Eles conseguem apenas encontrar defeitos e que o software funcionou durante as situações em que foi testado, mas não garantem nas outras situações que não foi testado. E mesmo que um teste não detecte defeitos, isso não quer dizer que o produto é de boa qualidade. Pois dependendo da forma como os testes foram conduzidos, pode ser que naquela situação e com aqueles dados fictícios, os defeitos não sejam revelados, mas no ambiente do cliente, podem aparecer. E mesmo que os testes não possam demonstrar a ausência de defeitos, como benefício dessa fase, indicam que as funcionalidades pareçam estar funcionando de acordo com o que foi solicitado e especificado.

Vamos a algumas definições encontradas nas literaturas da área sobre teste de software:

- O teste de software é uma atividade dinâmica e seu intuito é executar o programa ou modelo utilizando algumas entradas em particular e verificar se seu comportamento está de acordo com o esperado (DELAMARO, 2016);
- Teste é um conjunto de atividades que podem ser planejadas com antecedência e executadas sistematicamente (PRESSMAN; MAXIM, 2016, 466);
- Testar é verificar se o software está fazendo o que deveria fazer, de acordo com seus requisitos, e não está fazendo o que não deveria fazer (RIOS; MOREIRA, 2013).

Todo software deve sofrer um nível mínimo de teste e, de acordo com Pressman e Maxim (2016), quanto maior o nível de complexidade do sistema, mais testes, técnicas de testes e ferramentas se tornam necessárias para a obtenção da qualidade.

Para entender a etapa de teste de software, precisamos conhecer alguns conceitos e terminologias padrão para a Engenharia de Software ISTQB® (International Software Testing Qualifications Board).

A seguir, uma lista com alguns dos conceitos básicos utilizados em Teste de Software:

QUADRO 1 - GLOSSÁRIO DO TESTE COM TERMOS QUE SÃO UTILIZADOS EM TESTE DE SOFTWARE.

Conceito	Descrição
Ambiente de Teste (<i>Test Environment</i>):	Ambiente que contém hardware, instrumentação, simuladores, ferramentas de software e outros elementos de suporte necessários à realização de um teste.
Base de Teste	São todos os documentos a partir dos quais os requisitos de um determinado componente ou sistema podem ser inferidos. Documentação na qual os casos de teste estão baseados.
Condição de Teste (<i>Test Condition</i>):	Item ou evento de um componente ou sistema que pode ser verificado por um ou mais casos de teste, por exemplo: função, transação, característica, atributo de qualidade ou elemento estrutural
Defeito (<i>defect</i>):	Falha em um componente ou sistema que pode fazer com que o componente ou sistema falhe ao desempenhar sua função. Defeito, se descoberto durante a execução, pode levar a falha do componente ou do sistema.
Erro (<i>Error</i>):	Ação humana que produz um resultado incorreto.
Falha (<i>Failure</i>):	Desvio do componente ou sistema da entrega, resultado ou serviço esperado.
Ferramenta de gerenciamento de defeito (<i>Defect Management Tool</i>):	Facilita a gravação, monitoramento e alterações de defeitos. Possuem recursos orientados para o fluxo de trabalho a fim de rastrear, controlar a alocação, a correção e a nova realização de testes de defeitos.
Ferramenta de Modelagem de Teste (<i>Test Design Tool</i>):	Dá suporte à atividade de modelagem de teste por meio da geração de entradas/ <i>inputs</i> de teste a partir de uma especificação que pode estar armazenada em um repositório de ferramenta CASE.
Ferramenta de Teste (<i>Test tool</i>):	Dá suporte a uma ou mais atividades de um teste, como planejamento e controle, especificação, construção de arquivos iniciais e dados, execução e análise de testes.
Incidente (<i>Incident</i>):	É qualquer ocorrência de evento que requer uma investigação
Não conformidade (<i>Non-conformity</i>):	Trata-se do não atendimento a requisitos especificados.
Plano de Teste (<i>Test Plan</i>):	É um documento contendo o escopo, abordagem, recursos e cronograma das atividades de teste que se destina. Identifica itens de teste, recursos a serem testados, tarefas de teste, quem vai fazer cada tarefa, grau de independência do testador, o ambiente de teste, as técnicas de projeto de teste e critérios de entrada e de saída a serem usadas.

Fonte: BSTQB (online).

Podemos demonstrar a presença de defeitos quando executamos testes em um software, mas não podemos provar que eles não existem. Os testes quando são executados reduzem a probabilidade dos defeitos, mas não garantem que este sistema esteja perfeito e livre deles.

CICLO DE VIDA DO TESTE DE SOFTWARE

software testing

Quanto antes os testes de software se iniciarem, mais fácil será corrigir os defeitos encontrados e com custo reduzido. Por isso, de acordo com Lamounier (2021, p. 6), “adicionar testes de softwares ao ciclo de vida de um sistema (requisitos, projeto, codificação, testes e manutenção) se torna cada vez mais importante do que testá-lo quando o projeto já se encontra pronto, em sua versão final”.

O processo de teste, assim como o processo de desenvolvimento do software, tem um ciclo de vida que procura assegurar que o software atenda todos os requisitos e objetivos de qualidade. As etapas/fases do Ciclo de Vida de Teste de Software são:

- 1. Planejamento:** onde são elaboradas as estratégias de teste e o plano de teste a serem utilizados. Essa etapa deve permanecer ativa até que o projeto seja concluído.
- 2. Preparação:** onde é preparado o ambiente de teste, como equipamentos, pessoal, ferramentas de automação, base de testes, para que os testes possam ser executados conforme foram planejados.
- 3. Especificação:** onde as atividades de elaboração e revisão dos casos de testes e roteiros de testes são elaborados à medida que a equipe de desenvolvimento libera os módulos ou partes para o teste.
- 4. Execução:** onde são executados os testes planejados e os resultados obtidos são registrados.
- 5. Entrega:** onde o projeto é finalizado e toda documentação é finalizada e arquivada.

Podem ser encontradas outras fases/etapas para o Ciclo de Vida de testes, mas a estrutura básica consiste sempre nesta, ela deve ser baseada e compatível com a

metodologia usada no processo de desenvolvimento do sistema. Todas as fases devem ser realizadas de forma planejada e sistemática, ou seja, cada uma delas possui seus objetivos e resultados.

Problemas podem surgir em qualquer uma das fases do ciclo de vida de um software, e a probabilidade que surjam erros no código final e nos requisitos é maior. É necessário testar o software e tentar identificar essas falhas antes que elas ocorram no ambiente do cliente.

A seguir alguns motivos que justificam a importância do ciclo de vida de teste de software:

- Ajuda a assegurar que o software é confiável e com isso, garante que o cliente fique satisfeito.
- Ajuda na garantia da qualidade do software, o que contribui para a fidelização do cliente.
- Ajuda a reduzir os custos de manutenção do software.

Normalmente, um software tem que passar por alguns estágios de testes, como: (I) testes em desenvolvimento onde o sistema é testado durante o desenvolvimento para descobrir bugs e defeitos, (II) testes de release, onde a equipe de teste independente testa uma versão completa do sistema antes que ele seja liberado para o cliente para ver se ele atende ao que foi solicitado, (III) testes de usuário onde os usuários ou potenciais usuários do um sistema testam o sistema em seu próprio ambiente (SOMMERVILLE, 2018, p. 146).

O processo de teste envolve uma mistura de testes manuais e automatizados.

No teste manual, um testador executa o programa com alguns dados de teste e compara os resultados com suas expectativas; ele anota e reporta as discrepâncias aos desenvolvedores do programa. Em testes automatizados, os testes são codificados em um programa que é executado cada vez que o sistema em desenvolvimento é testado. Essa forma é geralmente mais rápida que o teste manual, especialmente quando envolve testes de regressão — reexecução de testes anteriores para verificar se as alterações no programa não introduziram novos bugs (SOMMERVILLE, 2018, p. 147).

Nos últimos anos, o uso de testes automatizados tem aumentado consideravelmente, apesar de que eles nunca poderão ser totalmente automatizados, pois os testes automatizados só verificam o que foi proposto a ser feito. Testes como uma interface de usuários não podem ser automatizados.

TÉCNICAS DE TESTE DE SOFTWARE



Os testes de software, para Lamounier (2021, p. 22) são “ferramentas de extrema importância para que o desenvolvimento evolua, de forma que quanto menos erros forem apresentados, maior será a garantia de qualidade demonstrada”. Para implementá-los de forma adequada é necessário adotar técnicas de testes de software, desde a fase de elicitação até sua entrega para garantir a qualidade do sistema.

As técnicas de teste são procedimentos técnicos e gerenciais que auxiliam na avaliação do software e nas suas melhorias. Elas podem ser aplicadas, independentemente do tipo de projeto de software ou aplicação que esteja sendo desenvolvida. É importante entender quais técnicas mais se adaptam ao tipo de sistema que será testado. Existem inúmeras técnicas de teste que podem ser usadas para avaliar diferentes aspectos ou para evitar que o sistema apresente bugs ou retorne notificações inesperadas.

A seguir vamos conhecer algumas dessas técnicas de teste.

- **Testes de Usabilidade:** são testes que simulam as condições de uso do software sobre a perspectiva do usuário final. São realizados durante toda a fase de criação do software e focam na facilidade de navegação entre as telas, clareza dos textos e as mensagens que são apresentadas aos usuários, dentre outros aspectos da interface do sistema (LAMOUNIER, 2021).

- **Teste funcional ou de caixa preta:** o teste funcional também é conhecido como teste de caixa preta e são utilizados para verificar a perspectiva do usuário e a

verificação e correção do sistema é com base nos requisitos funcionais. O teste funcional ou de caixa preta se baseiam nas especificações e consideram apenas entradas aceitas pelo componente (função) e saídas esperadas (figura 1).

FIGURA 1 - TESTE FUNCIONAL OU DE CAIXA PRETA.



Fonte: Lamounier (2021, p. 31).

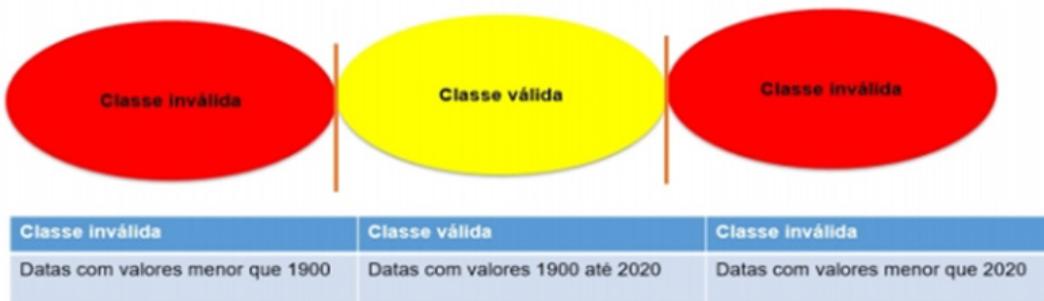
Para os testes funcionais podemos usar alguns critérios: o particionamento de equivalência e a análise de valor limite. O **particionamento de equivalência** tenta reduzir o número total de casos de teste necessários, particionando as condições de entrada em um número finito de classes de equivalência. Pode ser aplicada em qualquer nível de teste e muitos testadores aplicam ela sem saber (informalmente).

Este critério divide a entrada nas chamadas classes por equivalência e, desse modo, após a definição destas classes, pode-se assumir que qualquer elemento da classe pode ser considerado um representante fiel dela e qualquer um destes elementos tem o poder de se comportar de forma similar, isto é, caso apresente algum erro, qualquer outro elemento também detecta este erro, caso não tenha a detecção, nenhum outro também detectará. Suas classes são definidas por um conjunto de classes válidas e inválidas, como meio de entrada para testes, geralmente, sendo apresentadas uma classe válida e duas classes inválidas. Após a identificação das classes válidas e inválidas, é necessário determinar os casos de testes, de modo ao escolher um elemento da classe que gere um novo caso de teste responsável por englobar o maior número de classes possível (LAMOUNIER, 2021, p.32).

Vamos ver alguns conjuntos de diretrizes usadas para identificar as classes de equivalência:

- Se uma condição de entrada especificar um intervalo de valores (por exemplo, o programa "aceita valores de 10 a 100"), uma classe de equivalência válida (de 10 a 100) e duas classes de equivalência inválidas serão identificadas (menor que 10 e maior que 100).
- Um campo de entrada (input field) referente ao ano de aniversário aceita valores de 1900 até 2020 (figura 2).

FIGURA 2 - PARTICIONAMENTO POR EQUIVALÊNCIA.



Fonte: Lamounier (2021, p. 32).

Após ter sido realizado o particionamento por equivalência, é interessante criar uma tabela com os Casos de Teste para que fique claro a cobertura dos testes realizados, conforme quadro 2.

QUADRO 2 – CASOS DE TESTE.

Caso de teste (nome)	Valores	Status
C1	2000, 1984, 1901	Válido.
C2	1277, 1899	Inválido.
C3	2029, 2021	Inválido.

Fonte: Lamounier (2021, p. 33).

A **análise de valor limite** é uma técnica que também tem por objetivo primário reduzir o número de casos de testes. Ela é uma evolução natural da técnica de Partição de Classe de Equivalência e nos leva à ideia de que muitos problemas podem estar "ao redor" da fronteira das classes descobertas, ou seja, o maior número de erros possa estar nas fronteiras.

Vamos a um exemplo: Um campo de entrada referente a data de nascimento aceita valores de 1860 até 2860. Utilizando a análise do valor limite, o teste usaria quais valores?

Os valores seriam: 1859, 1860, 2860, 2861

- 1859 = valor limite mínimo inválido
- 1860 = valor limite mínimo válido
- 2860 = valor limite máximo válido
- 2861 = valor limite máximo inválido
-

Então, podemos considerar os seguintes passos para a análise de valor limite:

- Identificar as classes de equivalência;
- Identificar os limites de cada classe;
- Criar casos de teste para os limites escolhendo:
 1. Um ponto abaixo do limite.
 2. O limite.
 3. Um ponto acima do limite.
 4. Observe que “acima” e “abaixo” são termos relativos e dependem do valor

dos dados.

Exemplificando:

- Números inteiros: limite = 16; abaixo = 15; acima = 17.
- Números reais: limite = 5,00; abaixo = 4,99; acima = 5,01.
- **Teste Estrutural ou de Caixa Branca:** são testes preocupados com o fluxo do código fonte dos sistemas, sendo que o testador deve ter conhecimento do código fonte a ser testado. Lamounier (2021, p. 34) avalia “o comportamento interno do sistema, analisando o comportamento do software, a fim de avaliar o fluxo de dados, condições, caminhos lógicos a serem percorridos”.

Uma técnica de teste estrutural é o **teste de caminho básico**, onde é possível medir a capacidade lógica de um projeto e utilizá-la para guiar o número de caminhos a serem executados pelo sistema. Usando esta técnica, é possível executar pelo menos uma vez as instruções do programa a ser testado.

Normalmente, os **elementos de um sistema** que podem ser testados são:

- As linhas de comando;
- As funções implementadas;
- As variáveis definidas no sistema;
- Os loops existentes no software como:
 - FOR, LOOP, WHILE, DO WHILE
- Todos os ramos de uma decisão como:
 - IF (aninhado), CASE
- Requisitos de Software.

As técnicas de teste de software trazem várias vantagens para ajudar na condução de testes, possibilitando: (I) eliminar ambiguidades no código, (II) reduzir o tempo de testes, (III) definir quais condições serão necessárias para cada item que será testado e (IV) ajuda a verificar se o sistema faz o que se propôs a fazer.

FERRAMENTAS PARA AUTORIZAÇÃO DE TESTE DE SOFTWARE

Infelizmente, falhas e erros em software são mais comuns do que deveriam ser. Os bugs fazem com que a empresa desenvolvedora perca a confiança do cliente e custam muito dinheiro. Afinal, é preciso corrigir o bug e recuperar o tempo perdido do cliente que ficou parado enquanto o sistema não funcionava.

Você deve estar pensando: “Por que os sistemas apresentam tantos bugs assim?”. Um dos principais motivos é a falta de testes, ou seja, os sistemas nem sempre são testados devido aos atrasos no cronograma e, em outros casos, as equipes de software geralmente não gostam de fazer (ou não fazem) os devidos testes. Outro motivo é que pedir para um ser humano testar todo o sistema é impossível e ele vai levar muito tempo para isso.

Para resolver esse problema, colocamos a máquina testar, ou seja, escrever um programa que teste o sistema de forma automática. Uma máquina com certeza executaria o teste muito mais rápido do que uma pessoa e não ficaria cansada ou cometaria erros. Um teste automatizado é muito parecido com um teste manual, só que a máquina através de um script que fará os testes.

Imagine que você está testando manualmente a funcionalidade de cadastro de produtos em uma aplicação web. Você executaria três passos diferentes:

1. Pensaria em um cenário positivo e negativo para testar.
2. Após montar o cenário, executará a ação que quer testar.
3. Por fim, verificaria se o sistema se comportou da maneira que esperava.

Se você pensar, sempre vai executar estes três passos: monta o cenário (positivo e negativo), executa a ação e valida a saída. O cenário de teste descreve o que deve ser testado e ele é o passo inicial para a criação dos casos de testes e do roteiro. Um cenário de teste é um comportamento do sistema a ser testado. Cenários positivos procuram descrever operações que devem ser concluídas na aplicação, como por exemplo: efetuar login com sucesso. Já os cenários negativos descrevem operações que não devem ser concluídas na aplicação, como efetuar login com usuário inválido.

As vantagens de automatizar os testes:

- Gera um log do teste (laudo de teste) no console ou em um arquivo;
- Documenta a execução de toda a suíte de teste e documenta todas as falhas observadas;
- Provê informação relativa ao contexto do caso de teste e permite a análise de cada caso de teste executado;
- Facilita a análise da abrangência e do rigor dos testes (viabiliza auditar a qualidade do teste);
- Requer um explorador de log (facilita ao inspetor ou mantenedor a encontrar os elementos do log que interessem).

Os tipos de Testes que podem ser automatizados são: testes unitários, funcionais, de carga e desempenho, teste de segurança e inspeção automática de código fonte. Será que a máquina testa tudo? Infelizmente, testar todas as combinações é impossível, pois se tentarmos fazer isso, acabamos escrevendo muitos testes e com isso dificultamos a manutenção da bateria de testes, sem contar que às vezes o cronograma está atrasado. O ideal é escrever apenas um único teste para cada possível cenário diferente. A forma de fazer isso é aplicar as estratégias e técnicas de modelagem de teste de software e utilizar as ferramentas para a automação de testes de software.

A seguir, alguns exemplos de ferramentas para auxiliar na automatização dos testes de software existentes no mercado:

QUADRO 3 - FERRAMENTAS PARA AUTOMATIZAR OS TESTES DE SOFTWARE

Ferramentas	Descrição
Selenium	Ferramenta de teste open source, versátil e pode ser utilizada em aplicações voltadas para web nas plataformas e browsers Linux, Mac e Windows. É compatível com diversas linguagens de programação como: Java, Ruby, C#, PHP, Python, entre outras.
Telerik TestStudio	Ferramenta de testes considerada a mais abrangente que oferece vários tipos de testes e pode ser aplicada a diversas plataformas e a variadas linguagens.
Robotium	Ferramenta de teste com código aberto e focada em sistemas Android, possui alta performance e é voltado para testes de interface gráfica.
TestComplete	Ferramenta de teste que realiza testes em softwares voltados para web, mobile e também desktop.
Testing Whiz	Ferramenta de teste que possui diversos pacotes com soluções para testes de banco de dados, de API, de aplicativos para dispositivos móveis, dentre outros.
JMeter	Ferramenta desktop feita no Java de código aberto e que foi desenvolvida para executar testes funcionais e medir o desempenho de aplicações. O objetivo do JMeter é prover cenários de testes mais reais à forma de uso dos sistemas testados.
JUnit	Ferramenta que facilita o desenvolvimento e execução de testes unitários em código Java. Fornece uma completa API (conjunto de classes) para construir os testes e aplicações gráficas em modo console para executar os testes criados.

Fonte: a autora.

Em todas as atividades operacionais os testes podem (ou precisam) contar com o suporte de alguma ferramenta para a sua execução ou gerenciamento e automatização. As ferramentas de teste podem aliviar o fardo da produção e da execução de teste, da geração de informações e da comunicação. O testador precisa entender as técnicas de teste e as estratégias a serem usadas primeiro, para depois poder entender quais as ferramentas de teste que devem ser utilizadas para cada técnica.



Testes manuais versus testes automatizados

Engana-se que testes podem ser feitos apenas de forma manual, isto é, executados apenas com a interferência humana. Em vários casos e etapas, é possível a utilização de testes automatizados, que também garantem a qualidade desejada em um tempo reduzido e com maior precisão.

Os testes manuais, aliados a práticas de revisão e inspeção, que tem como característica a buscas por erros em artefatos de software, de maneira precoce, por sua vez, também possuem importante papel na garantia de qualidade, onde a equipe que desenvolve o sistema deverá ser capaz de estudar e analisar o problema e pensar em uma solução para, posteriormente, implementá-la.

Nesse contexto, os testes manuais são realizados, a fim de verificar que se as funcionalidades do sistema estão conforme os requisitos levantados, além de identificar erros e corrigi-los de modo que possam ser realizados quantas vezes forem necessários, o que, por muitas vezes, torna este tipo de teste dispendioso, cansativo e repetitivo, levando aos testadores a não verificar criteriosamente as mudanças significativas no código fonte, é neste cenário que surgem erros e defeitos acarretando prejuízos financeiros para a empresa de desenvolvimento e frustrando os clientes.

Os testes automatizados, por sua vez, vieram trazer uma posição contrária aos modelos manuais, ou seja, realizar testes de forma que não haja a interferência humana e com o mesmo objetivo garantir e melhorar a qualidade do sistemas.

Quando se fala de automação de testes, deve-se levar em conta que as ações e atividades são pré-definidas e analisam um conjunto de ações específicas, que, após criadas, facilmente podem ser reutilizadas.

Lamounier, Stella Marys D. Teste e controle de software: técnicas e automatização. Disponível em: Minha Biblioteca, Editora Saraiva, 2021.



Desenvolver softwares com qualidade é um desafio para a maioria dos profissionais da área de tecnologia. Por mais que se planeje e estude a construção de um sistema computacional, erros humanos são passíveis de acontecer, porém, é importante ter em mente que esses erros devem ser corrigidos e solucionados o quanto antes possível.

Lamounier, Stella Marys D. Teste e controle de software: técnicas e automatização. Disponível em: Minha Biblioteca, Editora Saraiva, 2021.p. 6.

CONSIDERAÇÕES FINAIS

Chegamos ao final da nossa terceira unidade! Espero que você tenha aproveitado ao máximo os conhecimentos apresentados sobre testes de software.

Foram expostos os conceitos básicos sobre testes de software, a importância de se realizá-los e entender o porquê ele é essencial para garantir a qualidade do software. Todos querem ter um produto de software que funcione corretamente e que não tenha erros. Para isso, faz-se necessário o uso de ferramentas que auxiliem no processo de teste de software.

Compreendemos ao longo da unidade a importância de realizar esse teste, o que acontece caso os testes não sejam realizados e que mesmo um teste não detectar defeitos, isso não quer dizer necessariamente que o produto não tenha uma boa qualidade.

Aprendemos também que quanto mais cedo os testes iniciarem, mais barato será para corrigir os defeitos encontrados. E para isso, é preciso ter um ciclo de vida de testes bem planejado e usar algumas técnicas de teste e ferramentas que podem ser aplicáveis no software em diferentes circunstâncias. As ferramentas podem ser utilizadas para classificar diferentes conceitos de testes, técnicas de design de situação de testes, técnicas de execução de teste e organizações de testes.

Espero que tenha entendido os conceitos básicos sobre testes de software e te convido a continuar seus estudos na próxima unidade!

Obrigado pela companhia.

Até mais!

LEITURA COMPLEMENTAR

Artigo: Ferramentas Para Planejamento e Execução de Testes Funcionais

Autores: Paulo Nietto

Link de acesso: https://www.researchgate.net/publication/272887427_Ferramentas_Para_Planejamento_e_Execucao_de_Testes_Funcionais_Trabalho_de_Diplomacao

Resenha: o artigo descreve sobre ferramentas automatizadas e manuais dedicados para o gerenciamento e execução de testes funcionais de software existentes no mercado. No artigo tem a identificação de quais ferramentas podem ser implantadas em uma empresa, considerando o contexto real de uma fábrica de softwares de médio porte envolvida com desenvolvimento de sistemas corporativos financeiros.

Fonte: Nietto, Paulo & Unifaccamp, Computação. (2014). Ferramentas Para Planejamento e Execução de Testes Funcionais (Trabalho de Diplomação).

MATERIAL COMPLEMENTAR



Livro: Base de Conhecimento em Teste de Software - 3^a Ed. 2012

Autores: Emerson Rios, Anderson Bastos, Ricardo Cristalli; Trayahú Moreira.

Editora: Martins Editora

Sinopse: Obra de referência para os profissionais da área, indicada para o leitor que começa a se preparar para os exames de certificação. A fim de auxiliá-lo ainda mais nessa tarefa, foram selecionados os temas mais abordados nos principais exames. De maneira complementar, esta obra, desde sua primeira edição, é também fonte de consulta para a execução das atividades relacionadas ao teste de software.



FILME/VÍDEO

Título: O Dilema das Redes

Ano: 2020

Sinopse: é um dos principais filmes quando se fala em tecnologia. Nele é mostrado como os "donos das tecnologias das redes sociais" têm o controle total da maneira que pensamos, agimos e vivemos através de algoritmos, e com isso, estão reprogramando toda uma geração.



UNIDADE

PROCESSO DE TESTE DE SOFTWARE

Professora Esp. Janaina Aparecida de Freitas

Plano de Estudos

- Conceitos sobre o Processo de Teste de Software.
- Ambiente de Teste de Software e Trabalho em Equipe.
- Métricas e Medição para Teste de Software.
- Gerência de Risco em Teste de Software.

Objetivos da Aprendizagem

- Entender os conceitos sobre o Processo de Teste de software;
- Compreender o funcionamento do ambiente de Teste de Software e o trabalho em equipe;
- Entender as Métricas e Medição que são usadas no Teste de Software e como podem ser aplicadas;
- Conhecer a Gerência de Risco em Teste de Software.

INTRODUÇÃO

Olá, aluno (a), seja bem-vindo (a) a última unidade e ao estudo sobre o processo de teste de software, o ambiente de teste e equipe de testes, as métricas e medição para teste e a gerência de risco em teste de software.

Nesta unidade iremos começar a entender sobre o processo de teste de software e a sua importância na execução do teste. O processo de teste de software, discutido no primeiro tópico, é um conjunto de atividades/tarefas necessárias para definir, desenvolver, testar e manter um produto de software de alta qualidade, como todo processo, é uma abordagem adaptável e executados em paralelo ao processo de desenvolvimento de software, desde o início do ciclo de vida do software.

No segundo tópico vamos compreender o funcionamento e a importância do ambiente de teste de software e do trabalho em equipe. Pois ao executar testes em um software devemos pensar no ambiente onde eles serão realizados, ou seja, toda a infraestrutura onde acontecerá. O ambiente de testes envolve configurações de hardware, software, ferramentas de automação, equipe de teste envolvida, aspectos organizacionais, dispositivos, suprimentos, rede, documentações e outros ambientes necessários para a execução, pois precisam simular o ambiente de produção.

No terceiro tópico vamos entender as métricas e medição que são usadas durante o teste de software e como podem ser aplicadas para obtermos o controle do projeto e, com isso, poder gerenciá-lo. Você já parou para pensar que a engenharia é, por sua natureza, uma disciplina quantitativa? E que um elemento-chave de qualquer processo de engenharia é a medição? Medimos e avaliamos os testes de software para estimar se estamos perto ou longe dos objetivos que foram definidos no projeto de teste, como cronograma, custos, compatibilidade com os requisitos, qualidade, entre outros.

No quarto e último tópico vamos conhecer a gerência de risco em teste de software. Hoje em dia qualquer empresa corre riscos, todos os dias, se em algum momento seus computadores e sistemas pararem de funcionar ou um site sair fora do ar e trazer muitos prejuízos para uma empresa. Devido a esses problemas que podem surgir, as empresas passaram a investir para evitar riscos de defeitos em seus softwares, criando planos de contingência para contornar os problemas, pois o risco é uma probabilidade de ocorrência de uma perda para a empresa.

Preparado (a) para continuar? Então, vamos seguir em frente.

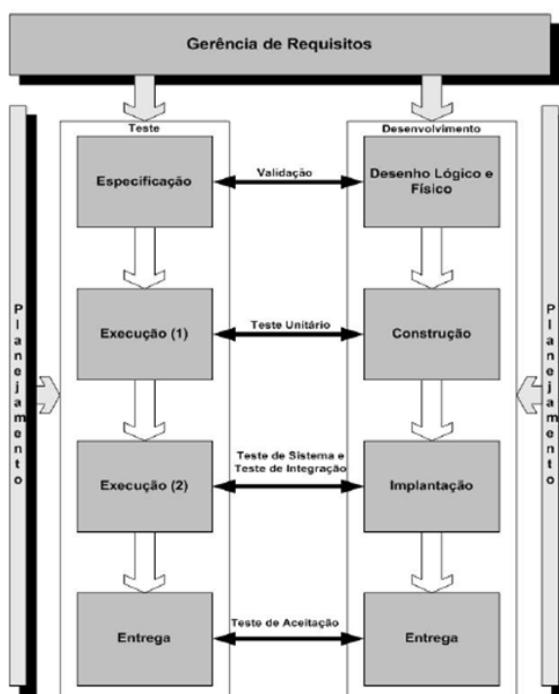
Boa leitura e bons estudos!

CONCEITOS SOBRE O PROCESSO DE TESTE DE SOFTWARE

Durante a produção de um sistema são necessárias diversas fases/etapas, as quais são compostas por várias tarefas. Este conjunto chama-se processo de software e deve sempre cumprir todos os estágios estabelecidos para ele. Para a execução do teste de software, também temos um processo para seguir, ou seja, um conjunto de atividades/tarefas necessárias para definir, desenvolver, testar e manter um produto de software de alta qualidade, como todo processo, é uma abordagem adaptável executados em paralelo ao processo de desenvolvimento de software, desde o início do ciclo de vida dele (figura 1).

FIGURA 1 – MODELO DE INTEGRAÇÃO ENTRE OS PROCESSOS DE DESENVOLVIMENTO E

Modelo de Integração entre os processos de desenvolvimento e teste



Fonte: Rios e Moreira (2013)

De acordo com Sommerville (2011), o processo de teste tem dois objetivos distintos:

QUADRO 1- OBJETIVOS DO PROCESSO DE TESTE

1. Demonstrar ao desenvolvedor e ao cliente que o software atende a seus requisitos.	<ul style="list-style-type: none">Para softwares customizados: significa que deve haver pelo menos um teste para cada requisito descrito no documento de requisitos.Para softwares genéricos: significa que deve haver testes para todas as características do sistema, além de suas combinações, que serão incorporadas ao release do produto.
2. Descobrir situações em que o software se comporta de maneira incorreta, indesejável ou de forma diferente das especificações.	<ul style="list-style-type: none">São situações decorrentes das consequências de defeitos de software.O teste de defeitos preocupa-se com a eliminação de comportamentos indesejáveis do sistema, tais como panes, interações indesejáveis com outros sistemas, processamentos incorretos e corrupção de dados.

Fonte: Adaptado de Sommerville (2011, p.145).

O primeiro objetivo do processo de teste leva ao teste de validação, onde se espera que o sistema execute corretamente usando determinado conjunto de casos de testes que refletem o uso esperado do sistema. Já o segundo objetivo leva a testes de defeitos, onde os casos são projetados para expor os defeitos. Resumindo, quando se executa os testes de validação, encontram-se defeitos no sistema; quando se executa os testes de defeitos, alguns dos testes mostraram que o programa corresponde a seus requisitos (SOMMERVILLE, 2011).

Um processo de teste de software procura estruturar as etapas, as atividades, os artefatos, os papéis e as responsabilidades do teste, com isso permite que a empresa controle todo o ciclo do teste, minimizando os riscos, agregando valor e garantindo qualidade ao software (DELAMARO, 2016).

Antes de pensar na ideia de um processo de testes de software, as empresas desenvolvedoras precisam entender que em primeiro lugar, quando se realiza um teste temos todas as atividades envolvidas antes, durante e após a execução dele, ou seja, a busca por erros e defeitos em um sistema é uma das várias fases que abrange o processo.

Quando testamos, executamos diversas atividades, como:

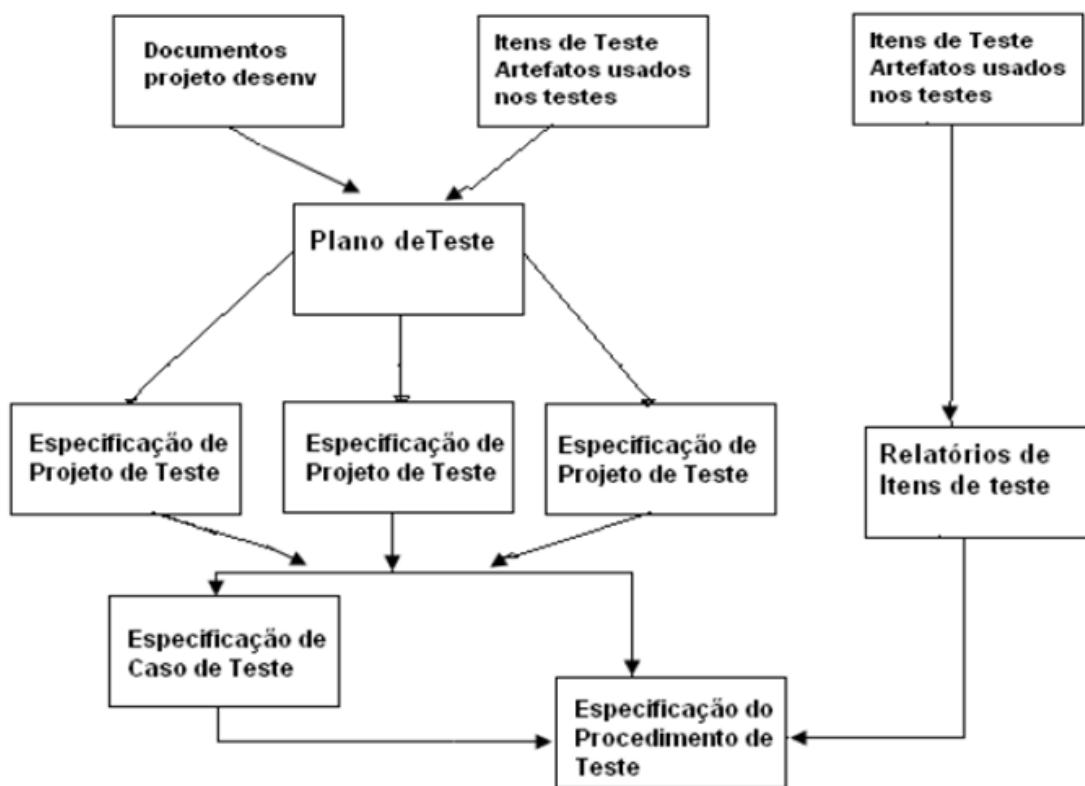
QUADRO 2 – ATIVIDADES DO PROCESSO DE SOFTWARE.

Planejamento e Controle dos Testes	Envolve atividades que determinam quais serão as abordagens dos testes e procura-se entender as metas e objetivos do projeto e do cliente, os riscos que envolve o projeto e o escopo (o que testar, como testar), a medição e análise dos resultados, como será a execução de ações corretivas e a tomada de decisões.
Análise e Modelagem dos Testes	Etapa onde as condições do teste são produzidas, ou seja, “o quê” será testado. Tem o propósito de exercitar as condições de uso do software, tentando atingir o máximo de cobertura e com o mínimo de casos de teste.
Implementação e Execução dos Testes	Onde se realiza a especificação dos procedimentos e/ou scripts de teste combinando os casos de teste, ou seja, onde se transforma as condições de teste em casos de teste e procedimentos de teste. Também é registrado os resultados da execução do teste; realizado comparações, identificado erros, falhas e defeitos (bugs) e onde se executa os casos de teste automatizados.
Avaliação dos Critérios de Saída e Relatórios dos Testes	Etapa onde é avaliado a execução de testes realizados nas etapas anteriores do processo. Também é avaliado se é necessário realizar mais testes (porcentagem de testes, número de requisitos cobertos) e se os critérios de saída devem ser alterados. Também é nesta etapa que são elaborados os relatórios de testes.
Atividades de Encerramento do Teste	Etapa onde é realizada a coleta de todos os dados de todas as outras etapas, e onde é gerado a documentação como planos de teste, condições de teste, casos de teste, base de teste utilizada etc.), porcentagens e números de testes realizados.

Fonte: Adaptado de Sommerville (2011, p. 146-147).

As atividades de teste podem acontecer de forma concorrente, não havendo uma ordem, pois quando e como elas serão executadas vai depender de quais técnicas e estratégias de teste serão adotadas no projeto (figura 2).

FIGURA 2 - RELACIONAMENTO ENTRE OS DOCUMENTOS DE TESTE DO PROCESSO DE TESTE DE SOFTWARE



Fonte: IEEE 829 (1998).

O processo de teste de software auxilia na organização dessas atividades, dos passos a serem seguidos, quais artefatos serão usados, papéis e responsabilidades da equipe envolvida.

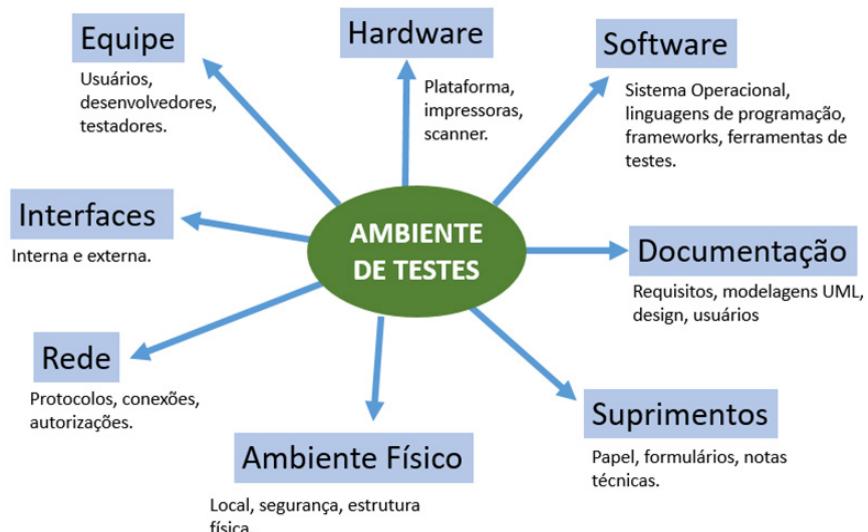
AMBIENTE DE TESTE DE SOFTWARE E TRABALHO EM EQUIPE

Quando pensamos em executar testes em um sistema, devemos pensar no ambiente onde serão realizados, ou seja, toda a infraestrutura onde ele será realizado.

Um ambiente de testes é um conjunto específico de configurações de hardware, software, ferramentas de automação, equipe envolvida, aspectos organizacionais, dispositivos, suprimentos, rede, documentações e outros ambientes necessários para a execução de testes que sejam precisos e que simulem o ambiente do usuário. Este ambiente deve proporcionar condições conhecidas e que sejam controladas para a realização dos testes (SOMMERVILLE, 2018).

A figura 3 mostra os elementos do ambiente de testes que devem ser levados em consideração para o planejamento antes dos testes iniciarem.

FIGURA 3 - ELEMENTOS DO AMBIENTE DE TESTES



Fonte: a autora.

O ambiente de teste deve ser pensado como uma estratégia de teste e adicionado ao planejamento dos testes antes dos mesmos iniciarem. O ideal é que ele seja similar ao ambiente do usuário, onde ele utilizará o software. Neste momento, o responsável pela organização do ambiente de testes, normalmente o arquiteto de testes, deve pensar em todos os elementos possíveis para a execução, como: massa ou base de testes, modelos de dados que serão usados, configuração dos softwares usados (devem ser iguais aos do usuário), tipo de testes que serão executados, técnicas de teste apropriadas.

As configurações usadas no ambiente de testes devem fornecer uma ideia de como serão conduzidos e como as atividades serão executadas. Por isso, é importante fornecer um ambiente conhecido e controlado para a execução, a fim de assegurar que os resultados sejam precisos e válidos na busca de erros, defeitos e falhas (LAMOUNIER, 2021).

O ideal, é trabalhar em ambientes de teste isolados, ou seja, que não sofram influências externas durante os testes. É difícil determinar os testes relacionados ao ambiente de produção, pois vai depender de alguns fatores, como: o tamanho do projeto, o orçamento disponível e cronograma.

Durante o planejamento do processo de teste de software é definido três tipos de ambiente:

1. Ambiente de Desenvolvimento: onde são realizados os testes unitários e os de integração. Ou seja, o ambiente conhecido como o computador de quem está desenvolvendo o sistema.
2. Ambiente de Testes: onde são realizados os testes de integração, sistema, regressão, aceitação e os testes funcionais. Ou seja, o ambiente utilizado para os testes manuais e automatizados.
3. Ambiente de Produção: onde são realizados os testes de estresse, performance e os testes de aceitação pelo usuário. Ou seja, o ambiente que os usuários irão usar.

Podem ser usados os ambientes virtuais (máquinas virtuais), pois na realidade atual, eles vêm ganhando espaço por serem mais econômicos. Uma máquina virtual é um software que permite ao arquiteto ou testador criar vários ambientes de testes, com diferentes configurações de software, hardware, sistemas operacionais, suprimentos, dispositivos, utilizando como se fosse a mesma máquina física do usuário (LAMOUNIER, 2021).

Ao definirmos o ambiente de teste, é importante considerar:

- (I) qual o sistema operacional usado;
- (II) qual a arquitetura do sistema;
- (III) identificar todos os componentes que fazem parte do sistema;
- (IV) como será o meio de acesso ao sistema;
- (V) quais as linguagens de programação e frameworks utilizados;
- (VI) como será a conectividade entre os ambientes (cliente).

Outro ponto importante a ser considerado no ambiente de teses é a presença de pessoas, ou seja, uma equipe de teste de software, pois sem elas nenhum projeto é testado. A equipe deve ser formada por pessoas que tenham um objetivo único e para que ele seja alcançado, elas devem ser e estar motivadas, treinadas e as tarefas devem ser específicas de cada pessoa, ou seja, considerando suas habilidades e competências (LAMOUNIER, 2021).

Assim, para uma empresa de desenvolvimento de software é importante a formação de equipes de teste que sejam constituídas por pessoas qualificadas tecnicamente e que saibam trabalhar em equipes, pois assim aumentam as chances de sucesso dos projetos desenvolvidos.

A seguir, é mostrada a matriz de responsabilidade com as funções que cada membro da equipe de testes deve desenvolver, segundo Rios (2013).

QUADRO 3 – MATRIZ DE RESPONSABILIDADES

Membro da Equipe	Função
Líder do Projeto de Testes	Responsável pela liderança de um projeto de testes específico, seja um projeto novo ou um de manutenção.
Arquiteto de Teste	Responsável pela montagem da infraestrutura de teste, montando o ambiente, escolhendo as ferramentas de teste e capacitando a equipe para executar o seu trabalho neste ambiente de teste.
Analista de Teste	Responsável pela modelagem e elaboração dos Casos de Teste e pelos Scripts de Teste. Pode ser que em alguns casos os scripts sejam elaborados pelos testadores.
Testador	Responsável pela execução dos casos de Teste e Scripts de Teste.

Fonte: Adaptado de Rios (2013).

Esses são os perfis mais conhecidos para formar uma equipe de testes. Pode ser que algumas empresas utilizem outros.

Qual o perfil da equipe de testes? Que todos sejam exploradores, gostem de resolver problemas, sejam incansáveis, muito criativos, um pouco perfeccionistas, que gostem de exercitar o julgamento e, muitas vezes, persuasivos. O trabalho de uma equipe de testes às vezes é muito estressante e cansativo, pois encontrar erros e defeitos em um sistema não é fácil e envolve atenção e cuidado. Os membros da equipe devem conhecer o processo de desenvolvimento do software no qual está inserido e também conhecer quais são os artefatos de entrada e saída de cada fase do processo, principalmente das que envolvem atividades de testes.

MÉTRICAS E MEDAÇÃO PARA TESTE DE SOFTWARE



A engenharia é, por sua natureza, uma disciplina quantitativa e o elemento-chave de qualquer processo de engenharia é a medição. Conforme Pressman e Maxim (2016, p. 653) a “métrica de produto ajuda os engenheiros de software a visualizar o projeto e a construção do software, focando nos atributos específicos e mensuráveis dos artefatos da engenharia de software”. Ela é usada para melhorar o entendimento dos atributos dos modelos criados e para avaliar a qualidade dos sistemas.

Por que medir os produtos? Para obter controle de um projeto e também para gerenciá-lo, além disso, a medição permite avaliar se estamos perto ou longe dos objetivos planejados. Um engenheiro de software coleta medidas e desenvolve métricas para obter indicadores.

Por que é importante? Sempre haverá um elemento qualitativo na criação de software. O problema é que a avaliação qualitativa pode não ser suficiente. É preciso ter critérios objetivos para ajudar a direcionar o projeto de dados, arquitetura, interfaces e componentes. Ao testar, precisamos de orientação quantitativa, que nos auxiliará na seleção de casos de teste e seus objetivos. A métrica de produto proporciona uma base por meio da qual análise, projeto, codificação e teste podem ser conduzidos mais objetivamente e avaliados de maneira mais quantitativa (PRESSMAN; MAXIM, 2016, p. 653).

Objetivo das métricas: (I) ajuda a caracterizar estatisticamente as atividades de teste de um projeto; (II) medir, avaliar e sugerir melhorias nas atividades de teste ou de desenvolvimento para projetos futuros e (III) permite a realização de estudos a respeito de decisões tomadas ao longo dos testes.

Objetivos da medição: (I) reduzir o número de falhas em programas entregues e (II) reduzir o esforço gasto em retrabalho desnecessário.

Qual é a diferença entre medida e métrica? Vamos conhecer alguns conceitos relacionados às métricas para facilitar o entendimento:

QUADRO 4 – CONCEITOS DE MEDIDA, MEDIÇÃO E INDICADOR

Medida	É a indicação quantitativa da extensão, quantidade, capacidade ou tamanho de algum atributo de um produto ou processo. Exemplo: nº de erros detectados na revisão de um módulo de software, quantidade de classes-chave.
Medição	É o ato de determinar uma medida. O IEEE define métrica como “uma medida quantitativa do grau com o qual um sistema, componente ou processo possui determinado atributo”. Exemplo: investigação de um nº de revisões de módulos para recompilar medidas do nº de erros encontrados em cada revisão.
Métrica	Medida quantitativa do grau de posse de um atributo dado por parte de um sistema, componente ou processo Exemplo: Média de erros detectados por revisão ou nº de erros encontrados por pessoa e hora em revisões
Indicador	É uma métrica ou combinação de métricas que proporcionam informações sobre o processo de software, em um projeto de software ou no próprio produto. Um engenheiro de software coleta medidas e desenvolve métricas para obter indicadores

Fonte: Adaptado de Pressman e Maxim (2016, p. 654).

Para Sommerville (2018), o objetivo da medição é usá-la para revisões e fazer julgamento sobre a qualidade de software, também pode ser usada para realçar áreas do software que podem ser melhoradas a longo prazo. Para Pressman e Maxim (2016, p. 676) as métricas de teste podem ser classificadas em duas grandes categorias:

1. Métricas que tentam prever o número provável de testes necessários em vários níveis de teste.
2. Métricas que focalizam a abrangência do teste para determinado componente.

Para os autores, a maioria das métricas para teste se concentram no processo de teste e não nas características técnicas dos testes que são executados, pois os testadores

se baseiam em métricas de requisitos, projeto e implementação para guiá-los no projeto e execução dos casos de teste.

As métricas usadas no teste de software podem ser divididas em (SOMMERVILLE, 2018):

- **Métricas básicas:** obtidas diretamente do esforço do teste. Por exemplo: quantidade de casos de testes usados, executados, bloqueados, reexecutados etc.
- **Métricas derivadas:** obtidas pelo Gerente ou pelo Líder de teste, através da conversão das métricas básicas em dados mais úteis. Exemplo: percentual de testes concluídos, taxa de defeitos descobertos.

A seguir uma lista de exemplos de métricas de Teste:

- Quantidade de falhas;
- Tempo gasto com teste (por fase);
- Esforço gasto com teste (por fase);
- Natureza das causas das falhas (artefato do processo de desenvolvimento);
- Distribuição de falhas por item de teste;
- Modos de observação das falhas (durante os testes ou após a entrega);
- Categoria das falhas (gravíssima, grave, normal, leve);
- Tempo para corrigir o defeito que originou uma falha;
- Esforço para corrigir o defeito que originou uma falha.

As métricas de teste podem ser classificadas em: métricas de produto, métricas de processo de teste e as métricas de projeto.

QUADRO 5 – CLASSIFICAÇÃO DAS MÉTRICAS DE TESTE DE SOFTWARE

Métricas de Produto	Auxiliam no controle da qualidade do produto que está sendo testado. Exemplo: Número de ocorrências, Status das ocorrências, Índice de Densidade de defeitos, Índice de Severidade de defeitos, Tempo para arrumar um defeito, Tempo médio para encontrar um defeito, Qualidade de falhas encontradas no produto, Tipos de defeitos encontrados e Cobertura dos testes, Efetividade de Caso de teste, Defeitos por quantidade de linhas de código (Kloc), Situação ou Tendência dos defeitos em função do tempo, Providências adotadas em relação aos defeitos, Defeitos por módulo e Defeitos por tecnologia utilizada.
Métricas de Processo	Auxilia no controle da qualidade do processo de teste. Exemplo: Número de Casos de teste, Taxa de falhas na primeira execução dos Casos de Teste, Custo dos testes, Relação entre defeitos e ocorrências, Ocorrências pendentes de correção, Probabilidade de defeitos, Mudanças no escopo e Densidade de defeitos por Unidade.
Métricas de Projeto	Producem relatórios sobre o status do projeto, andamento, funcionalidade, qualidade, despesas, consumos de recursos. Exemplo: tempo de teste estimado X tempo de teste efetivamente utilizado, Fator de segurança, Tempo necessário para executar um teste, Tempo disponível para o esforço de teste, Taxa de esforço de teste e Categoria de defeitos.

Fonte: Adaptado de Pressman e Maxim (2016, p. 655-656).

O ideal é que as métricas de teste sejam identificadas quando se inicia o projeto. Elas devem ser quantificáveis, que possam ser coletadas facilmente, que sejam simplificadas para que todos entendam e que tenham um propósito.

As métricas que forem sendo coletadas devem ser exploradas durante o andamento do projeto e devem ser guardadas para uso no futuro em outras estimativas de projetos novos. Assim, como temos diferentes empresas com diferentes projetos e processos de desenvolvimento, todos adaptáveis, também temos diferentes métricas para medir e avaliar a qualidade de software.



GERÊNCIAS DE RISCO EM TESTE DE SOFTWARE

Já parou para pensar que qualquer empresa corre riscos todos os dias, se em algum momento seus computadores e sistemas pararem de funcionar ou um site fora do ar? Tudo isso pode trazer muitos prejuízos para ela.

Para uma empresa, temos o risco relacionado à dependência de equipamentos e também da ocorrência de erros de software. Hoje, devido a esses problemas que podem surgir, as empresas passaram a investir para evitar riscos de defeitos em seus softwares, criando planos de contingência para contornar os problemas (PRESSMAN; MAXIM, 2016).

Nem sempre podemos aliar um risco a uma perda, pois um risco pode estar sempre presente, mas nem sempre gera uma perda. Existem riscos que sempre se transformam em perdas, por exemplo, um avião sempre corre risco de cair, mas a perda só existirá se isso ocorrer. Ou seja, o risco é uma probabilidade de ocorrência de uma perda para a empresa (RIOS, 2013).

Vamos a alguns conceitos usados no gerenciamento de riscos: (PRESSMAN; MAXIM, 2016):

- **Risco:** é a probabilidade de insucesso, de malogro de determinada coisa, em função de acontecimentos eventuais, incertos, cuja ocorrência não depende, exclusivamente, da vontade dos interessados. Uma perda grande para a empresa.
- **Análise de Risco:** é a avaliação dos recursos de informação, seus controles e suas vulnerabilidades.
- **Ameaça:** é a capacidade de alguém explorar a vulnerabilidade de um sistema.
- **Vulnerabilidade:** é uma falha de projeto, implementação ou programação.
- **Controle:** maneira de reduzir as causas de riscos.

Será que testar o software é uma atividade com riscos? Sim, a atividade de testar é bastante ligada ao risco, pois custa dinheiro e não é fácil garantir que nenhum defeito ocorra enquanto o software estiver em uso pelo usuário.

O risco é um dos elementos mais importantes ao se elaborar um projeto de testes, por isso precisa ser analisado e definido os níveis de prioridade. As equipes de teste das empresas devem procurar um nível de cobertura dos testes que minimizem a possibilidade de defeitos e falhas. Não é fácil classificar o risco e determinar o custo de criação de um controle que evite a ocorrência desse risco. A relação custo-benefício precisa ser avaliada antes de tomar qualquer decisão, porque o custo do controle do risco pode ser maior do que o risco mesmo (RIOS, 2013).

Um risco pode ser determinado de três formas, segundo o QAI – *Quality Assurance Institute*:

QUADRO 6 - FORMAS DE DETERMINAR UM RISCO

Intuição ou julgamento	Profissionais qualificados e com experiência na área de teste usam a sua intuição para dizer que a ocorrência de um risco pode custar mais caro do que os controles necessários para que ele não ocorra.
Consenso	Consenso na equipe de que aquele risco tem um grau alto de severidade, nesses casos, não há necessidade de medir-se financeiramente o custo do risco, pois há consenso que a sua ocorrência causará um prejuízo muito grande, e o bom senso indica a criação de algum tipo de controle que evite a sua ocorrência.
Fórmula de Risco	A magnitude do risco é calculada através de uma fórmula. Existem dados financeiros que permitem medir o custo da perda pela ocorrência do risco. Medindo-se cada custo pelo número de ocorrências, por exemplo, por ano, temos uma estimativa de perdas. Neste caso podemos ter resultados bastante precisos sobre as perdas.

Fonte: QAI – Quality Assurance Institute (online).

Uma lembrança importante que o testador deve sempre manter em mente ao fazer a sua análise de riscos é que eles mudam com o tempo por diversos fatores. Caso o sistema precise voltar a ser testado após as mudanças, em decorrência de alterações é necessário que os riscos também tenham que sofrer uma manutenção. Neste caso, uma nova estratégia de testes deve ser planejada.

Uma equipe de testes bem organizada consegue melhores resultados, pois o risco é um dos elementos trabalhados no momento de se elaborar o projeto de testes. Assim, ao fazer uma análise dos riscos, a equipe deve pensar na probabilidade da ocorrência, no impacto e na perda que estão associados a ele. Por exemplo, muitos riscos podem estar associados ao uso de uma nova tecnologia ainda não perfeitamente dominada pela equipe de desenvolvimento ou pelas prioridades estabelecidas para algumas funcionalidades específicas da área de negócio do cliente (RIOS, 2013).

Assim, uma análise de riscos bem organizada e detalhada feita pela equipe de testes ajuda a estabelecer quais as prioridades do que será testado, minimizando os defeitos ou a probabilidade de que esses riscos ocorram.

Pense na seguinte pergunta: “Quando os testes de software param? ”. Depois de tudo que estudamos, a resposta vai depender de cada projeto e de cada empresa desenvolvedora. Conforme Pressman e Maxim (2016, p. 656) os testes de software sempre trazem informações suficientes para uma tomada de decisão sobre os defeitos e riscos, com isso o grau de maturidade e qualidade do software que está sendo desenvolvido. Mas é importante saber que sempre temos uma versão beta para usarmos.



Riscos de software

Embora haja muito debate sobre qual é a melhor definição para risco de software, há um consenso geral de que o risco sempre envolve duas características: incerteza – o risco pode ou não ocorrer; ou seja, não existem riscos com 100% de probabilidade 1 – e perda – se o risco se tornar realidade, consequências ou perdas indesejadas ocorrerão [Hig95]. Quando os riscos são analisados, é importante quantificar o nível de incerteza e o grau de perda associados a cada risco. Para tanto, consideram-se diferentes categorias de risco. Riscos de projeto ameaçam o plano do projeto. Isto é, se os riscos do projeto se tornarem reais, é possível que o cronograma fique atrasado e os custos aumentem. Os riscos de projeto identificam problemas potenciais de orçamento, cronograma, pessoal (equipes e organização), recursos, clientes e requisitos e seu impacto sobre o projeto de software. Riscos técnicos ameaçam a qualidade e a data de entrega do software a ser produzido. Se um risco técnico em potencial se torna realidade, a implementação pode se tornar difícil ou impossível. Os riscos técnicos identificam problemas em potencial de projeto, implementação, interface, verificação e manutenção. Além disso, a ambiguidade de especificações, a incerteza técnica, a obsolescência técnica e a tecnologia “de ponta” também são fatores de risco. Riscos técnicos ocorrem porque o problema é mais difícil de resolver do que se pensava. Riscos de negócio ameaçam a viabilidade do software a ser criado e muitas vezes ameaçam o projeto ou o produto. Candidatos aos cinco principais riscos de negócio são (1) criar um excelente produto ou sistema que ninguém realmente quer (risco de mercado), (2) criar um produto que não se encaixa mais na estratégia geral de negócios da empresa (risco estratégico), (3) criar um produto que a equipe de vendas não sabe como vender (risco de vendas), (4) perder o apoio da alta gerência devido à mudança no foco ou mudança de profissionais (risco gerencial) e (5) perder o orçamento ou o comprometimento dos profissionais (riscos de orçamento). É extremamente importante observar que uma simples classificação de risco nem sempre funcionará. Alguns riscos são impossíveis de prever.

Fonte: Pressman e Maxim (2016, p. 778 – 779).



A identificação do risco é uma tentativa sistemática de especificar ameaças ao plano do projeto (estimativas, cronograma, recursos etc.). Ao identificar os riscos conhecidos e previsíveis, o gerente de projeto dá o primeiro passo para evitá-los quando possível e controlá-los quando necessário.

Fonte: Pressman e Maxim (2016, p. 780)

CONSIDERAÇÕES FINAIS

Chegamos ao final da nossa última unidade! Espero que você tenha aproveitado ao máximo os conhecimentos apresentados nesta etapa, onde aprendemos sobre o Processo de Teste de Software, o ambiente e equipe de testes, as métricas e medição para teste, bem como a gerência de risco em teste de software.

Foram apresentados o processo de teste de software e a sua importância na execução do teste para manter um produto de software de alta qualidade. Dentro desse processo compreendemos também o funcionamento e a importância do ambiente de teste de software e do trabalho em equipe. Pois ao executar testes em um software devemos pensar no ambiente onde os testes serão realizados, ou seja, toda a infraestrutura onde o teste será realizado.

Compreendemos ao longo da unidade a importância de entender as métricas e medição de teste de software e como elas podem ser aplicadas para obtermos o controle do projeto, poder gerenciá-lo e saber se estamos perto ou longe dos objetivos definidos e planejados.

Também conhecemos sobre a gerência de risco em teste de software e como é importante que as empresas invistam em estratégias para evitar riscos de defeitos em seus softwares, criando planos de contingência para contornar os problemas.

Espero que tenha entendido sobre os conteúdos aprendidos até aqui e obrigado pela companhia.

Forte abraço!

LEITURA COMPLEMENTAR

Artigo Fundamentação teórica das métricas de software

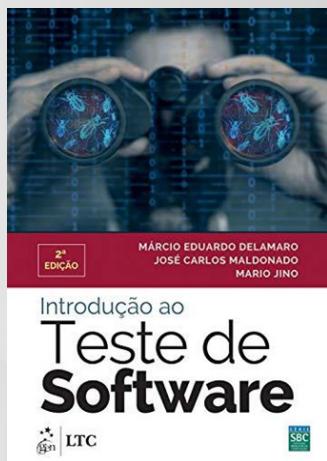
Autores: Adriano Garcia Tiago

Link de acesso: file:///C:/Users/janaina.freitas/Downloads/Tiago_AdrianoGarcia_M.pdf

Resenha: o artigo visa mostrar as pesquisas sobre tamanho e estimativas de complexidade para sistemas de software são o foco na fundamentação das bases da Engenharia de Software. O artigo descreve sobre conceitos importantes para o entendimento da base fundamental para uso de métricas de software.

Fonte: Tiago. Adriano Garcia Fundamentação teórica das métricas de Software -- Campinas. São Paulo: 2006.

MATERIAL COMPLEMENTAR



Título: Introdução ao Teste de Software

Autor: DELAMARO, Márcio.

Editora: GEN LTC; 2º edição, 2021

Sinopse: O livro de referência para os profissionais da área de teste de software. A proposta é oferecer aos profissionais as informações básicas relativas às técnicas de teste, bem como as formas de aplicá-las nos mais variados domínios e tipos de software.



FILME/VÍDEO

Título: O Quinto Poder

Ano: 2014

Sinopse: Ao fundar o polêmico site WikiLeaks, Julian Assange conta com o apoio do amigo Daniel Domscheit-Berg. O objetivo da página é fornecer uma plataforma para que denunciantes, anonimamente, exponham segredos do governo e crimes corporativos. Com o crescimento do site, a dupla logo passa a dar mais furos noticiosos do que a mídia convencional. O grau de influência de Assange aumenta e a relação entre os dois amigos acaba bastante abalada.

REFERÊNCIAS BIBLIOGRÁFICAS

ABNT. NBR ISO 9000-2005. Sistemas de Gestão da Qualidade de Software. Rio de Janeiro: ABNT, 2005.

BSTQB. ISTQB Accredited Material Provider. [s.d]. Disponível em: <https://bstqb.org.br/b9/atm>. Acesso em: 10 de jan. de 2023.

DELAMARO, Márcio. Introdução ao Teste de Software . Disponível em: Minha Biblioteca, (2^a edição). Grupo GEN, 2016.

GONÇALVEZ, Priscila de F.; BARRETO, Jeanine dos S.; ZENKER, Aline M.; et al. Testes de software e gerência de configuração. Grupo A, 2019. E-book. ISBN 9788595029361. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9788595029361/>. Acesso em: 07 dez. 2022.

GUERRA, Ana Cervigni. COLOMBO, Regina Maria Thienne. Tecnologia da Informação: Qualidade de Produto de Software. PBQP Software, 2000.

IEEE, The Institute of Electrical and Electronics Engineers. IEEE Std 829: Standard for Software Test Documentation. New York: IEEE Computer Society, September, 1998.

LAMOUNIER, Stella Marys D. Teste e controle de software: técnicas e automatização. Disponível em: Minha Biblioteca, Editora Saraiva, 2021.

PRESSMAN, Roger S.; MAXIM, Bruce R. Engenharia de software. (9^a edição). Grupo A, 2021.

PRESSMAN, Roger S; Maxim, Bruce R. Engenharia de software: Uma abordagem profissional. 8. ed. – Porto Alegre: AMGH, 2016.

PRESSMAN, Roger, S. e Bruce R. Maxim. Engenharia de software. São Paulo: Pearson Prentice Hall, 2016.

QAI. QAI – Quality Assurance Institute. [s.d]. Disponível em: <https://www.qaiglobalinstitute.com/>. Acesso em: 11 de jan. de 2023.

RIOS, Emerson. MOREIRA, Trayahú, Teste de Software, 3^a Ed. Alta Books, Rio de Janeiro, 2013.

SOMMERVILLE, Ian. Engenharia de Software, São Paulo: Pearson Prentice Hall, 2011.

SOMMERVILLE, Ian. Engenharia de Software. 10 ed. São Paulo: Pearson Education do Brasil, 2018.

SOMMERVILLE, Ian. Engenharia de Software. 10 ed. São Paulo: Pearson Education do Brasil, 2018.

SOMMERVILLE, Ian. Engenharia de Software. 9^a edição. Pearson, 2011.

ZANIN, Aline; JÚNIOR, Paulo A P.; ROCHA, Breno C.; et al. Qualidade de software. Grupo A, 2018.

CONCLUSÃO GERAL

Prezado (a) aluno (a),

Neste material, busquei trazer para você os principais conceitos a respeito da gestão de produtos e marcas. Para tanto abordamos as definições teóricas e, neste aspecto, acreditamos que tenha ficado claro para você o quanto é estratégico para gestores de todas as áreas de uma organização compreenderem os processos de identificação de necessidades e oportunidades no mercado para a criação e oferta de produtos e serviços que atendam as expectativas dos consumidores.

Destacamos também importância histórica das marcas para identificação e diferenciação dos produtos em relação aos concorrentes e vimos o quanto a construção de uma marca baseada em critérios sólidos podem contribuir para o sucesso de um produto ou empresa no mercado. Além dos aspectos teóricos que contribuíram profundamente para o entendimento dos assuntos aqui abordados, trouxemos vários exemplos e técnicas para uma melhor compreensão sobre criação e gestão de marcas fortes.

Levantamos também aspectos históricos que nos levaram a chegar nas formulações, processos e técnicas que hoje aplicamos nas organizações. Esse olhar para o passado para entender o presente e visualizar o futuro é algo inerente aos empreendedores que pensam em suas organizações como gestores eficientes e antenados.

Ao pensarmos em uma organização voltada a administração estratégica, como aqui abordamos e contemplando também a comunicação como cultura organizacional, temos que sempre levar em consideração o diálogo, o respeito e o ouvir nossos parceiros de trabalho, nossos colaboradores e todos aqueles que integram nossa equipe.

A partir de agora acreditamos que você já está preparado para seguir em frente desenvolvendo ainda mais suas habilidades para criar e desenvolver produtos e marcas de sucesso no mercado e realizar bons negócios.

Até uma próxima oportunidade. Muito Obrigado!

