

QUALIDADE DE SOFTWARE



PROFESSORA

Esp. Janaina Aparecida de Freitas

ACESSE AQUI
O SEU LIVRO
NA VERSÃO
DIGITAL!

DIREÇÃO UNICESUMAR

Reitor Wilson de Matos Silva **Vice-Reitor** Wilson de Matos Silva Filho **Pró-Reitor de Administração** Wilson de Matos Silva Filho **Pró-Reitor Executivo de EAD** William Victor Kendrick de Matos Silva **Pró-Reitor de Ensino de EAD** Janes Fidélis Tomelin **Presidente da Mantenedora** Cláudio Ferdinandi

NEAD - NÚCLEO DE EDUCAÇÃO A DISTÂNCIA

Diretoria Executiva Christiano Mincoff, James Prestes, Tiago Stachon **Diretoria de Design Educacional** Débora Leite **Diretoria de Graduação e Pós-graduação** Kátia Coelho **Diretoria de Permanência** Leonardo Spaine **Head de Curadoria e Inovação** Tania Cristiane Yoshiie Fukushima **Head de Produção de Conteúdo** Franklin Portela Correia **Gerência de Contratos e Operações** Jislaine Cristina da Silva **Gerência de Produção de Conteúdo** Diogo Ribeiro Garcia **Gerência de Projetos Especiais** Daniel Fuverki Hey **Supervisora de Projetos Especiais** Yasminn Talyta Tavares Zagonel **Supervisora de Produção de Conteúdo** Daniele C. Correia

FICHA CATALOGRÁFICA

Coordenador(a) de Conteúdo

Flávia Lumi Matuzawa

Projeto Gráfico e Capa

Arthur Cantarelli, Jhonny Coelho
e Thayla Guimarães

Editoração

Jean Nogueira

Design Educacional

Bárbara Neves

Revisão Textual

Ariane Andrade Fabreti

Ilustração

Natalia de Souza Scalassara

Fotos

Shutterstock

C397 CENTRO UNIVERSITÁRIO DE MARINGÁ.

Núcleo de Educação a Distância. **FREITAS**, Janaina Aparecida de.

Qualidade de Software.

Janaina Aparecida de Freitas.

Maringá - PR.: UniCesumar, 2021.

168 p.

"Graduação - EaD".

1. Qualidade 2. Software 3. Sistema. EaD. I. Título.

CDD - 22 ed. 005.1

CIP - NBR 12899 - AACR/2

ISBN 978-65-5615-251-6

Bibliotecário: João Vivaldo de Souza CRB- 9-1679



NEAD - Núcleo de Educação a Distância

Av. Guedner, 1610, Bloco 4Jd. Aclimação - Cep 87050-900 | Maringá - Paraná

www.unicesumar.edu.br | 0800 600 6360

Neste mundo globalizado e dinâmico, nós trabalhamos com princípios éticos e profissionalismo, não somente para oferecer educação de qualidade, como, acima de tudo, gerar a conversão integral das pessoas ao conhecimento. Baseamo-nos em 4 pilares: intelectual, profissional, emocional e espiritual.

Assim, iniciamos a Unicesumar em 1990, com dois cursos de graduação e 180 alunos. Hoje, temos mais de 100 mil estudantes espalhados em todo o Brasil, nos quatro campi presenciais (Maringá, Londrina, Curitiba e Ponta Grossa) e em mais de 500 polos de educação a distância espalhados por todos os estados do Brasil e, também, no exterior, com dezenas de cursos de graduação e pós-graduação. Por ano, produzimos e revisamos 500 livros e distribuímos mais de 500 mil exemplares. Somos reconhecidos pelo MEC como uma instituição de excelência, com IGC 4 por sete anos consecutivos e estamos entre os 10 maiores grupos educacionais do Brasil.

A rapidez do mundo moderno exige dos educadores soluções inteligentes para as necessidades de todos. Para continuar relevante, a instituição de educação precisa ter, pelo menos, três virtudes: inovação, coragem e compromisso com a qualidade. Por isso, desenvolvemos, para os cursos de Engenharia, metodologias ativas, as quais visam reunir o melhor do ensino presencial e a distância.

Reitor

Wilson de Matos Silva

Tudo isso para honrarmos a nossa missão, que é promover a educação de qualidade nas diferentes áreas do conhecimento, formando profissionais cidadãos que contribuam para o desenvolvimento de uma sociedade justa e solidária.



Professora Esp. Janaina Aparecida de Freitas

Possui graduação em Informática pela Universidade Estadual de Maringá (2010) e especialização em MBA em Teste de Software pela UNICEUMA (2012). Atualmente, cursa o Programa de Mestrado em Ciência da Computação na Universidade Estadual de Maringá (UEM) e é graduanda de Letras – Português/Inglês na Unicesumar. Atua como professora mediadora, professora conteudista em gravação de aulas ao vivo e em gravação de aulas conceituais nos cursos do NEAD – Núcleo de Educação a Distância da Unicesumar, para os cursos de graduação de Sistemas para Internet, Análise e Desenvolvimento de Sistemas, Gestão da Tecnologia da Informação e Engenharia de Software, nas disciplinas de Engenharia de Software, Design Gráfico, Tópicos Especiais, Gerenciamento de Software, Design de Interação Humano-Computador, Projeto Implementação e Teste de Software, há três anos. Além disso, tem experiência na iniciativa privada nas áreas de análise de sistemas e testes de software.

<http://lattes.cnpq.br/4906244382612830>

QUALIDADE DE SOFTWARE

Prezado(a) aluno(a), seja bem-vindo(a) à disciplina de Qualidade de Software. Este livro foi desenvolvido para você, que fica encantado(a) com softwares de qualidade. Com base nisso, procurei abordar assuntos interessantes e importantes que lhe ajudarão a entender como a qualidade é importante e como está presente no nosso dia a dia.

Na primeira unidade, abordaremos os conceitos relacionados à qualidade e iniciaremos apresentando-os junto com suas definições. É importante que você compreenda o que é o termo “qualidade”, porque é ele fundamental para a empresa e os clientes e como está presente no nosso dia a dia. Também falaremos sobre as definições de qualidade de produtos, assim como a importância dela nos processos de desenvolvimento do software.

Na segunda unidade, estudaremos os modelos de melhoria e avaliação de processo de software. Você compreenderá os principais modelos de melhoria e avaliação de processo de software e que a qualidade de um software está diretamente relacionada à qualidade do processo dele. Assim como o modelo de melhoria da maturidade de processo de software CMMI (Capability Maturity Model Integration), considerado um modelo que busca avaliar e melhorar a capacitação das empresas que desenvolvem um produto de software.

Depois, passaremos a compreender o modelo de qualidade de processo de software: modelo MPS.BR, o qual é voltado para a realidade do mercado de pequenas e médias empresas de desenvolvimento de software no Brasil. Seguindo para a norma de qualidade de software ISO/IEC 12207: processos de ciclo de vida de software, a qual estabelece uma estrutura comum para os processos desse ciclo e que tem, como objetivo principal, fornecer uma estrutura única para os envolvidos na produção de software.

Em seguida, compreenderemos a norma ISO/IEC 15504 – SPICE para a qualidade de software, sendo um padrão para a avaliação do processo de software e que visa determinar a capacitação de uma empresa bem como orientá-la para a melhoria contínua do processo citado.

Na terceira unidade, estudaremos os requisitos e a avaliação da qualidade de software. Você compreenderá os conceitos sobre o processo de avaliação da qualidade, o processo de avaliação da qualidade de produto de software e que, para uma avaliação mais efetiva, é importante utilizar modelos de qualidade que permitam estabelecer e avaliar requisitos de qualidade.

Aprenderemos os requisitos usados para pacotes de software e como são oferecidos e liberados para uso no mercado e, também, como é aplicada a norma NBR ISO/IEC 25051.

Veremos os requisitos de usabilidade na interface de software e como a norma ISO/IEC 92411-11 é usada para esclarecer os benefícios de medir a usabilidade quando pensamos em desempenho e satisfação do usuário.

Encerraremos com os requisitos usados para a documentação de usuário e como ele deve ser descrito e, também, como aplicar as normas ANSI/IEEE 1063 e a ISO 9127 para os requisitos mínimos. Finalmente, abordaremos as informações para usuários e compradores de software.

Na quarta unidade, começaremos a entender a importância das métricas de software para a qualidade do produto, por isso, iniciaremos expondo os principais conceitos relacionados a essas métricas. Também veremos a importância da revisão de software e como realizá-la, aplicando desde técnicas de inspeção até técnicas de pair programming.

Na última unidade, aprenderemos um pouco mais sobre como garantir a qualidade ao software que está sendo criado. Começaremos apresentando os principais conceitos usados na garantia da qualidade de software, em seguida, conhiceremos os padrões essenciais na construção do software, os quais, normalmente, são realizados por organizações de padronização, como a ISO (Organização Internacional de Padronização) e o IEEE (Instituto de Engenheiros Eletrotécnicos e Eletrônicos). Assim como as abordagens formais da SQA, em que são definidas a sintaxe e a semântica mais rigorosas para todas as linguagens de programação existentes e as criadas futuramente, e a realização de uma rigorosa abordagem da especificação dos requisitos de software.

Assim, convido você, caro(a) aluno(a), a entrar nesta jornada com empenho, dedicação e muita sede de conhecimento. Boa leitura e ótimos estudos!

ÍCONES



pensando juntos

Ao longo do livro, você será convidado(a) a refletir, questionar e transformar. Aproveite este momento!



explorando ideias

Neste elemento, você fará uma pausa para conhecer um pouco mais sobre o assunto em estudo e aprenderá novos conceitos.



quadro-resumo

No fim da unidade, o tema em estudo aparecerá de forma resumida para ajudar você a fixar e a memorizar melhor os conceitos aprendidos.



conceituando

Sabe aquela palavra ou aquele termo que você não conhece? Este elemento ajudará você a conceituá-la(o) melhor da maneira mais simples.



conecte-se

Enquanto estuda, você encontrará conteúdos relevantes online e aprenderá de maneira interativa usando a tecnologia a seu favor.



Quando identificar o ícone de QR-CODE, utilize o aplicativo Unicesumar Experience para ter acesso aos conteúdos online. O download do aplicativo está disponível nas plataformas:



Google Play



App Store

CONTEÚDO

PROGRAMÁTICO

UNIDADE 01

10

QUALIDADE DE
SOFTWARE

UNIDADE 02

40

MODELOS DE
MELHORIA E
AVALIAÇÃO DE
PROCESSOS DE
SOFTWARE

UNIDADE 03

75

REQUISITOS E
AVALIAÇÃO
DA QUALIDADE
DE SOFTWARE

UNIDADE 04

107

MÉTRICAS DE
SOFTWARE

UNIDADE 05

136

GARANTIA DA
QUALIDADE
DE SOFTWARE

FECHAMENTO

161

CONCLUSÃO GERAL



QUALIDADE DE SOFTWARE

PROFESSORA

Esp. Janaina Aparecida de Freitas



PLANO DE ESTUDO ▾

A seguir, apresentam-se as aulas que você estudará nesta unidade:

- Entender os conceitos de qualidade
- Compreender as definições de qualidade de produtos
- Entender a importância da qualidade nos processos de desenvolvimento do software
- Comparar a qualidade do processo x a qualidade do produto
- Compreender as definições de qualidade de software.

OBJETIVOS DE APRENDIZAGEM ▾

Conceitos de qualidade • Qualidade de produto • Qualidade de processo de software • Qualidade do processo x qualidade do produto • Qualidade de software.

INTRODUÇÃO



Caro(a) aluno(a), seja bem-vindo(a) ao estudo de qualidade de software. Esta unidade pretende que você compreenda os conceitos sobre qualidade e, então, iniciaremos apresentando tais conceitos e suas definições. Pretendemos que você compreenda o que é o termo “qualidade” e porque ele é importante para a empresa e os clientes e como está presente no nosso dia a dia.

Aprenderemos o que o termo “qualidade” significa e que ele, geralmente, é empregado para significar que um produto ou serviços tem excelência. Também significa que qualidade está relacionada com as necessidades de cada pessoa, podendo ser influenciada por fatores como cultura, tipo de produto ou de serviço prestado, percepções e necessidades de cada indivíduo.

Na sequência, compreenderemos as definições de qualidade de produtos. Aprenderemos que, nesta área, deverá ser avaliado o produto desenvolvido, analisando se o que foi produzido está de acordo com as necessidades explícitas e implícitas do cliente e que decorre da qualidade do processo utilizado em sua produção.

Entenderemos a importância da qualidade nos processos de desenvolvimento do software, compreenderemos que a qualidade do processo de desenvolvimento do software corresponde ao nível utilizado na implementação de um processo que seja aceitável e com qualidade e que esse processo inclui medições e critérios de qualidade.

Por fim, veremos as definições de qualidade de software, que é um conjunto de características cujo principal objetivo é garantir a conformidade de processos e um produto final que satisfaça às expectativas de seus usuários.



1 CONCEITOS DE QUALIDADE



A partir do momento que o software ficou cada vez mais integrado às atividades do dia a dia das empresas, a necessidade de produtos com qualidade se tornou essencial e um diferencial no mercado. A qualidade é considerada, hoje, um requisito muito importante em todas as áreas, pois todos querem produtos e serviços de excelência (GUERRA; COLOMBO, 2000).

Para iniciar, veremos a definição dada ao termo “qualidade”, pois ele pode variar de acordo com a abordagem utilizada. A seguir, algumas definições encontradas na literatura sobre o termo:

“Conformidade com as especificações”: definição abordada por Philip B. Crosby e que sugere que a qualidade deve ser verificada desde o início do desenvolvimento, para tentar evitar defeitos e diminuir o retrabalho (CROSBY, 1990).

“Adequação ao uso”: definição sugerida por Joseph M. Juran e que significa que as expectativas do cliente devem ser atendidas (JURAN; DEFEO, 2015).

“Qualidade é a totalidade das características de uma entidade que lhe confere a capacidade de satisfazer às necessidades explícitas e implícitas”: definição segundo a Norma Brasileira de Referência NBR ISO 8402 (ABNT, 1994), a qual diz que as necessidades explícitas são expressas na definição de requisitos que definem as condições em que o produto deve ser utilizado, seus objetivos, suas funções e qual será o desempenho esperado desse produto. As necessidades implícitas são necessárias para o usuário no manuseio do produto, no seu dia a dia, e não expressas em documentos. A entidade é o produto, o qual pode ser um bem ou um serviço (ABNT, 1994, p. 1).

O que é mais importante para uma empresa, em relação à qualidade, é que ela seja reconhecida pelo cliente, por meio da satisfação do produto que ele adquiriu. Para um produto com qualidade, devemos pensar nas necessidades do cliente, desde o início do desenvolvimento (SOMMERVILLE, 2011).

O conceito de qualidade e sua aplicabilidade vêm sendo cada vez mais difundidos nas empresas, sua busca faz com que o mercado fique cada vez mais competitivo e que as empresas passem a investir mais, mostrando que, em todas as etapas do desenvolvimento de seus produtos e serviços de software, é necessária a melhoria na qualidade, se elas quiserem sobreviver ou entrar neste mercado (PRESSMAN; MAXIM, 2016).

Como reconhecer se o produto ou serviço tem qualidade? A resposta a esta pergunta não é tão simples quanto se imagina, pois sabe-se o que é qualidade ao vê-la e, mesmo assim, é difícil de ser definida. De acordo com Pressman e Maxim (2016), temos cinco maneiras de identificar e defini-la:

Visão do usuário	Um produto atende às metas específicas de um usuário final, aquele que melhor atender às preferências do usuário.
Visão do fabricante	Conformidade com as especificações predefinidas no início do projeto. Se o produto atende às especificações originais do produto.
Visão do produto	Sugere que a qualidade pode ser ligada às funções e recursos de um produto. A qualidade poderá ser medida por meio de alguns atributos do produto. Mais qualidade - mais atributos - custos mais elevados.
Visão baseada em valor	Mede a qualidade tomando como base o quanto um cliente estaria disposto a pagar por um produto.
Transcendental	Sustenta que 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.

Quadro 1 - Maneiras de identificar e definir a qualidade

Fonte: adaptado de Pressman e Maxim (2016, p. 358).



A ideia de qualidade é, aparentemente, intuitiva; contudo, quando examinado mais longamente, o conceito se revela complexo.

Quando consideramos a qualidade do ponto de vista multidimensional, o qual, segundo Pressman e Maxim (2016), começa com uma avaliação da conformidade e termina com uma visão transcendental (estática), temos algumas dimensões que precisamos considerar. Elas são chamadas de Dimensões de Qualidade de Garvin (GARVIN, 1987 *apud* PRESSMAN; MAXIM, 2016).

Qualidade do desempenho	O software fornece todo o conteúdo, as funções e os recursos que são especificados como parte do modelo de requisitos, de forma a gerar valor ao usuário final?
Qualidade dos recursos	O software fornece recursos que surpreendem e encantam usuários finais que os utilizam pela primeira vez?
Confiabilidade	O software fornece todos os recursos e as capacidades sem falhas? Está disponível quando necessário? Fornece funcionalidade sem a ocorrência de erros?
Conformidade	O software está de acordo com os padrões de software locais e externos relacionados com a aplicação? Segue as convenções de projeto e codificação de fato? Por exemplo, a interface com o usuário está de acordo com as regras de projeto aceitas para a seleção de menus ou a entrada de dados?
Durabilidade	O software pode ser mantido (modificado) ou corrigido (depurado) sem a geração involuntária de efeitos colaterais indesejados? As mudanças farão com que a taxa de erros ou a confiabilidade diminuam com o passar do tempo?
Facilidade de manutenção	O software pode ser mantido (modificado) ou corrigido (depurado) em um período de tempo aceitável e curto? O pessoal de suporte pode obter todas as informações necessárias para realizar alterações ou corrigir defeitos?

Estética	Não há dúvida nenhuma de que cada um de nós tem uma visão diferente e muito subjetiva do que é estética. Mesmo assim, a maioria de nós concordaria que uma entidade estética tem certa elegância, um fluir único e uma “presença” que são difíceis de quantificar, mas que, não obstante, são evidentes. Um software estético possui estas características.
Percepção	Em algumas situações, temos alguns preconceitos que influenciarão nossa percepção de qualidade. Por exemplo, se for apresentado um produto de software construído por um fornecedor que, no passado, havia produzido software de má qualidade, ficaremos com nossa percepção de qualidade do produto de software influenciada negativamente. Similarmente, se um fornecedor tem uma excelente reputação, talvez percebamos qualidade, mesmo quando ela realmente não existe.

Quadro 2 - Dimensões da qualidade / Fonte: adaptado de Pressman e Maxim (2016, p. 358).

Algumas dessas dimensões podem ser consideradas apenas subjetivamente, e, por esta razão, precisamos de um conjunto de fatores de qualidade. Esses fatores são classificados em duas grandes categorias:

1. Fatores que podem ser medidos diretamente (por exemplo, defeitos revelados durante testes).
2. Fatores que podem ser medidos apenas indiretamente (por exemplo, usabilidade ou facilidade de manutenção).

Nas duas classificações, deve ocorrer a medição e a comparação do software com algum dado para chegar a uma indicação da qualidade.

Os fatores possuem uma categorização que mostram o que pode afetar a qualidade de software e possuem o foco em três importantes aspectos: características operacionais, habilidade de suportar mudanças e adaptabilidade a novos ambientes (Figura 1).

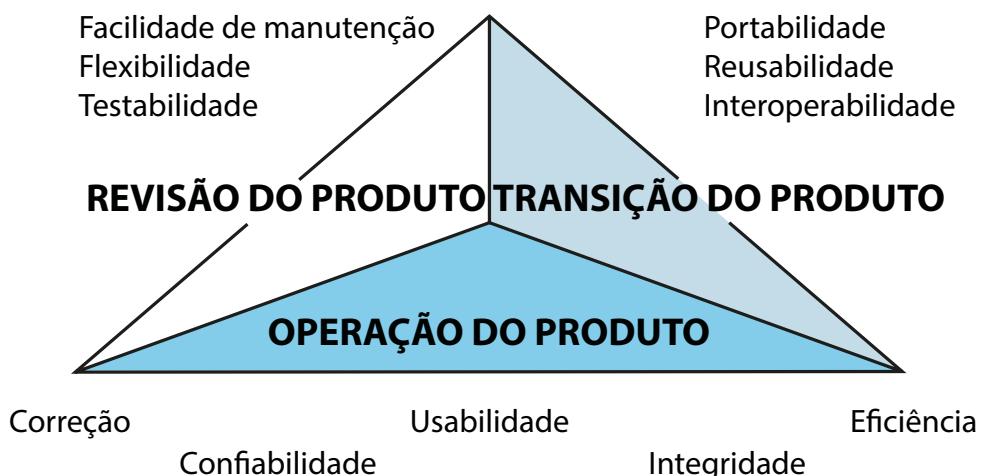


Figura 1 - Fatores de qualidade de software / Fonte: Pressman e Maxim (2016, p. 361).

A seguir, temos os fatores de qualidade de software e suas descrições (Quadro 3).

Fatores	Descrição
Correção	O quanto um programa satisfaz à sua especificação e atende aos objetivos da missão do cliente.
Confiabilidade	O quanto se pode esperar que um programa realize a função pretendida com a precisão exigida.
Eficiência	A quantidade de recursos computacionais e código exigidos por um programa para desempenhar sua função.
Integridade	O quanto o acesso ao software ou a dados por pessoas não autorizadas pode ser controlado.
Usabilidade	Esforço necessário para aprender, operar, preparar a entrada de dados e interpretar a saída de um programa.
Facilidade de manutenção	Esforço necessário para localizar e corrigir um erro em um programa.
Flexibilidade	Esforço necessário para modificar um programa em operação.

Testabilidade	Esforço necessário para testar um programa de modo a garantir que ele desempenhe a função destinada.
Portabilidade	Esforço necessário para transferir o programa de um ambiente de hardware e/ou software para outro.
Reusabilidade	O quanto um programa (ou partes de um programa) pode ser reutilizado em outras aplicações – relacionado com o empacotamento e o escopo das funções que o programa executa.
Interoperabilidade	Esforço necessário para integrar um sistema a outro.

Quadro 3 - Fatores de qualidade de software e suas descrições

Fonte: adaptado de Pressman e Maxim (2016, p. 361).

Mas será que conseguimos desenvolver medidas a todos os fatores de qualidade? Segundo Pressman e Maxim (2016), é difícil, e, em alguns casos, impossível, pois muitas das métricas podem ser medidas apenas indiretamente. Entretanto os fatores de qualidade ajudam a avaliar a qualidade de um produto que possibilitará a sólida indicação da qualidade de um software.



explorando Ideias

A qualidade depende do ponto de vista de quem a avalia, cujos usuários, desenvolvedores e organizações podem ter pontos de necessidades diferentes:

- Usuário: avalia o software sem conhecer seus aspectos internos, está apenas interessado na facilidade do uso, no desempenho, na confiabilidade dos resultados e no preço.
- Desenvolvedores: avaliam aspectos de conformidade em relação aos requisitos dos clientes e, também, aspectos internos do software.
- Organização: avalia aspectos de conformidade em relação aos requisitos dos clientes e desenvolvedores e, também, aspectos de custo e cronograma.

Fonte: adaptado de Andrade (2015, p. 13).



2 QUALIDADE DE PRODUTO

Para falarmos de qualidade de produto, é importante definirmos o que vem a ser um produto de software. De acordo com Andrade (2015, p. 15), “compreende os programas e procedimentos de computador e a documentação e dados associados, que foram projetados para serem liberados para o usuário”.

Como vimos anteriormente, existem diversas definições para o termo “qualidade”, como também existem diversas interpretações para qualidade de produto, tais como:

- Boa fabricação.
- Deve durar muito.
- Bom desempenho.
- Adaptável às necessidades específicas do cliente.
- Fácil de usar.
- Sem defeitos.

A especificação da qualidade de produto deve ser mais precisa e detalhada e deve decorrer da qualidade do processo utilizado em sua produção. A qualidade de produto deve estar em conformidade com requisitos determinados pelo cliente, que devem estar bem definidos para que seja possível gerenciá-lo de forma a reduzir o retrabalho e aumentar a produtividade da empresa de software (ANDRADE, 2015).

Segundo Pressman e Maxim (2016, p. 24), o padrão ISO 9126 “foi desenvolvido como uma tentativa de identificar os atributos fundamentais de qualidade para software de computador”. Para que um produto de software seja considerado um produto de qualidade, de acordo com a norma ISO/IEC 9126, deve possuir os seguintes atributos fundamentais de qualidade:

Funcionalidade	É a capacidade do software de prover funcionalidades que satisfaçam o usuário em suas necessidades explícitas e implícitas, dentro de um determinado contexto de uso. Satisfaz às necessidades?
Confiabilidade	O produto se mantém no nível de desempenho nas condições estabelecidas. Imune às falhas, sem defeitos?
Usabilidade	A capacidade de o software ser compreendido, seu funcionamento aprendido, ser operado e atraente ao usuário. Fácil de usar?
Eficiência	O tempo de execução e os recursos envolvidos são compatíveis com o nível de desempenho do software. É rápido e enxuto? Tem bom desempenho?
Manutenibilidade	A capacidade do produto de software de ser modificado, incluindo as melhorias ou extensões de funcionalidade quanto às correções de defeitos, falhas ou erros. É fácil de modificar?
Portabilidade	A capacidade de o sistema ser transferido de um ambiente para outro. É possível utilizar em várias plataformas?

Quadro 4 - Atributos fundamentais da qualidade de produto
Fonte: adaptado de 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 esta 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.

Do ponto de vista de quem produz, qualidade é o encontro com os requirementos e especificações, é o fim ou a meta (GUERRA; COLOMBO, 2000). E a melhor forma de alcançar a qualidade de produto é introduzi-la ao longo do processo produtivo, desde sua concepção até a entrega do produto final (SOMMERVILLE, 2011).



explorando Ideias

Se produzimos 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 e Maxim (2016, p. 365).



QUALIDADE DE PROCESSO DE SOFTWARE

Segundo Pressman e Maxim (2016), podemos definir processo de software como um conjunto de atividades, ações e tarefas que são realizadas na construção de um produto.

O termo processo de software também possui várias definições. A seguir, algumas delas bastante utilizadas na literatura:

- A ISO e a IEC (ABNT, 2018, p. 2): definem processo de software como: "Conjunto de Processos usados por uma organização para: planejar, gerenciar, executar, monitorar e melhorar as atividades relacionadas ao Software".
- Para Pressman e Maxim (2016, p. 458), processo de software consiste em 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".

O estudo da qualidade do processo de software é uma área relacionada diretamente à Engenharia de Software, sendo que o estudo de uma complementa o entendimento da outra. Nas duas disciplinas, os modelos do processo de desenvolvimento de softwares são estudados. Os modelos são considerados uma forma de explicar, em detalhes, como se desenvolve um software e as etapas envolvidas nesse processo.

Segundo Sommerville (2011, p. 473), os padrões de produto são os que se aplicam ao produto de software em desenvolvimento, incluindo padrões de do-

cumentos, como: “(i) a estrutura do documento de requisitos a ser produzido; (ii) padrões de documentação, como um cabeçalho-padrão de comentário para uma definição de classe de objeto e (iii) padrões de codificação, que definem como uma linguagem de programação deve ser utilizada”.

Os padrões de processos definem os procedimentos que devem ser seguidos durante o ciclo de desenvolvimento de software. Um processo de software bem definido é muito importante, porque, a partir dele, podemos estabelecer um plano para o desenvolvimento do projeto.

Aplicando a qualidade de processo de software, temos o aumento da qualidade do produto e a redução de retrabalho e, com isso, obtemos mais produtividade e diminuímos o tempo de desenvolvimento (SOMMERVILLE, 2011). Conforme Sommerville (2011, p. 472), “a melhoria do 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).

Mesmo com processos de software bem definidos, controlados e de boa qualidade, é indispensável que sejam considerados aspectos e mecanismos próprios de avaliação da qualidade dos produtos de software, resultantes do processo de desenvolvimento.

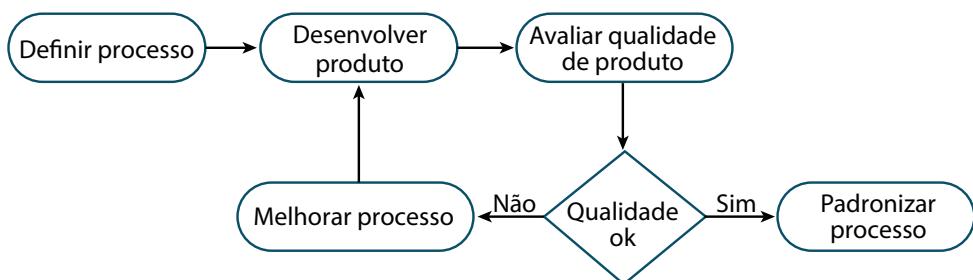


Figura 2 - Qualidade baseada em processos / Fonte: Sommerville (2011, p. 472).

Na Figura 2, mostramos a abordagem baseada em processos para atingir a qualidade de produto. Você mede esta qualidade e altera o processo até atingir o nível de qualidade requerida.



Quando estamos diante do desafio de garantir a qualidade de um software, estamos, na verdade, estabelecendo uma cultura de não-tolerância a erros, ou seja, estamos estruturando processos que possuam mecanismos de inibição e impedimento de falhas, possibilitando que os diversos artefatos gerados durante o ciclo de desenvolvimento tenham procedimentos que avaliam sua qualidade, possibilitando a identificação prematura de defeitos nesses artefatos. Todas as atividades cujo foco principal é garantir a qualidade de cada etapa do processo de engenharia de software devem ser consideradas dentro da dimensão da garantia da qualidade do processo de engenharia de software.

Fonte: Pressman e Maxim (2016).



Conforme Andrade (2015), um dos pontos mais importantes no estudo da qualidade está em perceber que a qualidade de um produto de software é resultante das atividades realizadas durante seu processo de desenvolvimento. Assim, podemos dizer que a qualidade do produto de software é necessária para a empresa, mas que a qualidade do processo de desenvolvimento do software é ainda mais necessária e importante para os negócios (Figura 3).

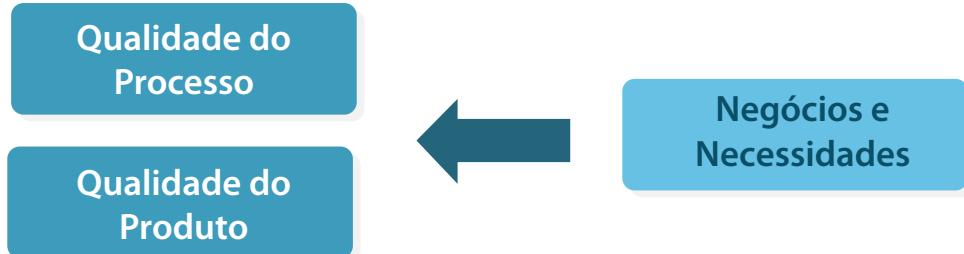


Figura 3 - Qualidade do processo e do produto / Fonte: a autora.

Esta descoberta que envolve a qualidade do produto e do processo surgiu durante a própria evolução dos conceitos de qualidade, ao longo dos anos. Sendo que, hoje, podemos consultar normas e padrões tanto para a qualidade do produto quanto para a qualidade do processo do software (GUERRA; COLOMBO, 2000).

Os certificados mais importantes para as empresas são aqueles que certificam o processo de produção de um software e não aqueles que certificam o software final, somente. Entretanto é normal encontrar empresas que procuram conseguir a certificação para os dois tipos de qualidade, pois, além do aumento da produtividade e da diminuição do retrabalho, elas buscam melhor relacionamento com seus clientes bem como melhor planejamento e gerenciamento das atividades de desenvolvimento e, com isso, levar à diminuição do número de defeitos dos produtos de softwares (Figura 4).



Figura 4 - Qualidade do processo de desenvolvimento x qualidade do produto de software
Fonte: a autora.

Segundo Pressman e Maxim (2016), um software tem qualidade se ele estiver adequado à empresa, ao cliente ou usuário e atenda às normas e aos padrões de qualidade que foram predefinidos durante seu desenvolvimento. Mesmo quando o software estiver pronto, ele terá qualidade se gerar dados e informações com qualidade, também.

Não há dúvida de que o processo de desenvolvimento usado tenha uma influência significativa sobre a qualidade de software e que bons processos são mais suscetíveis de conduzir o software de boa qualidade. O gerenciamento e a melhoria de qualidade de processo podem gerar softwares com menos defeitos. Contudo, é difícil avaliar os atributos de qualidade de software, como manutenibilidade, sem usar o software por um longo período. Portanto, é difícil dizer como as características de processo influenciam esses atributos. Além disso, por causa do papel do projeto e da criatividade no processo de software, a padronização de processos pode, por vezes, sufocar a criatividade, o que pode gerar softwares com menos qualidade (SOMMERVILLE, 2011, p. 458).

Quando se desenvolve um software, a relação entre a qualidade de processo e de produto é mais complexa. Conforme Sommerville (2011, p. 458), o desenvolvimento de software “é um processo criativo, em vez de mecânico, portanto, a influência de competências e experiências individuais é significativa”. Os fatores externos, como a novidade de uma aplicação ou a pressão do mercado, podem afetar, também, a qualidade de produto, independentemente do processo usado.

Qualidade do processo	Qualidade de produto
Deve garantir que o projeto realize todos os processos necessários para que possa atender aos requisitos.	Descobrir os defeitos gerados ao longo do projeto e eliminar suas causas.
Auditar os processos de acordo com os padrões e procedimentos previamente estabelecidos.	Por meio de testes e revisões.
Utilizando métodos e procedimentos para comparar o que foi realizado com o que deve ser feito.	Utilizando checklists, casos de uso de testes comparando o que foi feito pelo que é esperado pelo cliente.
Visa à prevenção de falhas.	Visa à detecção e correção de defeitos.

Deve assegurar que o processo é apropriado.	Deve assegurar que os produtos desenvolvidos estejam consistentes e alinhados.
Cuida da melhoria dos processos e padrões empregados.	Cuida da melhoria dos processos e padrões empregados.

Quadro 5 - Qualidade do processo x qualidade do produto

Fonte: adaptado de Sommerville (2011).

A qualidade do produto está diretamente relacionada à qualidade do processo de desenvolvimento. Segundo Sommerville (2011, p. 458), para que “o processo de desenvolvimento do software tenha sucesso, é necessário ter um bom gerenciamento do projeto, a fim de garantir que todo o processo seja executado de forma correta por seus envolvidos”. Para que esse sucesso seja alcançado, foram criadas várias normas e vários modelos de padronização para auxiliar no gerenciamento e na qualidade do software.



explorando Ideias

Um processo de software envolve um grande conjunto de elementos, tais como objetivos organizacionais, políticas, pessoas, comprometimentos, ferramentas, métodos, atividades de apoio e tarefas da engenharia de software. Para que o processo de software seja eficiente, ele precisa ser constantemente avaliado, medido e controlado. Quando as empresas não possuem conhecimento suficiente de todos os elementos do processo, essas atividades de avaliar, medir e controlar ficam difíceis de serem realizadas, neste caso, o processo de software passa a ser descontrolado, sem gerência e, até mesmo, caótico.

Algumas características de processos de softwares caóticos são: o processo é improvisado (profissionais e gerentes); o processo não é rigorosamente seguido e o cumprimento do mesmo não é controlado; o processo é altamente dependente dos profissionais atuais; a visão do progresso e da qualidade do processo é baixa; a qualidade do produto decorrente do processo é comprometida em função de prazos; quando são impostas datas urgentes para entrega dos produtos, frequentemente, a funcionalidade e a qualidade dos mesmos são comprometidas para atender ao cronograma. Não existe nenhuma base objetiva para julgar a qualidade do produto ou para resolver problemas de processo ou de produto, portanto, a qualidade do produto é difícil de ser prevista; as atividades ligadas à melhoria da qualidade, tais como revisões e testes, frequentemente são encurtadas ou eliminadas quando os projetos ultrapassam o cronograma previsto.

Fonte: Andrade (2015, p. 16).

QUALIDADE DE SOFTWARE



O termo qualidade de software também possui várias definições. A seguir, algumas que são utilizadas na literatura:

“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” (ROCHA *et al.*, 2005, p. 45).

“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, p. 414).

O objetivo de qualquer empresa de software é produzir software com qualidade, nos prazos estabelecidos e que obtenha a satisfação do cliente. Para isso, devemos produzir o software conforme os requisitos que foram levantados e desenvolvê-lo nos prazos estabelecidos e com um nível de defeitos aceitável, para a satisfação do cliente (FREITAS, 2019, p. 52).

Segundo Freitas (2019, p. 52), “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”. A cada dia, há o aumento do número de empresas que buscam melhorias para seus processos de desenvolvimento de software. Procuram o crescimento da produtividade, a diminuição do retrabalho e a melhor comunicação com seus clientes, por

meio do planejamento e da gestão de suas tarefas de desenvolvimento e da diminuição no número de erros e defeitos nos produtos entregues (PRESSMAN; MAXIM, 2016).

De acordo com Pressman e Maxim (2016), temos, basicamente, dois tipos de qualidade: **interna** e **externa**. A qualidade **interna** de um software é uma avaliação de seus componentes estruturais, sua arquitetura, seu estilo de codificação, sua adequação a padrões e métricas estabelecidas. A qualidade **externa** está mais ligada à funcionalidade do software, à facilidade de uso, ao tempo de resposta e à precisão de resultados. Normalmente, a boa qualidade interna leva à boa qualidade externa, mas nem sempre isto ocorre.

Dessa forma, programas de melhorias de qualidade, quando bem executados e bem-sucedidos, ajudam no aumento da produtividade, propiciam a redução no número de erros e defeitos no produto de software, atingem o cumprimento das metas estabelecidas de prazo e o crescimento da motivação da equipe desenvolvedora. Qual modelo ou norma deve ser aplicado? Isto dependerá de cada empresa, de suas estratégias de negócio e de mercado, considerando os benefícios e retornos sobre o investimento realizado. Como toda norma e qualquer modelo, pelo fato de estes serem idealizados com base em diversas considerações e práticas, eles não serão perfeitos e completos. As diversas normas e os 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, 2011).

O gerenciamento de qualidade de software para sistemas de software tem algumas preocupações principais:

Nível organizacional	O gerenciamento de qualidade está preocupado com o estabelecimento de um framework de processos organizacionais e padrões que levem a softwares de alta qualidade.
Nível de projeto	O gerenciamento de qualidade envolve a aplicação de processos específicos, verificando que os processos planejados foram seguidos e a garantia de que as saídas de projeto estejam em conformidade com os padrões aplicáveis ao projeto. No nível de projeto, o gerenciamento de qualidade também está preocupado com o estabelecimento de um plano de qualidade. O plano deve definir as metas de qualidade para o projeto e quais processos e padrões devem ser usados.

Quadro 6 - Principais preocupações do gerenciamento de qualidade de software
Fonte: adaptado de Sommerville (2011, p. 454).



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

(Roger Pressman e Bruce Maxim)

A qualidade de software é considerada o resultado do bom gerenciamento de projeto e uma prática consistente de engenharia de software. Conforme Pressman e Maxim (2016, p. 459), o “gerenciamento e a prática são aplicados no contexto de quatro grandes atividades que ajudam uma equipe de software a atingir alto padrão de qualidade de software”. As atividades são: (i) métodos de engenharia de software; (ii) técnicas de gerenciamento de projeto; (iii) ações de controle de qualidade e (iv) garantia da qualidade de software.

Métodos de engenharia de software	Existe uma ampla gama de conceitos e métodos capazes de levar a um entendimento relativamente completo do problema e a um projeto abrangente que estabeleça uma base sólida para a atividade de construção. Se aplicarmos esses conceitos e adotarmos os métodos de análise e projeto apropriados, a probabilidade de criarmos software de alta qualidade aumentará substancialmente.
Técnicas de gerenciamento de projeto	O plano de projeto deve incluir técnicas explícitas para o gerenciamento de mudanças e qualidade. Técnicas que levam a práticas ótimas de gerenciamento de projeto.
Ações de controle de qualidade	O controle de qualidade engloba um conjunto de ações de engenharia de software que ajudam a garantir que cada produto resultante atinja suas metas de qualidade. Os modelos são revistos de modo a garantir que sejam completos e consistentes. O código poderia ser inspecionado de modo a revelar e a corrigir erros antes de os testes começarem.

Garantia da qualidade de software	A garantia da qualidade estabelece a infraestrutura que suporta métodos sólidos de engenharia de software, gerenciamento racional de projeto e ações de controle de qualidade – todos fundamentais para a construção de software de alta qualidade. Além disso, a garantia da qualidade consiste em um conjunto de funções de auditoria e de relatórios que possibilita uma avaliação da efetividade e da completude das ações de controle de qualidade.
-----------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Quadro 7 - Atividades da garantia de qualidade de software

Fonte: Pressman e Maxim (2016, p. 371).

O objetivo principal da garantia da qualidade é fornecer à equipe técnica e administrativa os dados necessários para serem informados sobre a qualidade do produto.

De acordo com Sommerville (2011, p.455), o “gerenciamento de qualidade fornece uma verificação independente do processo de desenvolvimento de software”. O processo de gerenciamento de qualidade procura verificar os entregáveis de projeto para garantir que eles sejam consistentes com os padrões e objetivos da empresa.

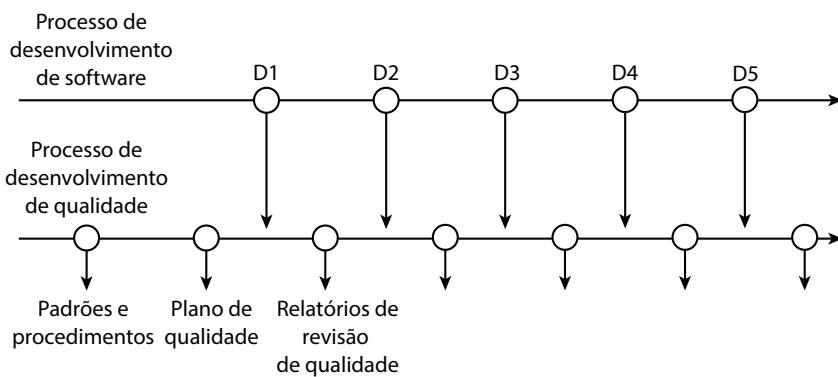


Figura 5 - Gerenciamento de qualidade e desenvolvimento de software

Fonte: Pressman e Maxim (2016, p. 455).

O gerenciamento de qualidade e desenvolvimento de software está, inevitavelmente, interligado a pessoas com responsabilidades de desenvolvimento e de qualidade.



explorando Ideias

O planejamento de qualidade é o processo de desenvolvimento de um plano de qualidade para um projeto. O plano deve estabelecer as qualidades desejadas para o software e descrever como elas devem ser avaliadas. Portanto, define o que o software de “alta qualidade” realmente significa para determinado sistema. Sem esta definição, os engenheiros podem fazer suposições, algumas vezes, conflitantes sobre quais atributos de produto refletem as mais importantes características de qualidade. O planejamento formal de qualidade é parte integrante dos processos de desenvolvimento baseados em planos, no entanto métodos ágeis adotam uma abordagem menos formal para o gerenciamento de qualidade.

Fonte: adaptado de Pressman e Maxim (2016, p. 456).



CONSIDERAÇÕES FINAIS

Caro(a) aluno(a), chegamos ao fim da nossa unidade, 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.

Vimos o que o termo significa e como ele é usado para indicar que um produto ou serviço tem excelência e está de acordo com o que foi definido em suas especificações. Aprendemos que a qualidade de algo está relacionada às necessidades de cada indivíduo e que muitos fatores podem influenciar, de forma diferente, dependendo da cultura, do tipo de produto ou serviço prestado, as percepções e necessidades de cada pessoa.

Na sequência da unidade, foram mostradas as definições de qualidade de produto. Aprendemos que ela deve ser avaliada conforme o que foi desenvolvido e se o que foi produzido está de acordo com as necessidades do cliente. Compreendemos a qualidade do processo e como ela é importante durante o desenvolvimento do software, e que o processo corresponde ao nível utilizado na implementação de um procedimento aceitável. A qualidade do processo, na realização dos produtos de software, inclui medições e critérios de qualidade.

E, por fim, compreendemos as definições usadas na qualidade de software e que ela é um conjunto de características cujo principal objetivo é garantir a conformidade de processos e um produto final que satisfaça às expectativas de seus clientes.



1. Os problemas com a qualidade de software foram inicialmente descobertos na década de 60, com o desenvolvimento do primeiro grande sistema de software, e continuaram a incomodar a engenharia de software ao longo do século XX. O software entregue era lento e pouco confiável, difícil de manter e de reusar. Em resposta à insatisfação com aquela situação, passaram a ser adotadas técnicas formais de gerenciamento de qualidade do software, as quais foram desenvolvidas a partir dos métodos usados na indústria manufatureira (SOMMERVILLE, 2011).

Considerando o texto apresentado, assinale a alternativa correta sobre qualidade de software:

- a) A qualidade de software tem, como principal objetivo, medir o tempo de resposta no processamento das informações.
 - b) A qualidade de software é um conjunto de características cujo principal objetivo é garantir um produto final que satisfaça às expectativas dos usuários.
 - c) A qualidade de software tem, como objetivo, procurar por erros e falhas no software após a entrega ao cliente.
 - d) A qualidade de software é um conjunto de atributos que interfere na capacidade do software de processar as informações.
 - e) A qualidade de software busca assegurar que o produto satisfaça às necessidades do cliente e envolve a atividade de manutenção do software.
2. Na indústria de software, diferentes empresas e setores industriais interpretam a garantia de qualidade e controle de qualidade de maneiras diferentes. Às vezes, garantia de qualidade significa, simplesmente, a definição de procedimentos, processos e padrões que visam reforçar que a qualidade de software seja atingida. Em outros casos, a garantia de qualidade também inclui todo o gerenciamento de configuração, atividades de verificação e validação aplicadas após o produto ter sido entregue por uma equipe de desenvolvimento (SOMMERVILLE, 2011).

Pensando sobre o texto apresentado, assinale a alternativa correta sobre qualidade de produto:



- a) É estar em conformidade com processos de software.
 - b) Procura assegurar que o projeto será concluído com a qualidade que o cliente deseja, satisfazendo às suas necessidades.
 - c) É a definição e a aplicação de processos de desenvolvimento na sua construção.
 - d) Avalia se o software se comporta ou não conforme o especificado pelo usuário.
 - e) É estar em conformidade com requisitos solicitados.
3. "Hoje em dia, a qualidade de software continua a ser um problema, mas a quem culpar? Os clientes culpam os desenvolvedores, argumentando que práticas des-cuidadas levam a um software de baixa qualidade. Os desenvolvedores de software culpam os clientes (e outros interessados), argumentando que datas de entrega absurdas e um fluxo contínuo de mudanças os forçam a entregar software antes de eles estarem completamente validados. Quem está com a razão? Ambos – e esse é o problema" (PRESSMAN; MAXIM, 2016, p. 412).

Com base no texto exposto, assinale a alternativa correta sobre qualidade de processo:

- a) Garante a conformidade dos processos de desenvolvimento com o objetivo de prevenir e eliminar os defeitos do produto de software.
- b) Serve para avaliar as conformidades determinadas pela empresa, por meio de seus processos, e, com isso, garantir ao cliente um material, processo, produto ou serviço concebido conforme padrões, procedimentos e normas.
- c) É uma atividade essencial na produção do software, pois ela afeta todas as fases do projeto e age de forma decisiva no sucesso com o cliente.
- d) Serve para tomar decisões de como desenvolver e implementar os requisitos do sistema, como as ferramentas e linguagens usadas.
- e) Procura mostrar e estabelecer a confiança de que o sistema está sendo imple-mentado de acordo com seu propósito.



4. “Até mesmo os desenvolvedores de software mais experientes concordam que software de alta qualidade é um objetivo importante. Mas como definir a qualidade de software? No sentido mais geral, a qualidade de software pode ser definida 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” (PRESSMAN; MAXIM, 2016, p. 415).

Considerando fragmento de texto, compare qualidade do produto x qualidade de processo.

5. “A determinação da qualidade é um fator-chave nos eventos do dia a dia – concursos de degustação de vinhos, eventos esportivos (por exemplo, ginástica), concursos de talentos etc. Nessas situações, a qualidade é julgada da forma mais fundamental e direta: comparação entre dois objetos sob condições idênticas e com conceitos predeterminados” (PRESSMAN; MAXIM, 2016, p. 416).

Considerando o texto apresentado, descreva a classificação dos fatores de qualidade.



UMA ONTOLOGIA DE QUALIDADE DE SOFTWARE

O primeiro passo em direção à qualidade de software consiste em entender seus conceitos para poder aplicá-los consistentemente. Infelizmente, não há consenso sobre a terminologia usada, o que provoca vários problemas, principalmente de interpretação, quando da definição de um programa de qualidade.

Na medida em que cresce a demanda por sistemas complexos, com grande responsabilidade no contexto das organizações, a qualidade desporta como um fator essencial no desenvolvimento de software. Sendo assim, cada vez mais, há uma disposição para se investir em qualidade. Contudo, uma das primeiras dificuldades encontradas na definição e implantação de um programa de qualidade está em compreender o que, de fato, significa qualidade de software.

No contexto de desenvolvimento de software, qualidade pode ser entendida como um conjunto de características a serem satisfeitas em um determinado grau, de modo que o produto de software atenda às necessidades explícitas e implícitas de seus usuários. Entretanto, não se obtém qualidade do produto de forma espontânea. Ela tem de ser construída. Assim, a qualidade do produto depende fortemente da qualidade de seu processo de desenvolvimento. Neste trabalho, contudo, não discutimos o processo de software em si, mas sim nos apoiamos na ontologia de processo de software desenvolvida. Para avaliar a qualidade, é preciso haver meios de medi-la. Ou seja, é preciso obter uma medida que quantifique o grau de alcance de uma característica de qualidade. Assim, para computar uma característica de qualidade, é necessário estabelecer uma métrica capaz de quantificá-la e fazer uma medição para determinar a medida, resultado da aplicação da métrica. Por exemplo, digamos que se deseja saber o tamanho de determinado produto de software. Então, a métrica utilizada poderia ser “número de linhas de código”. A medição po-



deria ser “contar o número de linhas de código, desconsiderando comentários, de cada programa contido no software”. E a medida seria o número obtido na medição, uma quantidade, por exemplo, “5000 linhas de código”. Muitas vezes, características de qualidade não podem ser medidas diretamente através de métricas, sendo necessário decompor-as em sub-características. Assim, há características que são diretamente mensuráveis e outras que são apenas indiretamente mensuráveis. Uma característica diretamente mensurável possui uma métrica correlacionada, à qual, através de uma medição, se aplica uma medida. As características indiretamente mensuráveis podem ser descritas em termos de outras características, sendo calculada, em última instância, via características diretamente mensuráveis. Desta forma, uma certa característica indiretamente mensurável pode ser computada de diferentes maneiras, em função das sub-características escolhidas, já que essa escolha é, muitas vezes, função da fase do processo ou artefato para o qual as sub-características estão sendo aplicadas, do paradigma ou da tecnologia de desenvolvimento. Em outras palavras, diferentes métricas podem ser aplicadas, dependendo da atividade do ciclo de vida ou do artefato ao qual se aplica. Em todos os casos, contudo, há de se respeitar a pertinência em relação ao paradigma e à tecnologia de desenvolvimento adotados no desenvolvimento. Processos possuem características de qualidade próprias, tais como desempenho, estabilidade e capacidade, mas informações sobre a qualidade do produto gerado podem ser também importantes na avaliação do processo, uma vez que a baixa qualidade do produto pode ser um reflexo de falha ou inadequação do processo utilizado. Assim, características de qualidade do processo podem ser computadas a partir de características de qualidade do produto. O contrário, contudo, não é possível. Ou seja, características de qualidade do produto só podem ser computadas a partir de outras características de produto.

Fonte: Duarte e Farbo (2006).



eu recomendo!



livro

Garantia da Qualidade de Software

Autor: Alexandre Bartie

Editora: Campus

Sinopse: totalmente alinhado com as mais modernas metodologias existentes no mercado (RUP – Rational Unified Process; CMM – Capability Maturity Model; SWEBOK – Software Engineering Body of Knowledge e PMI – Project Management Institute), este livro coloca você diante dos conceitos mais avançados sobre como aplicar um processo de garantia da qualidade de software na sua empresa. Usando uma abordagem simplificada e de fácil entendimento, possibilita aos leitores assimilar, gradualmente, os aspectos mais relevantes envolvidos na implantação de um processo de garantia da qualidade de software.



livro

Introdução à garantia de qualidade de software

Autor: Luiz Diego Vidal Santos e Catuxé Varjão de Santana Oliveira

Editora: Cia do eBook

Sinopse: este livro aborda tópicos como: as definições de garantia de qualidade de software, tarefas de verificação e validação, mais especificamente, testes de software, fases do teste de software, gerenciamento de mudanças, depuração e atores envolvidos. Além disso, mostra algumas ferramentas importantes que podem auxiliar o testador na tarefa de produzir software com qualidade, entre outras várias técnicas.





eu recomendo!



conecte-se

Qualidade, qualidade de software e garantia da qualidade de software são as mesmas coisas?

Este artigo tem, como objetivo, mostrar as diferenças entre os termos “qualidade”, “qualidade de software” e “garantia da qualidade de software”. Assim, podemos esclarecer a diferença e, até mesmo, alguns relacionamentos entre estes três termos.

<http://www.linhadecodigo.com.br/artigo/1712/qualidade-qualidade-de-software-e-garantia-da-qualidade-de-software-sao-as-mesmas-coisas.aspx>

Como você define qualidade de software?

Artigo que mostra como a qualidade é a base para o bom relacionamento com o cliente, os usuários e as equipes de desenvolvimento. Um software que não atende aos três stakeholders terá vida curta e será responsável por muitas dores de cabeça.

<https://www.tiespecialistas.com.br/como-voce-define-qualidade-de-software/>



MODELOS DE MELHORIA E AVALIAÇÃO DE processos de software

PROFESSORA

Esp. Janaina Aparecida de Freitas

PLANO DE ESTUDO ▾

A seguir, apresentam-se as aulas que você estudará nesta unidade:

- Melhoria de processo de software
- Modelo de melhoria da maturidade do processo de software: CMMI
- Modelo de qualidade de processo de software: modelo MPS.BR
- Normas de qualidade de software: norma ISO/IEC 12207
- Norma ISO/IEC 15504 – SPICE

OBJETIVOS DE APRENDIZAGEM ▾

Entender os principais modelos de melhoria e avaliação de processo de software

- Entender o modelo de melhoria da maturidade do processo de software: CMMI (Capability Maturity Model Integration)
- Compreender o modelo de qualidade de processo de software: modelo MPS.BR
- Entender a norma de qualidade de software ISO/IEC 12207: processos de ciclo de vida de software
- Compreender a norma ISO/IEC 15504 – SPICE para a qualidade de software.

INTRODUÇÃO



Caro(a) aluno(a), seja bem-vindo(a) à nossa segunda unidade, em que estudaremos os modelos de melhoria e avaliação de processo de software. Esta unidade pretende que você compreenda os principais modelos de melhoria e avaliação de processo e que a qualidade de um software está diretamente relacionada à qualidade do seu processo.

Entenderemos o modelo de melhoria da maturidade de processo de software CMMI (Capability Maturity Model Integration), considerado um modelo que busca avaliar e melhorar a capacitação das empresas que desenvolvem um produto de software.

Na sequência, passaremos a compreender o modelo de qualidade de processo de software: o modelo MPS.BR, o qual é voltado para a realidade do mercado de pequenas e médias empresas de desenvolvimento de software no Brasil.

Aprenderemos a norma de qualidade de software ISO/IEC 12207: processos de ciclo de vida de software, que estabelece uma estrutura comum para tais processos e tem, como objetivo principal, fornecer uma estrutura única para os envolvidos na produção de software.

Por fim, compreenderemos a norma ISO/IEC 15504 – SPICE para a qualidade de software, sendo um padrão para a avaliação do processo, visando determinar a capacitação de uma empresa e, também, a orientá-la para a melhoria contínua do processo de software.



1

MELHORIA DE PROCESSO DE SOFTWARE



Na unidade anterior, falamos de como é importante termos um produto de software com qualidade, mas isto é uma tarefa complexa e que envolve muitos fatores que devem ser medidos, avaliados e melhorados continuamente. Aprendemos, também, que a qualidade de um software está diretamente relacionada à qualidade do seu processo.

Para começar, relembraremos algumas características importantes para ter um processo de software maduro e eficiente.

O processo deve ter uma execução consistente, sempre que for necessária.

O processo deve ser flexível durante o processo de execução para a melhor adaptação das necessidades específicas.

O processo dever ser documentado com uma notação representativa (identificação por meio de texto, figuras, fluxos etc.).

O processo deve ser apropriado para que os usuários possam realizar as suas tarefas.

O processo deve ter treinamento para os usuários que realizam as atividades, de forma a obterem conhecimento, habilidade e experiência.

O processo deve ter manutenção para garantir a sua evolução contínua.

O processo deve ter controle de mudanças para garantir a integridade e disponibilidade dos artefatos produzidos.

O processo deve apoiar equipes, ter ferramentas e recursos apropriados para a realização.

Quadro 1 - Algumas características para um processo de software eficiente / Fonte: adaptado de Andrade (2015).

As empresas que possuem um processo de software de alta qualidade têm políticas, padrões e estruturas organizacionais para esse processo, e isso acarreta, para a empresa, construir uma:

- Infraestrutura com processos eficazes, que sejam utilizáveis e consistentes e que possam ser aplicados em toda a empresa.
- Cultura corporativa que usa os métodos, os procedimentos e as práticas dos negócios para que perdurem na empresa.

A melhoria de processo de software, segundo a ISO 9000-3, consiste na abordagem de uma prática com ações que sejam orientadas para a alteração dos processos aplicados à aquisição, ao fornecimento, ao desenvolvimento e, também, à manutenção ou ao suporte de sistemas de software (ANDRADE, 2015).

Qual motivação uma empresa de software tem para tentar melhorar seu processo? A forte competição do negócio, a regulação externa e a forte necessidade de melhorar o desempenho de sua empresa e de seus produtos de software desenvolvidos. E o que melhorar? A empresa deve pensar em ter um conjunto de melhores práticas para o seu processo de software (Figura 1).

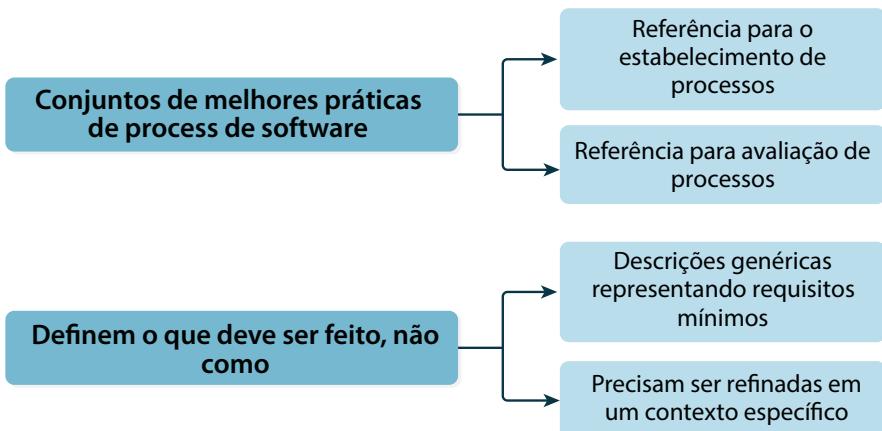


Figura 1 - O que melhorar? / Fonte: a autora.

O que usar para melhorar? O considerável interesse das empresas pela melhoria dos seus processos, nas últimas décadas, motivou o surgimento de várias normas e de vários modelos de referência. Muitas dessas normas e muitos desses modelos podem ser adotados pelas empresas como base para que implantem padrões de melhorias em seus processos de software.

A implantação da melhoria do processo é considerada uma atividade complexa e intensa para a empresa. Os envolvidos nas iniciativas devem: (i) possuir conhecimento sobre engenharia de software (ii); serem capazes de orientar a implantação das melhorias nos processos da empresa; (iii) serem capazes de aumentar as chances de um resultado positivo.

A principal chave para a garantia da qualidade é adotar normas e modelos de padrões de qualidade de software, pois eles definem características que todos os componentes do software devem possuir e como o processo de software deve ser conduzido para assegurar a qualidade.

A seguir, uma lista das normas e modelos usados e seus históricos:

CMMI	<p>Em 1986 – Início do desenvolvimento do modelo (solicitação do governo).</p> <p>Em julho de 1987 – Breve descrição do modelo de maturidade.</p> <p>Em setembro de 1987 – Versão preliminar do questionário de maturidade.</p> <p>Em 1991 – Primeira versão do CMM 1.0.</p> <p>Em 1993 – Modelo completamente definido: CMM 1.1.</p> <p>Atual – CMMI v. 1.3 (janeiro de 2013).</p>
ISO/IEC 12207	<p>Tecnologia de informação – Processos de ciclo de vida de software Versão original (1995).</p> <p>Emenda 1 (2002).</p> <p>Emenda 2 (2004).</p>
ISO/IEC 15504	<p>Tecnologia de informação – Avaliação de processos</p> <p>Em 1991 – Estudo sobre a necessidade de uma norma para a avaliação de processos de software.</p> <p>Em 1993 – Início do projeto SPICE (Software Process Improvement and Capability determination).</p> <p>Em 1998 – Versão inicial da “norma SPICE” (publicada como Relatório Técnico – TR).</p> <p>Em 2003 – Encerramento do projeto SPICE e publicação da parte 2.</p> <p>Em 2004 – Publicação das partes 1, 3 e 4.</p>

MPSBR

Melhoria do Processo de Software Brasileiro
Dezembro de 2003 – Início do programa mobilizador para a Melhoria do Processo de Software Brasileiro, coordenado pela Softex (Associação para Promoção da Excelência do Software Brasileiro), com apoio do Ministério da Ciência e Tecnologia (MCT) e do Banco Interamericano de Desenvolvimento (BID).
Abril de 2005 – Versão 1.0.
Maio de 2006 – Versão 1.1.
Maio/junho de 2007 – Versão 1.2.
MR-MPS-SV:2012.
MR-MPS-SV:2015.

Quadro 2 - Lista de normas e modelos de padrões de qualidade de software
Fonte: adaptado de Andrade (2015).

Um modelo de melhoria deve permitir: (i) compreensão do estado atual do processo; (ii) desenvolvimento da visão do processo desejado; (iii) lista de ações necessárias à melhoria do processo por ordem de prioridade; (iv) geração de um plano para ações e alocação de recursos.

Os modelos de melhoria de processo de software ajudam a empresa a se organizar e estabelecer uma ordem de prioridades de tal forma que a qualidade possa efetivamente ser “construída” ao longo do processo e fazer parte, inherentemente, do produto gerado. Em uma melhoria de processo de software, o processo passa por uma avaliação, os resultados dessa avaliação conduzem a melhorias, as quais mostram mudanças que devem ser realizadas no processo e isso deve ser feito continuamente (ANDRADE, 2015, p. 64).

Nas próximas aulas, discutiremos as normas e os modelos de padrões de qualidade de software: CMMI (*Capability Maturity Model Integration*), que abrange a qualidade do processo de software; ISO/IEC 15504 – SPICE, que orienta a empresa à melhoria contínua e à determinação da capacidade de seus processos e o MPS.BR, um projeto para a melhoria do processo de software brasileiro.



O dilema da qualidade de software pode ser mais bem sintetizado enunciando-se a Lei de Meskimen – Nunca há tempo para fazer a coisa certa, mas sempre há tempo para fazê-la de novo. Meu conselho: tomar o tempo necessário para fazer certo da primeira vez quase nunca é uma decisão errada.

(Roger Pressman e Bruce Maxim)



2 MODELO DE MELHORIA DA MATURIDADE DO processo de software: CMMI



O CMMI (*Capability Maturity Model Integration* – Modelo Integrado de Maturidade e Capacitação) foi desenvolvido pelo SEI (*Software Engineering Institute*) e é considerado um modelo de referência para a melhoria da maturidade dos processos da empresa. De acordo com Rezende (2005), o CMMI é um modelo que busca avaliar e melhorar a capacitação das empresas que desenvolvem um produto de software, propondo práticas que auxiliam as empresas na busca contínua de soluções para o seu crescimento no mercado com qualidade.

Um ponto importante é compreender as palavras-chave do modelo, que são: maturidade e capacidade.

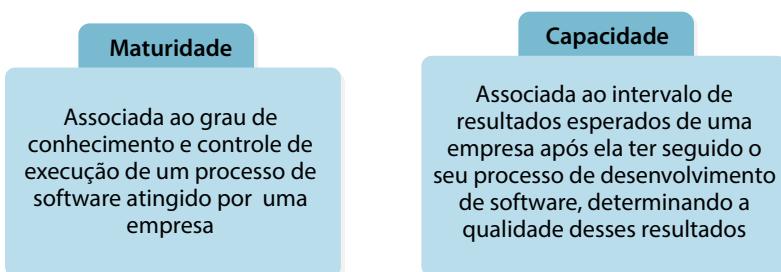


Figura 2 - Palavras-chave do modelo / Fonte: a autora.

Assim, maturidade e capacidade se complementam e, juntas, servem como ótimos indicadores para avaliar a confiabilidade no desenvolvimento de um software com qualidade.

Na Figura 3, são apresentados os objetivos principais do CMMI:

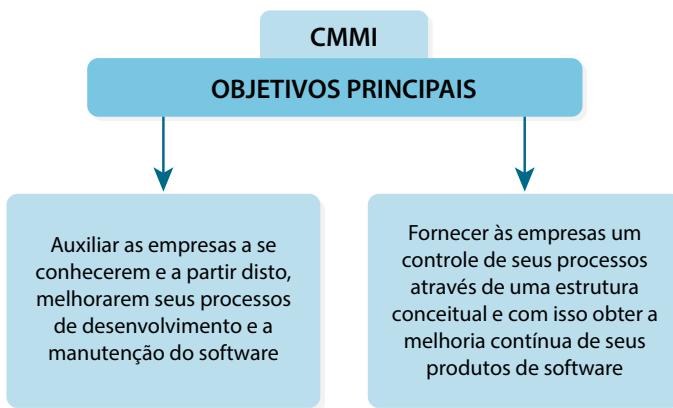


Figura 3 - Objetivos principais do CMMI / Fonte: adaptada de Rezende (2005).

O modelo apenas indica o que deve ser feito, não diz como implementar determinadas práticas, descreve um caminho de melhoria de maturidade, por meio de cinco níveis distintos, cada um com características individuais que determinam qual é a capacitação do processo (Figura 4). Quanto maior o nível, maior é a maturidade dos processos de desenvolvimento de software de uma empresa (REZENDE, 2005).

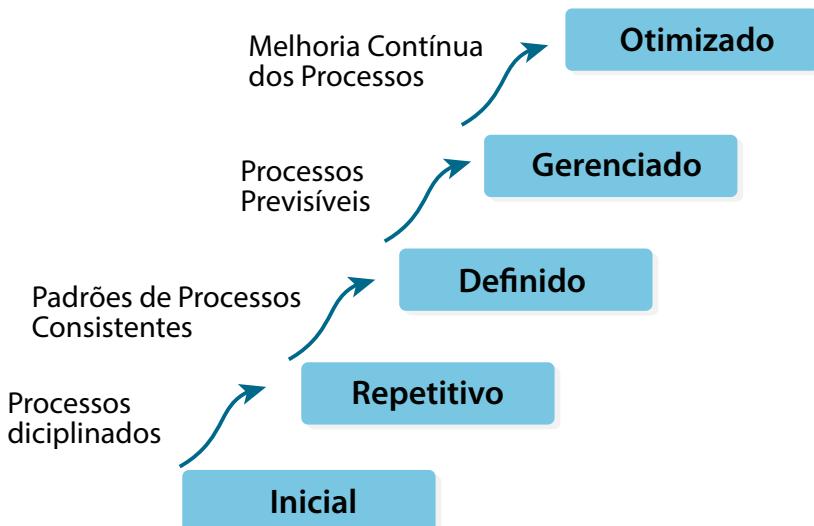


Figura 4 - Níveis do modelo CMMI / Fonte: adaptada de Machado (2016, p. 52).

Com exceção do nível 1 (inicial), todos os outros níveis são compostos por Áreas-Chave do Processo (KPA – *Key Process Area*), que compreendem as metas de melhoria do processo em cada nível. Cada área-chave é dividida nas práticas-chave, que são os quesitos que deverão ser cumpridos pela empresa na implantação do modelo.

Essas práticas-chave especificam o que deve ser seguido, como documentos exigidos, treinamentos ou políticas definidas para as atividades desenvolvidas pela empresa, mas não especificam como devem ser implementadas (Figura 5).

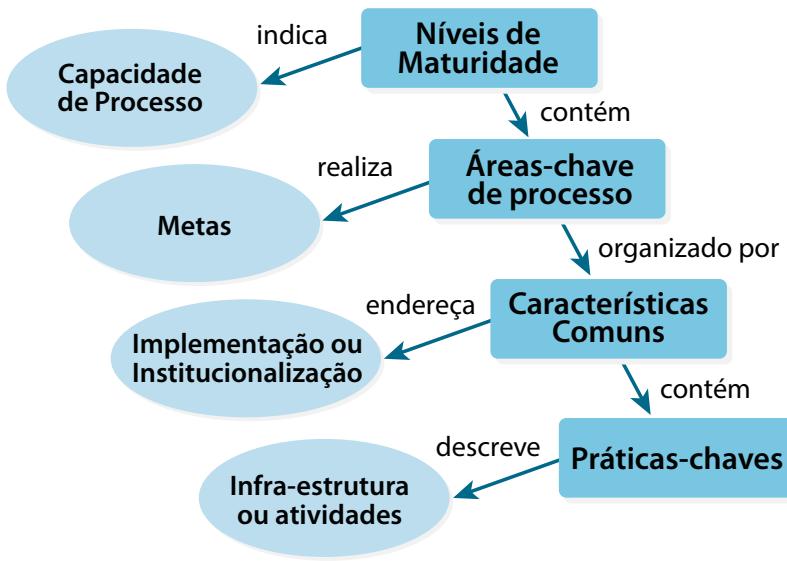


Figura 5 - Estrutura do modelo CMMI / Fonte: a autora.

A seguir, o Quadro 3 apresenta os níveis de maturidade e as áreas de processos, segundo Retamal (2005, on-line). Este quadro nos mostra que cada nível de maturidade é composto por várias áreas-chave de processo e estes possuem objetivos específicos e genéricos. Cada objetivo específico é alcançado por meio de algumas práticas bem definidas, cada objetivo genérico especifica várias funcionalidades comuns e estas estão ligadas às práticas genéricas, estas práticas são os detalhes operacionais que devem ser abordados pelo processo ou pela metodologia de desenvolvimento usada na empresa.

NÍVEIS E CARACTERÍSTICAS	ÁREAS-CHAVE DE PROCESSO
Nível 5: otimizado. Processos são continuamente melhorados com base em inovação e análise de causas de problemas.	Prevenção de defeitos. Gerenciamento de mudanças tecnológicas. Gerenciamento de mudanças de processo.
Nível 4: gerenciado. Processos são quantitativamente entendidos e gerenciados.	Gerenciamento quantitativo de projeto. Desempenho do processo organizacional.
Nível 3: definido Processos caracterizados no nível organizacional e as ações gerenciais são mais proativas. Definição dos processos organizacionais	Foco no processo organizacional. Treinamento organizacional. Análise e tomada de decisão. Validação. Verificação. Integração de produtos. Solução técnica. Desenvolvimento dos requisitos. Gerenciamento de riscos. Gerenciamento integrado de projeto.
Nível 2: repetitivo Processos são estabelecidos e gerenciados no nível dos projetos, e as ações gerenciais são, frequentemente, reativas.	Gerenciamento de configuração. Garantia da qualidade de processo e produto. Medição e análise. Gerenciamento de acordo com fornecedores. Monitoramento e controle de projetos. Planejamento de projetos. Gerenciamento de requisitos.
Nível 1: inicial Processos são imprevisíveis, não gerenciados e reativos.	Não existem áreas de processos neste nível.

Quadro 3 - Níveis de maturidade e áreas de processos

Fonte: adaptado de Retamal (2005, on-line).

As áreas-chave dos processos podem ser agrupadas em quatro categorias, como é mostrado no Quadro 4:

Categoria	Áreas-chave de processos
Gerência de processo	<ul style="list-style-type: none"> ■ Foco no processo de empresa. ■ Definição dos processos da empresa. ■ Treinamento organizacional. ■ Desempenho do processo organizacional. ■ Inovação e implantação organizacional.
Gerência de projeto	<ul style="list-style-type: none"> ■ Planejamento de projeto. ■ Monitoramento e controle de projeto. ■ Gerenciamento contratos com fornecedores. ■ Gerência integrada de projeto. ■ Gerência de riscos. ■ Gerenciamento quantitativo do projeto.
Engenharia	<ul style="list-style-type: none"> ■ Desenvolvimento de requisitos. ■ Gerência de requisitos. ■ Solução técnica. ■ Integração do produto. ■ Verificação. ■ Validação.
Supporte	<ul style="list-style-type: none"> ■ Gerência de configuração. ■ Garantia da qualidade de processos e do produto. ■ Medição e análise. ■ Análise e resolução de decisão. ■ Análise e resolução causal.

Quadro 4 - As categorias de áreas-chave de processos do CMMI

Fonte: adaptado de Retamal (2005, on-line).

Para ficar mais compatível com a proposta da ISO, o SEI criou o CMMI em duas representações: contínua e em estágios.

- **Representação por estágios:** mantém as ideias do modelo CMM com algumas alterações nas áreas de processo, além da adição de novos nomes: o nível 2 (repetitivo) passa para gerenciado e o nível 4 (gerenciado), para gerenciado quantitativamente.

- **Representação contínua:** focada nos processos individuais, como o gerenciamento de requisitos. O importante é o quanto a empresa está capacitada nesta área de desenvolvimento, sem se preocupar com os outros processos (RETAMAL, 2005, on-line).

O modelo CMMI tornou-se um modelo conhecido pelos seus detalhes e pelas orientações para a melhoria nos processos de software. O CMMI influenciou outros modelos, como o SPICE – ISO/IEC 15504 e o MPS.BR.



explorando Ideias

A melhoria da qualidade de processo de software deve ser o dia a dia de uma organização de software. Com o sucesso desta empreitada, a previsão para o início da próxima década é um esforço concentrado na redução substancial dos prazos para a entrega de produtos pelas organizações e a exportação do software brasileiro. A pressão dos concorrentes será intensa; organizações que forem capazes de integrar, harmonizar e acelerar seus processos de desenvolvimento e manutenção de software terão a primazia do mercado.

Fonte: Guerra e Colombo (2000, p. 32).





MODELO DE QUALIDADE DE PROCESSO DE software: modelo MPS.BR

O modelo MPS.BR (Melhoria de Processos do Software Brasileiro) é considerado, conforme Andrade (2015, p. 89), “simultaneamente um movimento para a melhoria e um modelo de qualidade de processo voltada para a realidade do mercado de pequenas e médias empresas de desenvolvimento de software no Brasil”.

O MPS.BR é um modelo de qualidade de processos de software que auxilia as empresas brasileiras a implantarem processos em conformidade com os principais padrões e modelos de qualidade de software no mercado internacional (SOFTEX, 2016).



Ele é baseado no CMMI, nas normas ISO/IEC 12207 e ISO/IEC 15504 e na realidade do mercado brasileiro. No Brasil, uma das principais vantagens do modelo é seu custo reduzido de certificação em relação às normas estrangeiras, sendo ideal para micro, pequenas e médias empresas (ANDRADE, 2015, p. 90).

A Figura 6 ilustra os principais componentes do modelo, além de suas principais fontes (os modelos ISO/IEC 12207, ISO/IEC 15504 (SPICE) e CMMI).

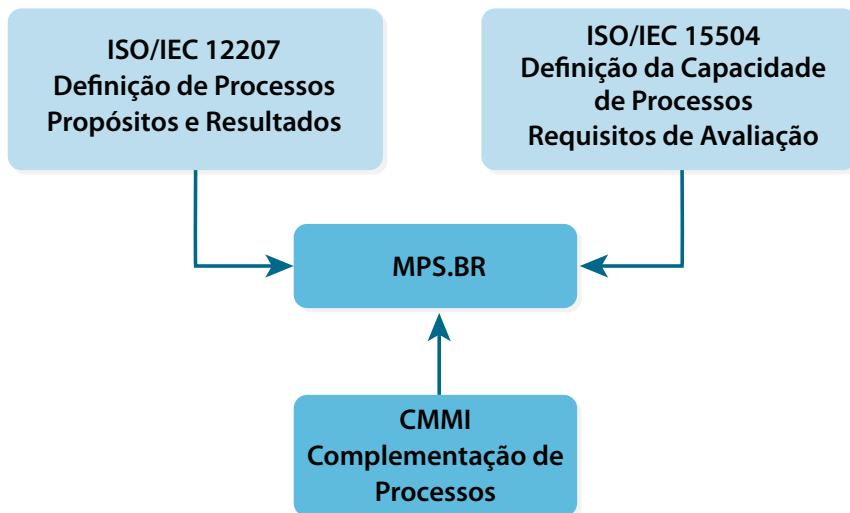


Figura 6 - MPS.BR: desenvolvimento e aprimoramento e suas principais fontes
Fonte: adaptado de Softex (2016).

O modelo possui níveis de maturidade que procuram estabelecer uma escala de evolução dos processos de software, e isto caracteriza estágios de melhoria durante a implementação de processos na empresa.

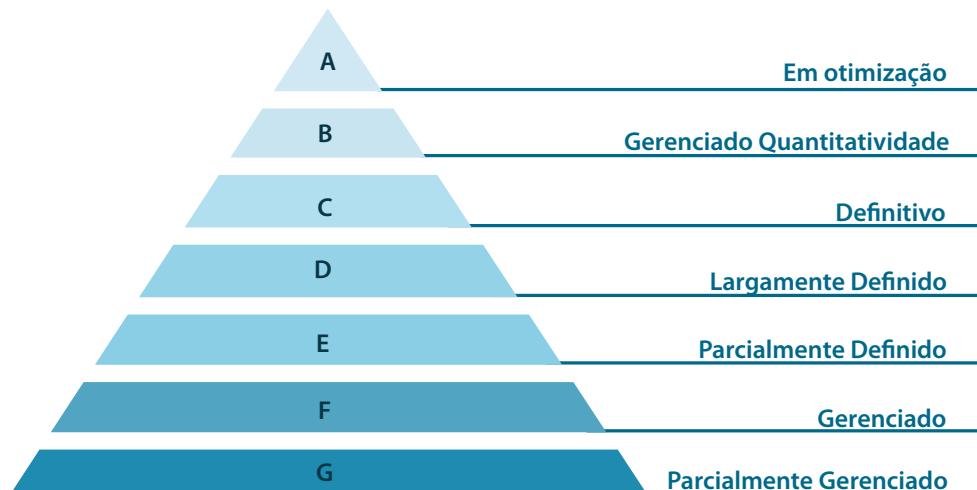


Figura 7 - Níveis do Modelo MR-MPS / Fonte: adaptada de Softex (2016).

Conforme Softex (2016, p.18):



os níveis são acumulativos, ou seja, se a empresa se encontra no nível F, ela possui o nível de capacidade do nível F que inclui os atributos de processo dos níveis G e F para todos os processos relacionados no nível de maturidade F (que também inclui os processos de nível G). Ao passar do nível G para o nível F, os processos do nível de maturidade G passam a ser executados no nível de capacidade correspondente ao nível F.

Conforme o Quadro 5, o modelo MR-MPS define sete níveis de maturidade:

Nível G - Parcialmente gerenciado	Primeiro nível do modelo, é composto pelos processos de gerência de projeto e pela gerência de requisitos.
Nível F - Gerenciado	Composto pelo nível de maturidade G, acrescido dos processos de gerência de configuração, garantia da qualidade, medição e aquisição.
Nível E - Parcialmente definido	Composto pelo nível de maturidade F, acrescido dos processos de treinamento, definição do processo organizacional, avaliação e melhoria do processo organizacional e adaptação do processo para gerência de projetos.
Nível D - Largamente definido	Composto pelo nível de maturidade E, acrescido dos processos de desenvolvimento de requisitos, solução técnica, validação, verificação, integração de produto, instalação e liberação de produto.
Nível C - Definido	Composto pelo nível de maturidade D, acrescido dos processos de gerência de riscos e análise de decisão e resolução.
Nível B - Gerenciado quantitativamente	Composto pelo nível de maturidade C, acrescido dos processos de desempenho do processo organizacional e pela gerência quantitativa de projeto.
Nível A - Em otimização	Nível mais elevado do modelo MPS.BR; é composto pelo nível de maturidade B, acrescido dos processos de inovação e implantação na organização e análise e resolução de causas.

Quadro 5 - Níveis de maturidade do modelo MR-MPS / Fonte: adaptado de Softex (2016).

O progresso de um nível de maturidade, de acordo com Andrade (2015, p. 90), “se obtém quando são atendidos os propósitos e todos os resultados esperados dos respectivos processos e dos atributos de processo estabelecidos para aquele nível”.

Cada nível de maturidade possui suas áreas de processo, em que são analisados conforme o apresentado na Figura 8.

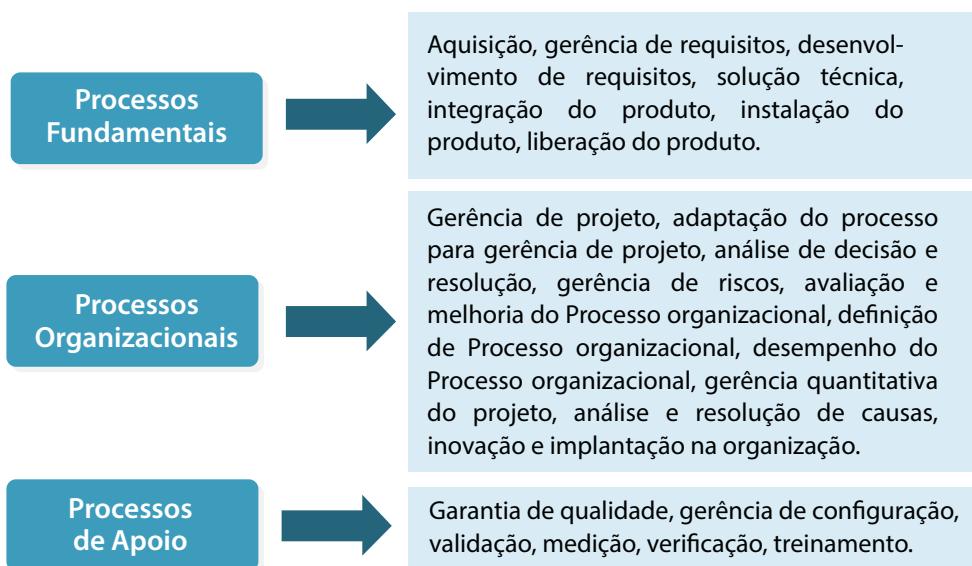


Figura 8 - Áreas de processo de cada nível de maturidade / Fonte: adaptada de Softex (2016).

A capacidade do processo, segundo Andrade (2015, p. 90), é “representada por um conjunto de atributos de processo descrito em termos de resultados esperados”. O grau de refinamento e de institucionalização é expresso na capacidade de o processo ser executado na empresa.



A qualidade de software pode ser avaliada por diferentes visões, ou seja, pode ser avaliada do ponto de vista do usuário, do desenvolvedor e, também, da empresa que o comercializa. Independentemente da visão, todos concordam que a usabilidade de um software é um fator extremamente importante, um software difícil de usar, provavelmente, será considerado com qualidade por nenhuma das visões citadas.

Fonte: adaptado de Andrade (2015, p. 44).



NORMAS DE QUALIDADE DE SOFTWARE norma ISO/IEC 12207

```
if(operation == "MIRROR_X":  
    error_mod.use_x = True  
    error_mod.use_y = False  
    error_mod.use_z = False  
if(operation == "MIRROR_Y":  
    error_mod.use_x = False  
    error_mod.use_y = True  
    error_mod.use_z = False  
if(operation == "MIRROR_Z":  
    error_mod.use_x = False  
    error_mod.use_y = False  
    error_mod.use_z = True  
  
def operator_mirroring(self, context):  
    ob = context.object  
    r_o.select_set(True)  
    intext.scene.objects.active = ob  
    bpy.context.selected_objects.append(ob)  
    ob.select_set(True)  
    ob.name = "one"  
    ob.data.name = "one"  
  
    int("please select exactly one object")  
  
    --- OPERATOR CLASSES -----  
  
    types.Operator:  
        X mirror to the selected object.  
        bpy.ops.mirror.mirror_x()  
        error_X"  
  
    context:  
        context.active_object is not None
```

A ISO/IEC 12207 – Tecnologia da Informação – Processos de ciclo de vida de software estabelece uma estrutura comum para esses processos e tem, como objetivo principal, fornecer uma estrutura única para os envolvidos na produção de software e auxiliar a definir seus papéis na forma de processos bem definidos, ajudando a empresa a entender melhor as atividades que deverão ser executadas durante o desenvolvimento do software (RETAMAL, 2005, on-line). Essa estrutura foi criada de maneira a ser flexível, modular e adaptável às necessidades das empresas.

A ISO/IEC 12207 foi desenvolvida com a participação de vários países, entre eles, o Brasil. Inicialmente, foi publicada em 1995 (versão NBR, em 1998). Mais tarde, veio a sofrer duas emendas (Figura 9):

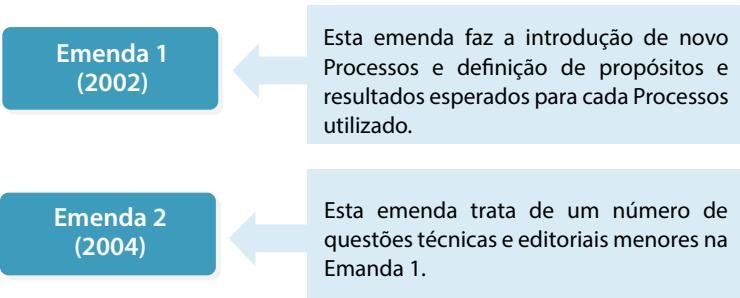


Figura 9 - Emendas / Fonte: a autora.

Após essas emendas, sai uma nova revisão para alinhamento, com a ISO 15288 (Engenharia de Sistemas – Processos de Ciclo de Vida de Sistemas), chamada 12207 R WD3 (junho de 2006).

A conformidade a essa norma é definida como a execução de todos os processos, de todas as atividades e tarefas selecionadas no processo de adaptação para o projeto de software (segundo a ISO/IEC 12207:1995). Deve ser demonstrado que a implementação de qualquer processo do conjunto declarado pela empresa resulta na realização do propósito e dos resultados correspondentes (segundo a Emenda 1: 2002).

Em função disto, a norma está baseada em dois princípios básicos, que são a **modularidade** e a **responsabilidade**. Em modularidade, os processos têm mínimo acoplamento e máxima coesão, ou seja, todas as partes que envolvem um processo são fortemente relacionadas, e as interfaces têm um número mínimo. Em responsabilidade, é estabelecido um único responsável por cada processo, facilitando o uso da norma, principalmente em projetos em que vários indivíduos podem estar, legalmente, envolvidos.

A norma agrupa as atividades que podem ser executadas durante o ciclo de vida em cinco processos fundamentais, oito processos de apoio e quatro processos organizacionais (Quadro 6):

PROCESSOS FUNDAMENTAIS		PROCESSOS DE APOIO	Desenvolvimento	PROCESSO DE ADAPTAÇÃO		
Aquisição		Documentação				
Fornecimento		Gerência de Configuração				
Operação	Operação	Garantia da Qualidade, Verificação Validação, Revisão Contínua				
	Manutenção	Auditória, Usabilidade, Gerência de Resoluções de Problemas, Gerência de Solicitação de Mudanças, Avaliação do Produto				
PROCESSOS ORGANIZACIONAIS						
Gerência	Engenharia de Domínio		Melhoria			
Gestão de Ativos	Infraestrutura					
Gestão de Programa de Reuso	Recursos Humanos					

Quadro 6 - ISO/IEC 12207 (2000): Processos / Fonte: adaptado de Andrade (2015).

Segundo Andrade (2015), os **processos fundamentais** atendem ao contrato entre fornecedor e contratante e à execução do desenvolvimento, da operação ou da manutenção de produtos de software durante seu ciclo de desenvolvimento. Os **processos de apoio** auxiliam e contribuem para o sucesso e a qualidade do projeto de software, e os **processos organizacionais** são empregados por uma empresa para estabelecer e implementar uma estrutura constituída pelos processos do ciclo de vida e pelo pessoal envolvido no seu desenvolvimento. O **processo de adaptação** define atividades que são necessárias para a norma se adaptar melhor às tarefas desenvolvidas pela empresa em seus projetos. Esta adaptação ocorre com base nos fatores que diferenciam uma empresa de outras, como as estratégias de aquisição, os modelos de ciclo de vida do projeto, as características do software e a cultura organizacional.

Em função disto, a norma pode ser adaptável a qualquer empresa, projeto, modelo de ciclo de vida, cultura e técnicas de desenvolvimento (ANDRADE, 2015).

A norma ISO/IEC 12207 é importante, também, para padronizar a nomenclatura e os processos juntos a outras normas. Ela é largamente utilizada no SPICE, o que torna possível um mapeamento entre os modelos.



A qualidade se mostrou um fator primordial, pois é ela que define a satisfação do usuário em relação ao software desenvolvido. Dessa forma, a implantação de normas e modelos de padrões de qualidade de software produz um software mais confiável e que atende aos requisitos que foram levantados.

Fonte: adaptado de Andrade (2015, p. 30).



Conforme Andrade (2015, p. 81), o projeto SPICE (*Software Process Improvement and Capability Determination*) “surgiu quando o grupo ISO se reuniu buscando estabelecer um padrão para a avaliação do processo de software, pois constatou a deficiência da Norma ISO 9001 em relação ao setor de software”.

O projeto foi criado em 1993 e tinha, como objetivo, gerar normas que pudessem avaliar os processos de software. O projeto foi concluído em 1995, quando a primeira versão foi gerada e, em 1998, foi publicada a norma ISO/IEC TR 15504: *Information Technology – Software Process Assessment* (Tecnologia da Informação – Avaliação de Processos de Software), como um relatório técnico.

A ISO/IEC 15504 (SPICE) é uma norma em elaboração conjunta pela ISO (*International Organization for Standardization*) e pelo IEC (*International Electrotechnical Commission*), sendo um padrão para a avaliação do processo de software visando determinar a capacitação (Figura 10) de uma empresa (REZENDE, 2005). Ela visa, também, orientar a empresa para a melhoria contínua do processo de software (Figura 11).

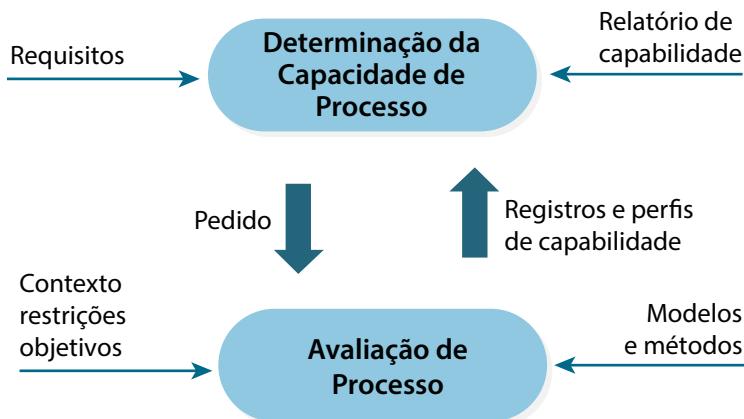


Figura 10 - Uso da ISO 15504 para a determinação da capacidade

Fonte: adaptada de ISO (1998).

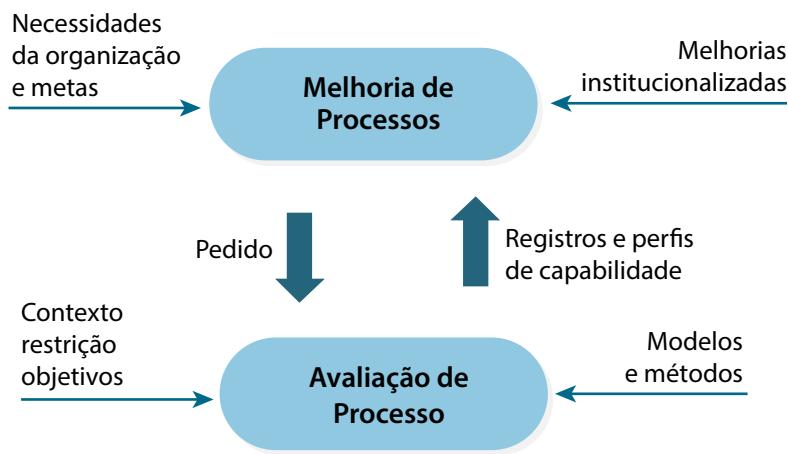


Figura 11 - Uso da ISO 15504 para a melhoria do processo / Fonte: adaptada de ISO (1998).



SPICE é um modelo contínuo, ou seja, define níveis de maturidade para determinados processos de acordo com diferentes características e o usuário determina sobre qual nível de maturidade o processo será avaliado. A avaliação pode ser realizada no contexto de melhoria contínua, identificando as oportunidades de melhoria dos processos de uma organização, ou para determinação da capacidade de processo, determinando a conformidade dos processos de uma organização em relação a requisitos ou contratos (ANDRADE, 2015, p. 81).

Segundo Rezende (2005), o SPICE possui um modelo de referência que serve como base para o processo de avaliação. O modelo de referência é um conjunto de processos fundamentais padronizados, que auxiliam e guiam para o bom desenvolvimento da engenharia de software. Este modelo é dividido, conforme Rezende (2005, p. 45), em cinco grandes “categorias de processo: Cliente-Fornecedor, Engenharia, Suporte, Gerência e Organização. Cada uma destas categorias é detalhada em processos mais específicos. Tudo isso é descrito em detalhes pela norma”.

O modelo SPICE também define seis níveis de capacitação para cada processo, que, segundo Rezende (2005, p. 45), podem ser: “incompleto, executado, gerenciado, estabelecido, previsível e otimizado. O resultado de uma avaliação retrata um perfil da empresa em forma de matriz, onde temos os processos nas linhas e os níveis nas colunas”.

O modelo compreende duas dimensões da norma ISO/IEC 15504, que são:

Dimensão de processo	Que se limita à verificação da execução ou não dos processos. Caracteriza o propósito dos processos relacionados ao conjunto de atividades do ciclo de vida do software e é dividida em três agrupamentos.
Dimensão de capacidade	Define os níveis do processo em uma escala de medida de seis pontos que podem ser utilizados como base na avaliação da execução de um processo de uma organização e como guia para a melhoria da execução.

Quadro 7 - Dimensões da norma ISO/IEC 15504 / Fonte: Andrade (2015, p. 82).

A seguir, o Quadro 8 mostra a estrutura das categorias de processos.

PROCESSO	CATEGORIAS
Processos fundamentais	Definem as atividades do ciclo de desenvolvimento do software, composto pelas categorias: cliente-fornecedor e engenharia.
Processos organizacionais	Definem as atividades que a organização deve executar, composto pelas categorias: gerência e organização.
Processos de apoio	Definem as atividades para garantir a qualidade do projeto de software, composto pela categoria: suporte.
Processos de engenharia	Abrange os processos relacionados ao desenvolvimento e à manutenção do software, especificando, implementando ou mantendo o produto de software e sua documentação para o cliente.
Processos de suporte	Consiste nos processos de apoio ou suporte que podem ser empregados por qualquer outro processo durante o ciclo de vida do software.
Processos de gerência	Consiste em processos que contêm práticas que podem ser utilizadas por quem administra qualquer processo ou projeto dentro do ciclo de vida de desenvolvimento de software.
Processos organizacionais	Consiste em estabelecer objetivos tanto de negócio como de recursos humanos da organização, abrange processos relacionados às atividades gerais de uma organização que ajudam a alcançar esses objetivos.

Quadro 8 - Estrutura das categorias de processos da norma ISO/IEC 15504 / Fonte: adaptado de Andrade (2015, p. 82).

O Quadro 9 mostra a estrutura dos níveis de capacitação de cada processo.

Nível 0 – Incompleto	Processo não é implementado ou não atinge seus objetivos.
Nível 2 – Gerenciado	Processo anterior é executado e gerenciado, isto é, planejado, controlado, acompanhado, verificado e corrigido, estando em conformidade com padrões e requisitos estabelecidos (qualidade, prazo e custo).
Nível 3 – Estabelecido	Processo anterior é executado e gerenciado com base nos princípios de uma boa engenharia de software, de forma eficaz e eficiente.
Nível 4 – Previsível	Processo anterior é executado dentro dos limites de controle estabelecido para atingir os objetivos definidos do processo e com medições detalhadas de desempenho, que são analisadas chegando ao entendimento quantitativo da capacidade do processo e a uma execução gerenciada quantitativamente.
Nível 5 – Em otimização	Processo anterior é alterado e adaptado continuamente, visando satisfazer os objetivos do negócio da organização, estabelecendo metas quantitativas de eficiência e eficácia para o desempenho do processo.

Quadro 9 - Estrutura dos níveis norma ISO/IEC 15504

Fonte: adaptado de Andrade (2015, p. 82).

A norma ISO/IEC 15504 está organizada em nove partes. Três partes são normativas e as demais são informativas, conforme a Figura 12.

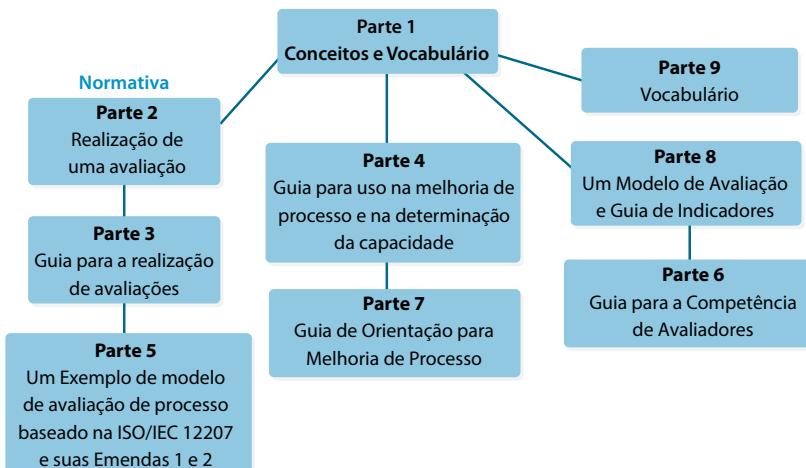


Figura 12 - Partes da norma ISO/IEC 15504 / Fonte: adaptado de Andrade (2015).

A seguir, o Quadro 10 explica cada uma das partes da norma ISO/IEC 15504:

Parte 1 – Conceitos e vocabulário (informativa)	Provê uma introdução geral aos conceitos de avaliação de processos e um glossário de termos relacionados à avaliação.
Parte 2 – Realização de uma avaliação (normativa)	Define os requisitos normativos para a realização de uma avaliação de processo e define uma infraestrutura de medição para avaliar a capacidade do processo. Essa infraestrutura de medição define nove atributos de processo, agrupados em seis níveis de capacidade.
Parte 3 – Guia para a realização de avaliações (informativa)	Provê orientações para interpretar os requisitos para a realização de uma avaliação.
Parte 4 – Guia para uso na melhoria de processo e na determinação da capacidade de processo (informativa)	Provê orientações para a utilização de avaliação de processo para propósitos de melhoria de processo e de determinação da capacidade.
Parte 5 – Um exemplo de modelo de avaliação de processo baseado na ISO/IEC 12207 e suas Emendas 1 e 2 (informativa)	Contém um exemplo de modelo de avaliação de processo que é baseado no modelo de processo de referência definido na ISO/IEC 12207 e suas Emendas 1 e 2.
Parte 6 (informativa) – Guia para competência de avaliadores	Descreve a competência, a formação, o treinamento e a experiência dos avaliadores que são fundamentais para a execução da avaliação de processos. Esta parte fornece mecanismos que podem ser utilizados com o objetivo de comprovar a competência e validar a formação, o treinamento e a experiência dos avaliadores.

Parte 7 (informativa) – Guia de Orientação para Melhoria de Processo	Esta parte descreve como definir as entradas para a execução de uma avaliação e como utilizar os resultados obtidos visando à melhoria do processo. Fazendo isto, uma organização define um método para a escolha dos processos que estão descritos na parte 2 do modelo SPICE, que é parte de uma abordagem para a melhoria da organização.
Parte 8 (informativa) – Um modelo de avaliação e guia de indicadores	Descreve como definir as entradas para a execução de uma avaliação e como utilizar os resultados obtidos visando à determinação da capacidade do processo. Esta parte auxilia a organização a identificar os pontos fortes e fracos associados ao desenvolvimento de processos e os riscos de firmar um contrato com um fornecedor.
Parte 9 (normativa) – Vocabulário	Define todos os termos, em ordem alfabética, usados por todas as partes que compõem o modelo.

Quadro 10 - Partes da norma ISO/IEC 15504 / Fonte: adaptado de Andrade (2015, p. 86-87).



explorando Ideias

A compatibilização com a norma ISO/IEC 12207, visto que há muita superposição entre a dimensão de processos do ISO/IEC TR 15504 e da ISO/IEC 12207, provoca a duplicação de esforços, inconsistências e dificuldade na utilização da descrição de processos da ISO/IEC 12207, que contêm conceito de capacidade embutido.

Para solucionar estes problemas, a dimensão de processos foi removida da ISO/IEC 15504, se tornando um anexo da norma ISO/IEC 12207, publicada como AMD1 e AMD2, e permitindo a ISO/IEC 15504 utilizar qualquer modelo compatível de descrição de processo como modelo de referência. Sendo assim, a ISO/IEC 15504 será uma norma para a avaliação da capacidade de processos. O maior benefício destas alterações está na abordagem comum de avaliação, não mais restrita aos processos do ciclo de vida do software, e sim, pode ser executada em qualquer tipo de processo, em qualquer domínio.

Fonte: Andrade (2015, p. 88).

CONSIDERAÇÕES FINAIS

Caro(a) aluno(a), chegamos ao final da nossa unidade, em que estudamos os modelos de melhoria e avaliação de processo de software.

Na primeira aula, falamos sobre os principais modelos de melhoria e avaliação de processo de software e que a qualidade de um software está diretamente relacionada à qualidade do seu processo.

Na segunda aula, foi estudado o modelo de melhoria da maturidade de processo de software CMMI (*Capability Maturity Model Integration*), considerado um modelo que busca avaliar e melhorar a capacitação das empresas que desenvolvem um produto de software.

Na sequência, explicamos o modelo MPS.BR, considerado um modelo de qualidade de processo voltado para a realidade do mercado de empresas de desenvolvimento de software no Brasil. O modelo possui níveis de maturidade que procuram estabelecer uma escala de evolução dos processos de software, e isso caracteriza estágios de melhoria durante a implementação de processos na empresa.

Aprendemos, também, a norma de qualidade de software ISO/IEC 12207, a qual estabelece uma estrutura comum para os processos de ciclo de vida de software, e como é importante, também, para padronizar a nomenclatura e os processos junto a outras normas. Ela é largamente utilizada no SPICE, o que torna possível um mapeamento entre os modelos.

E, por fim, foi estudada a norma ISO/IEC 15504 – SPICE para a qualidade de software, sendo um modelo contínuo que define níveis de maturidade para determinados processos, de acordo com diferentes características para o processo a ser avaliado.

Depois destes conhecimentos adquiridos ao longo da unidade, você está preparado(a) para seguir adiante? Então, continue sua leitura nas próximas unidades!



1. 1. "Quando um novo projeto de software se inicia, seu planejamento é um desafio e nem sempre é possível conseguir preencher todas as lacunas existentes no momento de sua elaboração. Dificuldades, imprevistos e retrabalho são comuns em muitos projetos, gerando desconforto para a empresa e para o cliente" (DIAS, 2019, on-line). Disponível em: <https://medium.com/contexto-delimitado/os-processos-de-software-56a2e70fddfb>. Acesso em: 26 out. 2020.

Com base no texto apresentado e nas empresas que possuem um processo de software de alta qualidade com políticas, padrões e estruturas organizacionais para um processo, descreva o que isso acarreta a empresa construir.

2. Produzir software com qualidade é, portanto, uma tarefa árdua. Esta dificuldade é maior no contexto das empresas de pequeno porte, já que raramente trabalham com processos formais e geralmente enfrentam problemas de escassez de recursos humanos e financeiros. Cenário tão alarmante fez levantar a seguinte hipótese: 'A qualidade de um sistema de software e o sucesso de um projeto de software são largamente definidos pela qualidade do processo utilizado para desenvolvê-lo' (AZEVEDO, 2014, on-line). Disponível em: <https://www.leanti.com.br/artigos/18/melhoria-de-processo-de-software,-por-onde-comecar.aspx>. Acesso em: 26 out. 2020.

Considerando o texto apresentado, a respeito da motivação que uma empresa de software tem para tentar melhorar seu processo, assinale a alternativa correta:

- a) A forte competição dentro da empresa, a regulação externa e a ampla necessidade de melhorar o desempenho de sua empresa e de seus produtos de software desenvolvidos.
- b) A forte competição do negócio, a regulação externa e a ampla necessidade de melhorar o desempenho de sua empresa e de seus produtos de software desenvolvidos.
- c) A forte competição do negócio, a regulação externa e a ampla necessidade de melhorar o desempenho das outras empresas e dos produtos de software desenvolvidos por outros fornecedores.



- d) A grande demanda de serviços e produtos na área de desenvolvimento de software, a regulação externa e a forte necessidade de melhorar o desempenho de sua empresa e de seus produtos de software desenvolvidos.
- e) e) A grande demanda de produtos com qualidade, a regulação externa e a forte necessidade de melhorar o desempenho de sua empresa e de seus produtos de software desenvolvidos.
3. “A melhoria de processos deve ser realizada em passos curtos, que se encaixem dentro das limitações comuns às pequenas empresas. A ideia é substituir a adoção compulsória de processos pela adoção gradual de pequenas práticas que gerem impacto positivo na produção de software e possam servir posteriormente para compor processos mais sofisticados” (AZEVEDO, 2014, on-line). Disponível em: <https://www.leanti.com.br/artigos/18/melhoria-de-processo-de-software,-por-onde-comecar.aspx>. Acesso em: 26 out. 2020.

Com base no texto apresentado, analise as afirmativas, a seguir, sobre as representações do CMMI:

- I - O CMMI tem três representações: contínua, em estágios e gerenciado.
- II - A representação por estágios mantém as ideias do modelo CMM, com algumas alterações nas áreas de processo.
- III - A representação gerenciada teve a adição de novos nomes: o nível 2 (repetitivo) passa para gerenciado e o nível 4 (gerenciado) para gerenciado quantitativamente.
- IV - A representação contínua é focada nos processos individuais, como o gerenciamento de requisitos.

É correto o que se afirma em:

- a) I, apenas.
- b) II e IV, apenas.
- c) III e IV, apenas.
- d) I, II e IV, apenas.
- e) I, II, III e IV.



4. “O MPS.BR é um movimento criado em uma parceria entre o Ministério da Ciência, Tecnologia e Inovação e a SOFTEX – Associação para Promoção da Excelência do Software Brasileiro e algumas empresas e instituições de ensino. O programa foi criado e tem como foco principal as micros, pequenas e médias empresas do mercado de software. Como desenvolvedores observam todos os dias, essas empresas têm algumas dificuldades e poucos recursos para a melhoria dos processos, embora tenham grande necessidade de fazê-lo” (DEVMEDIA, 2014, on-line). Disponível em: <https://www.devmedia.com.br/melhoria-do-processo-de-software-brasileiro/29915>. Acesso em: 26 out. 2020.

Considerando o fragmento de texto apresentado, descreva cada nível de maturidade e suas áreas de processo.

5. “A ISO/IEC 12207 – Tecnologia da Informação – Processos de ciclo de vida de Software - estabelece uma estrutura comum para os Processos de ciclo de vida de Software e tem como objetivo principal fornecer uma estrutura única para os envolvidos na produção de Software e auxiliar a definir seus papéis na forma de Processos bem definidos, ajudando a empresa a entender melhor as atividades que deverão ser executadas durante ao desenvolvimento do Software” (RETAMAL, 2005, on-line).

Pensando sobre o texto apresentado e sabendo que a norma agrupa as atividades que podem ser executadas durante o ciclo de vida, descreva o processo de adaptação.



ANÁLISE COMPARATIVA DAS NORMAS E DOS MODELOS DE PADRÕES DE QUALIDADE DE SOFTWARE

A seguir, temos uma análise comparativa das normas e dos modelos de padrões de qualidade discutidos: CMMI, PSP, SPICE, MPS.BR e norma ISO/IEC 12207. No modelo CMMI, o objetivo é determinar a capacitação da empresa e apoiar a sua evolução de acordo com os níveis estabelecidos pelo modelo, enquanto que, no modelo SPICE, o objetivo é conhecer e avaliar os processos da empresa para determinar a capacitação e promover a sua melhoria. Na norma ISO/IEC 12207, o objetivo é estabelecer uma terminologia e um entendimento comum para os processos entre todos os envolvidos com o software. No modelo PSP, o objetivo é auxiliar o desenvolvedor a estimar e planejar suas tarefas e a acompanhar seu desempenho, e, no modelo MPS.BR, o objetivo é orientar as empresas brasileiras a implantar a melhoria de processos de software em conformidade com os padrões internacionais.

A abordagem do modelo CMMI é na avaliação dos processos de software e seu enquadramento na empresa de acordo com os níveis de maturidade; no modelo SPICE, a avaliação dos processos da empresa é em relação aos níveis de capacitação. Na norma ISO/IEC 12207, a abordagem é na definição dos processos para a aquisição, o fornecimento, o desenvolvimento, a operação e a manutenção do software. O modelo PSP concentra-se no projeto, na codificação e na fase de testes de desenvolvimento de software. O modelo MPS.BR buscar o desenvolvimento de um modelo de referência para a melhoria de processo do software que fosse adequado à realidade das empresas PMEs.

O foco no modelo CMMI é dado nos aspectos técnicos e de engenharia, gerência de projetos, e o PSP é um modelo que trabalha totalmente focado em índices, o que faz com que o desenvolvedor, constantemente, situe-se para efeito de qualidade e produtividade. O SPICE tem seu foco nos aspectos técnicos e de engenharia e nos gerenciamentos organizacionais. O modelo MPS.BR tem seu foco na estratégia de implementação, que é totalmente voltado para a realidade das micro e pequenas empresas brasileiras. A norma ISO/IEC 12207 foi concebida de modo a ser aplicável a qualquer empresa que desenvolva e/ou realize atividades de manutenção de produtos de software.

Em relação à característica empresas-alvo, o modelo CMMI é para empresas que



precisam de comprovação formal de sua capacitação e para as normas; já os modelos PSP, SPICE e ISO/IEC 12207 são para empresas em geral. O modelo MPS.BR é voltado às empresas brasileiras de pequeno e médio porte.

Para a característica definição de processos, o modelo CMMI estabelece 18 áreas-chave de processos organizados em cinco níveis crescentes de maturidade, enquanto que, no modelo SPICE, são estabelecidos 29 processos organizados em cinco categorias; na norma ISO/IEC 12207, são estabelecidos 17 processos organizados em três categorias. O modelo PSP estabelece sete níveis de processo de forma incremental, o modelo MPS.BR estabelece três componentes, e um deles (MR-MPS) é dividido em sete níveis de maturidade.

O modelo CMMI não tem flexibilidade, pois os níveis e áreas-chave de processo, que são a base do modelo, não podem ser alterados. O modelo SPICE tem flexibilidade, permitindo a definição de perfis de processo e práticas de acordo com os objetivos da empresa. Um dos preços dessa flexibilidade é a inexistência de um roteiro claro para melhoria; a norma ISO/IEC 12207 possui uma classificação de processos que pode ser utilizada conforme os objetivos da empresa, o que a torna flexível. O modelo PSP tem níveis superiores que adicionam características aos níveis implantados; já o MPS.BR tem níveis e áreas-chave do processo que são a base do modelo e, por isso, não há flexibilidade.

O modelo CMMI teve influência e inspiração dos princípios de Shewhart, Deming, Juran, Crosby. O modelo SPICE teve influência e inspiração TQM, PDCA, CMM, STD, Trillium, Malcolm, Baldrige, Bootstrap; a norma ISO/IEC 12207 TQM e PDCA. O modelo PSP teve influência e inspiração dos modelos CMMI e TSP, já o modelo MPS.BR teve influência e inspiração das normas e dos modelos CMMI, ISO/IEC 122207 e SPICE.

A seguir, faremos uma comparação geral entre os modelos e as características: benefícios e dificuldades.

Os benefícios do modelo CMMI são: o estabelecimento de diretrizes para a melhoria contínua na empresa e a difusão extensa nos EUA. No modelo SPICE, a norma internacional está em elaboração e há a expansão maior em relação aos outros modelos citados. Na norma ISO/IEC 12207, há a definição de uma taxonomia para processos, útil para qualquer empresa.



Em comparação ao CMMI, o SPICE leva desvantagem (dificuldades) no sentido da disponibilidade de informações, uma vez que a CMMI oferece download gratuito, O SPICE não, e tem um custo elevado. O CMMI é ativamente patrocinado pelo DoD dos EUA e foi criado antes do SPICE, mas, posteriormente, substituído pelo CMMI, que incorpora muitas das ideias do SPICE, além de outros benefícios do CMM. Muitas empresas preferem apostar no CMMI, devido ao fato de este ser um modelo americano. O SPICE tem algumas dificuldades de aplicações no sentido econômico, já que foi experimentado, primeiro, num cenário militar (MENDES, 2005). No CMMI, os níveis de maturidade não podem ser omitidos e a competência é construída por estágios, onde cada nível de maturidade oferece fundamento necessário para melhorias do nível seguinte. Porém se a empresa não tem disciplina de gerenciamento, o processo como um todo é sacrificado. A importância dos níveis para as empresas é que elas podem focar, somente, em algumas atividades de melhorias de cada vez (PADILHA, 2000).

O SPICE apresenta uma estrutura para avaliação e melhoria de processos. É uma norma internacional, não dedicada apenas para software. Ela introduz o conceito de modelo de referência de processo, que é externo à norma. A norma ISO/IEC 12207 pode ser o modelo de referência de processo quando o SPICE for aplicado a software; o SPICE tem documentação extensa, um modelo bem detalhado, níveis de capacidade e nove atributos, requisitos para avaliação e orientações na aplicação, ela dá suporte a todo ciclo de vida do software (PIRES, 2000).

Os benefícios do modelo MPS.BR são os sete níveis de maturidade, o que possibilita a implantação mais gradual e mais visibilidade dos resultados de melhoria de processo, com prazos mais curtos. Possui compatibilidade com o modelo CMMI, conformidade com as normas ISO/IEC 15504 (SPICE) e ISO/IEC 12207 e apresenta



um escalonamento de níveis de maturidade com mais granularidade nos níveis iniciais do que aquela utilizada nos modelos citados anteriormente, permitindo uma estratégia de implantação e evolução mais adequada às condições das empresas brasileiras. A dificuldade é que a avaliação conjunta de grupo empresarial objetiva a redução dos custos, porém há uma perda de foco, pois não há uma especificidade para cada empresa, e sim, um mesmo modelo de referência para todas elas (FILHO; ROCHA; TRAVASSOS, 2006).

O modelo PSP tem benefícios por requerer que o desenvolvedor mantenha uma base histórica de dados dos projetos anteriores, o que possibilita que os mesmos trabalhem totalmente focados no planejamento. Também, os próprios desenvolvedores podem autoanalisaçõs dentro do desenvolvimento, possibilitando que administrem seus prazos e custos, índices de qualidade e produtividade, os quais são cada vez mais atrativos para a gerência. O PSP é um modelo que auxilia o gerenciamento das equipes, pois possibilita que o administrador efetue comparações entre o seu pessoal e tenha sempre controle do que está sendo desenvolvido na empresa, além de ajudar os gerentes e os engenheiros a aprender a coletar dados, controlar e melhorar o planejamento e o gerenciamento do cronograma de projetos.

No modelo CMMI, as dificuldades são a pouca consideração à diversidade das empresas e a aplicação em pequenas empresas. No modelo SPICE, as dificuldades são devido à grande quantidade de informações, exigindo treinamento para a sua aplicação. Na norma ISO/IEC 12207, as dificuldades são apenas uma definição de taxonomia de processos e, também, não especifica os detalhes de como implementar ou executar as atividades e tarefas incluídas nos processos.

Fonte: a autora.



eu recomendo!



livro

Implantando a Governança de TI: Da Estratégia à Gestão de Processos e Serviços

Autor: Aguinaldo Aragon Fernandes e Vladimir Ferraz de Abreu

Editora: Brasport



Sinopse: este livro apresenta uma visão integrada e inovadora de Governança de TI, que pode ser adaptada para vários ambientes organizacionais. A partir de um modelo genérico, os autores detalham as etapas de planejamento, implementação e gestão da Governança de TI, abrangendo desde o plano do Programa de Governança de TI, passando pelo alinhamento estratégico da TI ao negócio, a elaboração do portfólio de TI, as operações de serviços de TI, os modelos de relacionamento com usuários e fornecedores, e, por fim, a gestão do desempenho e do valor da TI. Nesta nova edição, são analisados as características e os benefícios de mais de 30 modelos de melhores práticas que podem ser aplicados aos processos de TI, dentre eles: CobiT, ITIL, ISO/IEC 20000, USMBOK, os principais modelos do PMI (PMBOK, Gestão de Portfólio e Gestão de Programas), PRINCE2, ISO 27001 e 27002, eSCM-SP, eSCM-CL, CMMI, MPS.BR, BPM, CBOK, BABOK, BSC, Seis Sigma e outros modelos. Além disso, mostra os modelos agrupados por disciplina e representa, de forma clara, o relacionamento entre os modelos de melhores práticas.



conecte-se

O que é o CMMI?

Este artigo explica o CMMI e que, para entender bem o que é CMMI, é importante saber que existem três visões diferentes do modelo.

<https://promovesolucoes.com/cmmi-o-que-e-e-como-usar/#:~:text=CMMI%20%C3%A9%20uma%20sigla%20na,de%20Capacidade%20e%20Maturidade%20Integrado>

Este artigo explica o MPS.BR, os conceitos e processos e, também, que foram considerados as normas e os modelos internacionalmente reconhecidos, como o CMMI (Capability Maturity Model Integration), as normas ISO/IEC 12207 e ISO/IEC 15504, e a realidade do mercado brasileiro de software.

<https://blogdaqualidade.com.br/o-que-e-o-mps-br/>



REQUISITOS E AVALIAÇÃO DA QUALIDADE de software

PROFESSORA

Esp. Janaina Aparecida de Freitas

PLANO DE ESTUDO ▾

A seguir, apresentam-se as aulas que você estudará nesta unidade:

- Processo de avaliação da qualidade de produto de software
- Especificação da avaliação
- Requisitos para pacotes de software
- Requisitos de usabilidade na interface
- Requisitos para documentação de usuário

OBJETIVOS DE APRENDIZAGEM ▾

Compreender o processo de avaliação da qualidade de produto de software • Entender a especificação da avaliação na qualidade de software • Aprender os requisitos usados para pacotes de software • Aprender os requisitos de usabilidade na interface de software • Aprender os requisitos usados para a documentação de usuário.

INTRODUÇÃO



Caro(a) aluno(a), seja bem-vindo(a) à nossa terceira unidade, onde estudaremos os requisitos e a avaliação da qualidade de software. Esta unidade pretende que você compreenda os conceitos sobre o processo de avaliação da qualidade e os requisitos usados para pacotes de software, de usabilidade na interface e para a documentação de usuário.

Compreenderemos o processo de avaliação da qualidade de produto de software e que, para essa avaliação ser mais efetiva, é importante utilizar modelos de qualidade que permitam estabelecer e avaliar requisitos de qualidade.

Na sequência, passaremos a entender a especificação da avaliação na qualidade de software. Nesta etapa, temos que analisar, durante a avaliação do produto de software, três conceitos: a mediação, a pontuação e o julgamento.

Aprenderemos os requisitos usados para pacotes de software, como são oferecidos e liberados para uso no mercado e como é aplicada a norma NBR ISO/IEC 25051.

Veremos, também, os requisitos de usabilidade na interface de software e como a norma ISO/IEC 92411-11 é usada para esclarecer os benefícios de medir a usabilidade quando pensamos em desempenho e satisfação do usuário.

Por fim, aprenderemos os requisitos usados para a documentação de usuário, como ele deve ser descrito e como aplicar as normas ANSI/IEEE 1063 e a ISO 9127 para os requisitos mínimos. Veremos, também, as informações para usuários e compradores de software.

Vamos começar? Bons estudos!



PROCESSO DE AVALIAÇÃO DA QUALIDADE DE produto de software

Vimos que ter um produto de software, com qualidade, nem sempre é uma tarefa fácil, e que, às vezes, é bem complexa, pois envolve muitos fatores que devem ser avaliados, medidos e melhorados continuamente, durante todo o ciclo de vida do produto.

Para que a avaliação de um produto de software seja mais efetiva, é importante utilizar um modelo de qualidade que permita estabelecer e avaliar requisitos de qualidade e que o processo de avaliação seja bem definido e bem estruturado.

A série de normas ISO/IEC 14598 definem o processo de avaliação dos produtos de software e que pode ser utilizada em conjunto com a série ISO/IEC 9126 (GUERRA; COLOMBO, 2000).

As normas ISO/IEC 9126 e a série ISO/IEC 14598 foram reunidas na ISO/IEC 25000, que fornece orientação para o uso da nova série de normas internacionais denominadas Requisitos e Avaliação de Qualidade de Sistemas de Software (SQuaRE). Seu objetivo é fornecer uma visão geral dos conteúdos, características de qualidade e um guia para o uso dessas características. A norma ISO/IEC 25000 não está concluída, está em plena construção ainda.

A norma ISO/IEC 14598 está subdividida em seis partes, conforme o quadro a seguir:

Parte 1: visão geral	Define um processo de avaliação de produtos de software, em geral. Esta parte define os termos técnicos utilizados nas demais partes, contém requisitos gerais para especificação e avaliação de qualidade de software e esclarece os conceitos gerais. Adicionalmente, ela também fornece uma estrutura para avaliar a qualidade de quaisquer produtos de software e estabelece os requisitos para métodos de medição e avaliação de produtos de software.
Parte 2: planejamento e gestão	Etapa de apoio ao processo de avaliação de produto e está relacionada ao suporte e à gestão da avaliação corporativa ou departamental, fornecendo exemplos de medidas que correspondem a uma subcaracterística.
Parte 3: processo para desenvolvedores	Etapa de requisitos e orientações para o processo de avaliação de projeto na situação de desenvolvimento.
Parte 4: processo para adquirentes	Etapa de requisitos e orientações para o processo de avaliação de projeto na situação de aquisição e avaliação independente.
Parte 5: processo para avaliadores	Etapa de requisitos e orientações para o processo de avaliação de projeto na situação de incluir a avaliação de terceira parte.
Parte 6: documentação de módulos de avaliação	Etapa que é de apoio ao processo de avaliação de produto também, e trata da documentação de módulos de avaliação.

Quadro 1 - Norma ISO/IEC 14598 subdividida em seis partes

Fonte: adaptado de Guerra e Colombo (2000, p. 99).

Na Figura 1, temos a relação entre as normas da série, em que “cada processo de avaliação (partes 3, 4 e 5) pode ser usado em conjunto com o suporte à avaliação (partes 2 e 6)”, de acordo com Guerra e Colombo (2000, p. 100).

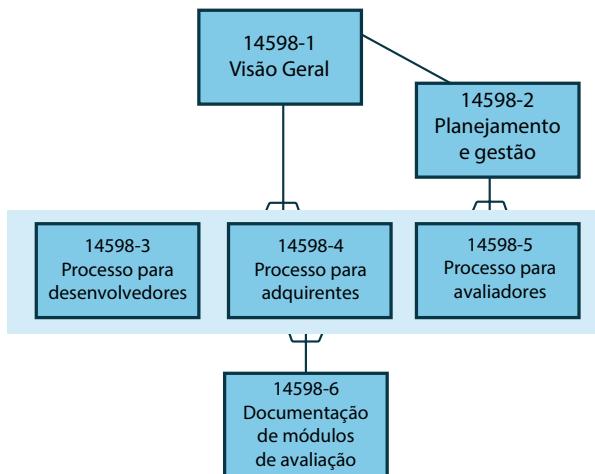


Figura 1 - Relacionamento entre as partes da ISO/IEC 14598

Fonte: Guerra e Colombo (2000, p. 100).

A norma ISO/IEC 9126 define um modelo de qualidade que possui um propósito geral, que mostra as características de qualidade e procura fornecer exemplos de métricas. A Figura 2 mostra o relacionamento entre as NBRs ISO/IEC 14598 e ISO/IEC 9126.

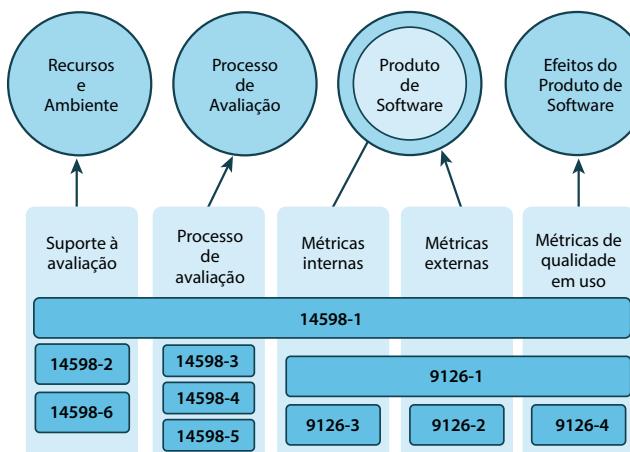


Figura 2 - Relacionamento entre as séries 9126 e 14598

Fonte: Guerra e Colombo (2000, p. 101).

Para Guerra e Colombo (2000, p. 101), temos que:



A norma NBR ISO/IEC 14598-1 contém conceitos de como avaliar a qualidade de software e define um modelo de processo de avaliação genérico; junto com as normas NBR ISO/IEC 14598-2 e NBR ISO/IEC 14598-6, estabelecem itens necessários para o suporte à avaliação; e junto com as normas NBR ISO/IEC 14598-3, NBR ISO/IEC 14598-4 e NBR ISO/IEC 14598-5, estabelecem processo de avaliação específico para desenvolvedores, acquirentes e avaliadores de software, respectivamente.

Especificamente, a norma NBR ISO/IEC 14598-3 trata do processo de avaliação para desenvolvedores de software e é aplicada em todas as fases do ciclo de vida de desenvolvimento. O foco dessa norma é a seleção de indicadores que preveem a qualidade do produto final por meio da medição da qualidade de produtos intermediários. A seguir, os requisitos e as recomendações indicados na norma:

Requisitos organizacionais	Infraestrutura que permita a obtenção de dados e modificações nos processos baseados na análise dos dados.
Requisitos de projeto	Processo de desenvolvimento disciplinado, que permita o planejamento e a condução das medições do software e sua avaliação; coordenação das atividades de avaliação com suporte; modelo de desenvolvimento similar a projetos anteriores, na sua organização de desenvolvimento com mesmo conjunto de atributos, deve, também, ser aplicado nos projetos, para permitir a análise dos dados.
Retroalimentação para a organização	Muitos métodos de análise de dados requerem dados de projetos desenvolvidos anteriormente, sob condições similares e com requisitos de qualidade comparáveis. A utilização de dados obtidos em outras experiências é o recurso para melhorar o processo de avaliação e os módulos de avaliação.

Quadro 2 - Requisitos e recomendações indicados na norma

Fonte: Guerra e Colombo (2000, p. 101).

As funções do desenvolvedor relacionados à norma NBR ISO/IEC 14598-3 são:

Procurar tornar os dados coletados disponíveis para a organização, para seu uso em outros projetos de desenvolvimento.

Procurar rever os resultados da avaliação e a validade do processo de avaliação, os indicadores e as medidas aplicadas.

Procurar melhorar os módulos de avaliação, incluir a coleta de dados sobre indicadores extras de maneira a validá-los para uso posterior, quando necessário.

Quadro 3 - Requisitos e recomendações indicados na norma

Fonte: Guerra e Colombo (2000, p. 102).

A norma NBR ISO/IEC 14598-4, segundo Guerra e Colombo (2000), trata do processo de avaliação para adquirentes de software e que deve ser usada para a aceitação ou a seleção de um produto de software. Essa norma, especificamente, contém requisitos e orientações que são usados para a avaliação da qualidade de produto quando houver a aquisição de produtos: (i) prateleira; (ii) produtos de software de uso geral; (iii) modificações em produtos de software existentes. Ela ajuda a estabelecer algumas regras para a decisão de aceitação de um único produto ou para produtos alternativos.

Conforme Guerra e Colombo (2000, p. 103), a “norma NBR ISO/IEC 14598-1 fornece requisitos e recomendações para implementação prática da avaliação de produto de software”. É uma norma que pode ser utilizada por fornecedores e compradores de software, laboratórios de avaliação de produtos de software, por usuários experientes e por entidades certificadoras, sendo que cada um tem um objetivo específico.

A norma NBR ISO/IEC 9126-1 pode ser usada quando pensamos em termos de características de qualidade, por possuir reconhecimento internacional. Porém não é muito prático o processo de mensurar essas características diretamente, precisamos desdobrá-las em atributos que permitam que sejam medidos e pontuados (GUERRA; COLOMBO, 2000). Esse processo de avaliação inclui quatro etapas que contêm dez atividades, conforme a Figura 3.

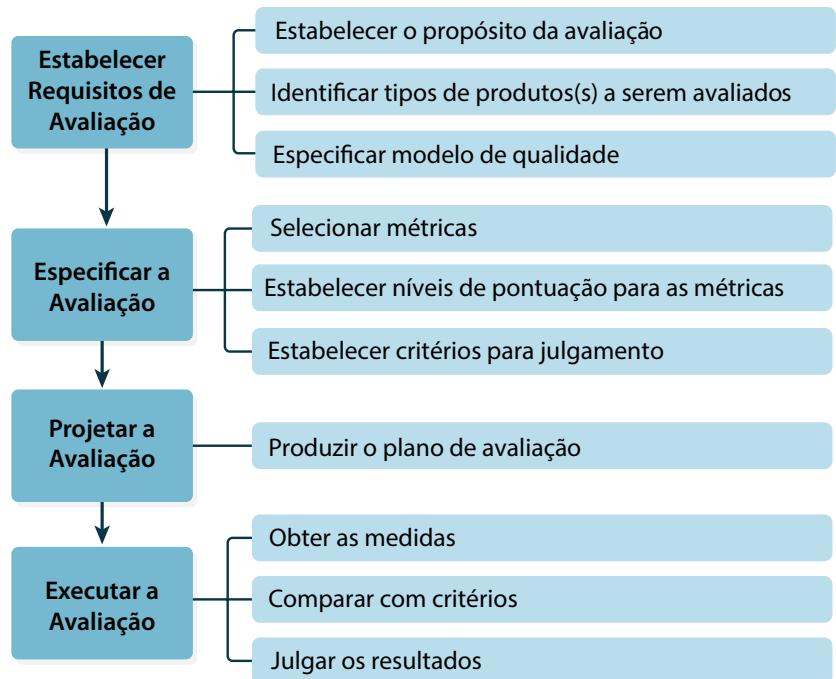


Figura 3 - Processo de avaliação segundo a norma NBR ISO/IEC 14598-1

Fonte: Guerra e Colombo (2000, p. 104).

A norma NBR ISO/IEC 14598-5 apresenta as características consideradas fundamentais no processo de avaliação, conforme o Quadro 4:

Repetível	A avaliação repetida de um mesmo produto, com mesma especificação de avaliação, realizada pelo mesmo avaliador, deve produzir resultados que podem ser aceitos como idênticos.
Reproduzível	A avaliação do mesmo produto, com mesma especificação de avaliação, realizada por um avaliador diferente, deve produzir resultados que podem ser aceitos como idênticos.
Imparcial	A avaliação não deve ser influenciada frente a nenhum resultado particular.
Objetiva	Os resultados da avaliação devem ser factuais, ou seja, não influenciados pelos sentimentos ou opiniões do avaliador.

Quadro 4 - Características fundamentais norma NBR ISO/IEC 14598-5

Fonte: Guerra e Colombo (2000, p. 103).

De acordo com Guerra e Colombo (2000, p. 104), o “processo de avaliação proposto pela norma NBR ISO/IEC 14598-5 é semelhante ao da parte 1, incluindo uma etapa a mais para as etapas da avaliação”, que são: (i) análise de requisitos da avaliação; (ii) especificação da avaliação; (iii) projeto da avaliação; (iv) execução da avaliação; (v) conclusão da avaliação. Na Figura 4, podemos ver as entradas e saídas de cada etapa da avaliação da norma.

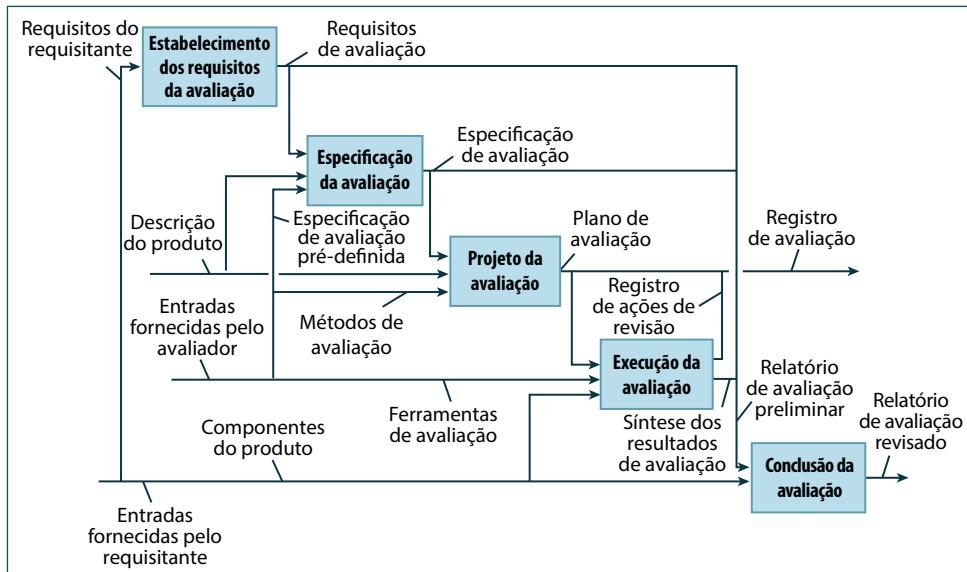


Figura 4 - Processo de avaliação segundo a norma NBR ISO/IEC 14598-5
Fonte: Guerra e Colombo (2000, p. 105).



pensando juntos

[Qualidade de software é] Uma gestão de qualidade efetiva aplicada de modo a criar um produto útil que forneça valor mensurável para aqueles que produzem e para aqueles que o utilizam.

(Roger Pressman e Bruce Maxim)

No Quadro 5, temos o detalhamento de cada uma das atividades do processo de avaliação, segundo a norma NBR ISO/IEC 14598-5.

Atividade	Descrição
Estabelecimento dos requisitos da avaliação	Etapa onde se descreve os objetivos da avaliação. Vários pontos de vista podem ser considerados, dependendo dos diferentes usuários do produto, tais como comprador, fornecedor, desenvolvedor e operador; devem ser descritos os objetivos e os requisitos da avaliação que dependem da necessidade do solicitante da avaliação, ou seja, do objetivo final.
Especificação da avaliação	Etapa onde se deve definir o escopo da avaliação e as medidas a serem executadas no produto submetido à avaliação, nos seus vários componentes. O nível de detalhes na especificação da avaliação deverá ser tal que, na sua base, a avaliação seja repetível e reproduzível.
Projeto da avaliação	Etapa onde devem ser documentados os procedimentos a serem usados pelo avaliador, para executar as medidas especificadas na fase anterior. O avaliador deve produzir um plano de avaliação que descreve os recursos necessários para executar a avaliação especificada, assim como a distribuição desses recursos nas várias ações a serem executadas.
Execução da avaliação	Etapa onde devem ser obtidos resultados da execução de ações para medir e verificar o produto de software, de acordo com os requisitos de avaliação, como descrito na especificação da avaliação e como planejado no plano. Ao executar essas ações, tem-se o rascunho do relatório de avaliação e os registros dela.
Conclusão da avaliação	Etapa onde se deve revisar o relatório da avaliação e disponibilizar os dados resultantes da mesma para o requerente da avaliação.

Quadro 5 - Detalhamento de cada uma das atividades do processo de avaliação, segundo a norma NBR ISO/IEC 14598-5. / Fonte: adaptado de Guerra e Colombo (2000, p. 105-106).

Na atividade de estabelecimento dos requisitos da avaliação, os solicitantes podem ser:

Equipes de desenvolvimento	Usam o resultado da avaliação para identificar ações corretivas e determinar estratégias de evolução.
Vendedores	Usam a qualidade como marketing.
Compradores	Para avaliar os produtos que estão competindo no mercado.
Usuários	Para obter confiança no produto.
Comunidade de software	Para identificar e validar os métodos mais adequados de avaliação e, assim, aumentar a credibilidade das técnicas.
Laboratórios de avaliação	Para desenvolver uma abordagem de avaliação mais consistente.

Quadro 6 - Solicitantes de uma avaliação

Fonte: adaptado de Guerra e Colombo (2000, p. 105-106).

A análise de requisitos procura definir os requisitos de qualidade do produto. Ela inicia a partir dos objetivos que foram definidos para a avaliação e que procuram traduzir, diretamente, o interesse do requisitante.



explorando Ideias

Deve-se avaliar a qualidade do produto liberado por diversas razões, tais como:

- Identificar e entender as razões técnicas para as deficiências e limitações do produto, que podem manifestar-se por meio de problemas operacionais e problemas de manutenção.
- Comparar um produto com outro, mesmo que indiretamente.
- Formular um plano de ação de como fazer o produto de software evoluir.

Um plano de avaliação quantitativa deve conter introdução, objetivos, características da qualidade aplicáveis, lista de prioridades, objetivos da qualidade, cronograma, definição de responsabilidades, categorias de medição, uso e análise de dados, relatórios e demais requisitos de interesse.

As responsabilidades de avaliação e de garantia da qualidade devem constar em um plano que vise ao uso e à melhoria da tecnologia de avaliação, implementação, transferência e avaliação da tecnologia de avaliação usada, e, por fim, o gerenciamento da experiência de avaliar.

Fonte: Andrade (2015, p. 50).



2 ESPECIFICAÇÃO DA AVALIAÇÃO

O método de avaliação, para Guerra e Colombo (2000, p. 109), é um “procedimento com que se descrevem as ações a serem realizadas por um avaliador, para obter o resultado da medição ou verificação especificadas, quando aplicadas num produto ou em componentes especificados” para um produto de software.

Na fase de especificação da avaliação de um produto de software, temos que analisar três importantes conceitos: medição, pontuação e julgamento (Figura 5).

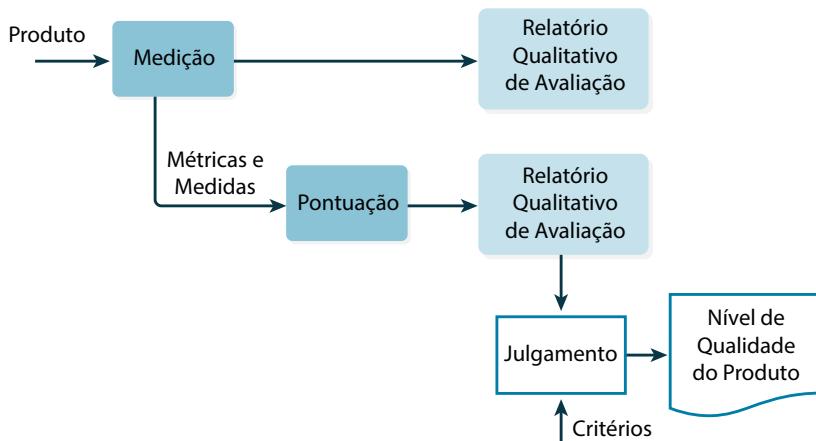


Figura 5 - Conceitos num processo de avaliação / Fonte: Guerra e Colombo (2000, p. 108).

Os conceitos num processo de avaliação, conforme a Figura 5:

- **Medição:** aplicação de uma medida de qualidade de software a um produto específico. A partir desta ação, é gerado o relatório qualitativo de avaliação.
- **Pontuação:** a partir das métricas e medidas obtidas, as informações são transformadas em um sistema numérico preestabelecido de pontuação. A partir desta transformação, é gerado outro relatório qualitativo de avaliação.
- **Julgamento:** a partir do relatório qualitativo, passamos à emissão de um juízo sobre o nível de qualidade do produto de software.

Por meio da medição de um produto de software, é possível:



Conhecer qualitativamente o nível de qualidade; após a pontuação, é possível conhecer quantitativamente o nível de qualidade. Assim, com a aplicação dos critérios de julgamento, é possível fazer o julgamento da qualidade. Para fazer medição em produto de software, é necessária a utilização de medida de qualidade de software. Esta, por sua vez, precisa de um “Método de avaliação” e de uma “Escala”, previamente escolhidos, que podem ser usados para determinar o valor que uma particularidade recebe em um produto de software específico (GUERRA; COLOMBO, 2000, p. 108).

O que seria essa “escala”? É um rótulo numérico atribuído aos atributos de uma entidade e que resultam em uma representação fidedigna do atributo. A escala deve possuir um conjunto de valores com propriedades bem definidas. O quadro, a seguir, tem exemplos de tipos de escalas que podem ser usadas:

Escala nominal	Corresponde a um conjunto de categorias. Produz dados qualitativos.
Escala ordinal	Corresponde a um conjunto ordenado de pontos de escala. Produz dados qualitativos.
Escala de intervalos	Corresponde a uma escala ordenada, com pontos de escala equidistantes. Produz dados quantitativos.
Escala de proporção	Possui pontos de escala equidistantes e, também, um zero absoluto. Produz dados quantitativos.

Quadro 7 - Exemplo de tipos de escalas / Fonte: adaptado de Guerra e Colombo (2000, p. 108).

Mas como determinar a qualidade de uma escala? Para Guerra e Colombo (2000, p. 109), a “qualidade de uma escala nominal, em sentido abrangente, descreve sua habilidade de refletir veridicamente uma medida obtida”. Esta descrição envolve a construção de uma semântica da escala, como adjetivos, palavras, frases e construção das sentenças, que nortearão a resposta correta do assunto avaliado.

O avaliador pode sofrer influências ao avaliar um produto de software, como: (i) preferências pessoais; (ii) experiências anteriores; (iii) conhecimento da área de aplicação do software. Assim, é recomendado que as medidas a serem utilizadas estejam claras e objetivas ao avaliador. Para Guerra e Colombo (2000, p. 109), a “questão da subjetividade do avaliador deve ser eliminada, procurando executar uma avaliação que seja imparcial e objetiva, com a utilização das diretrizes especificadas na norma NBR ISO/IEC 14598-5”.

Para realizar uma avaliação, é necessário estabelecer os critérios de aceitação, as medidas e os níveis de pontuação, conforme o quadro a seguir:

Critério de aceitação	Predeterminação dos níveis de pontuações considerados satisfatórios ou não satisfatórios, para um atributo de uma entidade, de acordo com a norma NBR ISO/IEC 14598-1, medida e nível de pontuação.
Medida	Compreende um método e uma escala de medição. Por escala, entende-se: um conjunto de valores com propriedades definidas; por medição, a determinação de um valor (que pode ser um número ou uma categoria) para um atributo de uma entidade.
Nível de pontuação	Pontuações de uma escala ordinal, que é utilizada para categorizar uma escala de medição.

Quadro 8 - Critérios de aceitação, as medidas e os níveis de pontuação

Fonte: adaptado de Guerra e Colombo (2000, p. 109).



A norma aponta três pontos de vista a serem considerados em uma avaliação da qualidade de software: do usuário, da equipe de desenvolvimento e do gerente. Cada um deles tem foco de interesse próprio: (i) o usuário está interessado no uso do software, e não nos seus aspectos internos. Ele quer saber se as funções requeridas estão disponíveis no software e o quanto eficiente, confiável e fácil de usar é o produto; (ii) a equipe de desenvolvimento está interessada tanto na qualidade dos produtos intermediários como na qualidade do produto final. O processo de desenvolvimento requer que o usuário e a equipe utilizem as mesmas características de qualidade de software, uma vez que elas se aplicam tanto para os requisitos como para a aceitação; (iv) a visão do gerente é comercial. Ele está interessado na qualidade de forma geral, e não em características específicas de qualidade. Sua responsabilidade é otimizar a qualidade, dentro das limitações de custo, recursos humanos e prazos.

Fonte: adaptado de Guerra e Colombo (2000, p. 110).

AULA

REQUISITOS PARA PACOTES de software

Para começarmos esta aula, entenderemos, primeiro, o termo pacote de software. De acordo com Guerra e Colombo (2000, p. 111), é um “conjunto completo e documentado de programas fornecidos a diversos usuários para uma aplicação ou função genérica”.

A seguir, alguns exemplos de pacotes de software:

- Processadores de texto.
- Planilhas eletrônicas.
- Bancos de dados.
- Software gráficos.
- Programas para funções técnicas ou científicas.
- Programas utilitários, entre outros.

Pacote de software é uma classe específica de produtos de software conhecida como produto de software comercial de prateleira e que recebeu a sigla COTS – *Commercial Off-The-Shelf*.

Segundo Pressman e Maxim (2016, p.70), componentes de software comercial de prateleira ou COTS são:



desenvolvidos por vendedores que os oferecem como produtos, disponibilizam a funcionalidade almejada juntamente com as bem definidas interfaces, sendo que essas interfaces permitem que o componente seja integrado ao software a ser desenvolvido.

No início do projeto, deve ser decidido se o software de aplicação será comprado ou construído. Hoje, é possível comprar software de prateleira (COTS) com uma grande variedade de domínios.

Alguns sistemas complexos de grande porte, na maioria das vezes, consistem, por exemplo, em uma mistura de componentes de prateleira e aqueles especialmente desenvolvidos. Para Sommerville (2011, p.193), os COTS “podem ser adaptados e ajustados aos requisitos dos usuários. Essa abordagem pode ser mais barata e rápida do que desenvolver um sistema em uma linguagem de programação convencional”.

Para a avaliação de pacotes de software como são oferecidos e liberados para uso no mercado, é aplicada a norma NBR ISO/IEC 25051.

A seguir, é mostrado o quadro de possíveis usuários dessa norma:

Fornecedores que estejam especificando os requisitos para um pacote de software; projetando um modelo para descrever produtos; julgando seus próprios produtos; emitindo declarações de conformidade; submetendo produtos à certificação ou à obtenção de marcas de conformidade.

Entidades de certificação que pretendam estabelecer um esquema de certificação por terceira parte.
Laboratórios de teste que terão de seguir as instruções durante a execução de testes para a certificação ou a emissão de marca de conformidade.
Laboratórios de avaliação que executam medição e pontuação de requisitos de qualidade de software.
Entidades de credenciamento que credenciam entidades de certificação e laboratórios de testes e de avaliação.
Auditores, quando julgam a competência de laboratórios de teste e de avaliação.
Compradores que pretendam comparar seus próprios requisitos com os aqui descritos; comparar os requisitos necessários para executar determinada tarefa com a informação presente nas descrições de produtos existentes; procurar por produtos certificados; verificar se os requisitos foram atendidos.
Usuários que se pretendam beneficiar com produtos melhores.

Quadro 9 - Possíveis usuários da norma NBR ISO/IEC 25051

Fonte: Guerra e Colombo (2000, p. 112).

A norma NBR ISO/IEC 25051 estabelece os requisitos para: (i) pacotes de software; (ii) documentação de teste; (iii) instruções para avaliação de conformidade de um pacote de software com relação aos requisitos preestabelecidos. Entretanto, alguns produtos de software podem necessitar de alguns requisitos adicionais, por exemplo, requisitos de segurança para software com diferentes níveis de acesso de usuários, e os requisitos de qualidade que são estabelecidos na norma NBR ISO/IEC 25051 podem ser utilizados para produtos de software que não sejam, especificamente, COTS. Para saber, pode ser verificado aqueles que podem ser usados para quaisquer produtos de software, entre os requisitos estabelecidos na norma (GUERRA; COLOMBO, 2000, p. 112).

Veja, na Figura 6, a estrutura básica do conteúdo da norma NBR ISO/IEC 25051 para um pacote de software:

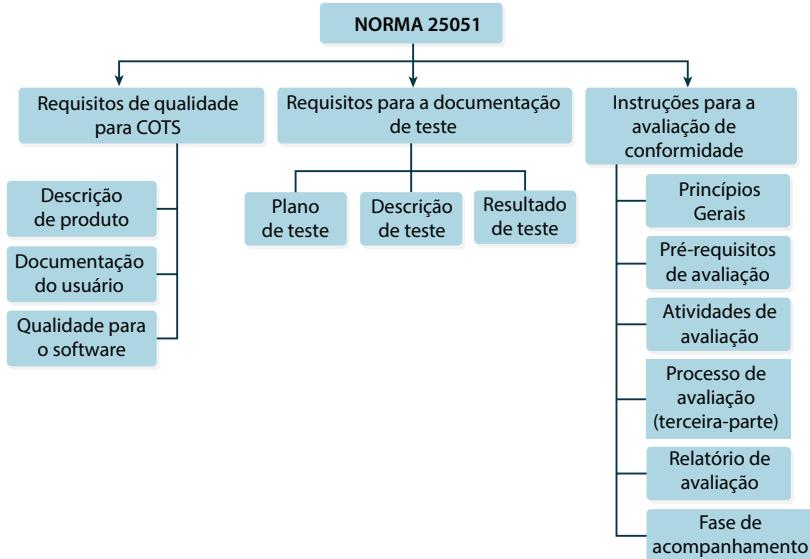


Figura 6 - Estrutura da norma NBR ISO/IEC 25051 / Fonte: Guerra e Colombo (2000, p. 113).

A seguir, apresentaremos uma descrição de cada um dos elementos da estrutura da norma NBR ISO/IEC 25051:

Requisitos de qualidade para COTS: um pacote de software deve possuir toda a documentação do pacote, que é composta pela: (i) descrição de produto; (ii) documentação do usuário; (iii) qualidade para o software.

Requisitos para a documentação de teste: deve possuir (i) plano de teste (ii); descrição do teste; (iii) resultado de teste.

Instruções para a avaliação de conformidade: deve possuir (i) princípios gerais; (ii) pré-requisitos da avaliação; (iii) atividades de avaliação; (iv) processo de avaliação (terceira parte); (v) relatório de avaliação; (vi) fase de acompanhamento.



O último componente dos requisitos de qualidade é referente ao software; para isso, utiliza as mesmas definições das características de qualidade da norma NBR ISO/IEC 9126-1. As características de Funcionalidade, Confiabilidade e Usabilidade são destacadas e devem ser verificadas com o uso do produto. Não há requisitos específicos para os aspectos de Eficiência, Manutenibilidade, Portabilidade e Qualidade em uso. Qualquer requisito declarado na documentação do pacote, referente às características citadas, deve estar em conformidade.

Fonte: Guerra e Colombo (2000, p. 116).

REQUISITOS DE USABILIDADE NA INTERFACE

Os projetos de interfaces para usuários e serviço ao cliente são os que geram mais valor agregado às empresas de computadores. As interfaces com usuários mostram a diferenciação de produtos, num mercado dominado pela tendência dos sistemas abertos (GUERRA; COLOMBO, 2000). Se o usuário tiver dificuldades em usar um software, de nada servirá.

A norma ISO/IEC 9241 procura esclarecer os benefícios de medir a usabilidade quando pensamos em desempenho e satisfação do usuário. Conforme Andrade (2015, p. 45), a “usabilidade dos computadores depende do contexto de uso e afirma que o nível de usabilidade alcançado dependerá das circunstâncias específicas nas quais o produto é usado”. Essa norma é o padrão internacional mais usado para a avaliação de usabilidade de sistemas interativos e considera o ponto de vista do usuário com relação ao produto que está sendo usado.

A norma ISO/IEC 9241 é dividida em 14 partes, mas focaremos na parte 11, que são as orientações sobre usabilidade de sistemas. A parte 11 (ISO, 2018) da norma define usabilidade como a capacidade com que um sistema ou produto pode ser usado por usuários para atingir objetivos específicos com eficiência, eficácia e satisfação, quando estes são usados em um contexto específico.

A norma ISO 9241-11 (ISO, 2018) define outros conceitos: (i) usuário: pessoa que tem interação com o sistema; (ii) contexto de uso: ambiente físico onde o sistema é usado; (iii) eficácia: quando os usuários têm seus objetivos específicos

atingidos com precisão e completeza, acessando as informações corretas; (iv) eficiência: quando os usuários têm seus objetivos específicos atingidos com relação à quantidade de recursos gastos; (v) satisfação: quando os usuários têm conforto e aceitabilidade dos sistemas.

A ISO 9241-11 enfatiza que a usabilidade dos computadores é dependente do contexto de uso e que o nível de usabilidade alcançado dependerá das circunstâncias específicas nas quais o produto é usado. O contexto de uso consiste de usuários, tarefas, equipamentos (hardware, software e materiais), e do ambiente físico e social, pois todos esses podem influenciar a usabilidade de um produto dentro de um sistema de trabalho. As medidas de desempenho e satisfação do usuário avaliam o sistema de trabalho como um todo, e, quando um produto é o foco de interesse, estas medidas fornecem informações sobre a usabilidade daquele produto no contexto particular de uso proporcionado pelo restante do sistema de trabalho. Os efeitos das mudanças em outros componentes do sistema de trabalho, tal como: tempo de treinamento do usuário ou melhoria de iluminação, podem também ser medidos pelo desempenho e satisfação do usuário (ANDRADE, 2015, p. 46).

A seguir, alguns benefícios que se pode ter ao adotar a norma NBR 9241-11:

A estrutura pode ser usada para identificar os aspectos de usabilidade e os componentes do contexto de uso a serem considerados no momento da especificação, projeto ou avaliação de usabilidade de um produto.

O desempenho (eficácia e eficiência) e a satisfação dos usuários podem ser usados para medir o grau em que um produto é usável em um contexto particular.

Medidas de desempenho e satisfação dos usuários podem fornecer uma base de comparação da usabilidade relativa de produtos, com diferentes características técnicas, que são usados no mesmo contexto.

A usabilidade planejada para um produto pode ser definida, documentada e verificada (ex.: como parte de um plano de qualidade).

Quadro 10 - Benefícios da norma NBR 9241-11 / Fonte: adaptado de Andrade (2015, p. 46).

A incorporação de características e alguns atributos podem melhorar a usabilidade dos sistemas e beneficiar os usuários em um contexto de uso específico. Segundo Guerra e Colombo (2000, p. 120), “um produto pode ter significativamente diferentes níveis de usabilidade, quando usado em diferentes contextos”.

Informações são necessárias para especificar ou medir a usabilidade:

- Uma descrição detalhada dos objetivos pretendidos.
- Uma descrição dos componentes do contexto de uso, incluindo usuários, tarefas, equipamento e ambientes. Esta pode ser uma descrição de um contexto existente ou uma especificação dos contextos pretendidos. Os aspectos relevantes do contexto e o nível de detalhes requeridos dependerão do escopo das questões apresentadas.
- Os valores reais ou desejados de eficácia, eficiência e satisfação para os contextos pretendidos.

Quadro 11 - Características e atributos para melhorar a usabilidade / Fonte: adaptado de Andrade (2015, p. 47).



explorando Ideias

O contexto de uso consiste em usuários, tarefas, equipamentos, tais como hardware, software e materiais; ambiente físico e social, que podem influenciar a usabilidade de um produto num sistema de trabalho. Medidas de performance e satisfação do usuário avaliam o sistema de trabalho como um todo. Quando um produto é o foco de interesse, essas medidas fornecem informações sobre a usabilidade daquele produto num contexto de uso particular, fornecido pelo resto do sistema de trabalho. Esses efeitos de mudanças em outros componentes do sistema de trabalho, tais como a quantidade de treinamento ao usuário, ou a melhoria da iluminação, podem ser medidos pela satisfação e performance do usuário.

Fonte: Guerra e Colombo (2000, p. 120).



REQUISITOS PARA DOCUMENTAÇÃO DE usuário



Um artefato importante e fundamental gerado no processo de desenvolvimento de um software é a documentação deste. Ele deve descrever o software, o modo como as atividades devem ser realizadas e como a avaliação e a modificação do software podem ser feitas sem muitas dificuldades.

Temos duas normas sobre documentos de usuário: a norma ANSI/IEEE 1063 (2001), que descreve os requisitos mínimos sobre a estrutura e o conteúdo da informação do documento do usuário de software, e a norma ISO 9127 (1988), que descreve a documentação de usuário e a informação para compradores de produtos de software.

Norma ANSI/IEEE 1063

Segundo a norma (IEEE, 2001), o documento do usuário de software é o artefato que fornece informações aos usuários na forma de material impresso ou em formato online. Ela se aplica à documentação que direciona o usuário em: (i) instalação do software; (ii) operação e uso do software; (iii) gerenciamento do software.

Além disso, o documento do usuário de software deve:

- (i) Identificar o produto de software.
- (ii) Descrever onde será aplicado.
- (iii) Identificar o público-alvo que utilizará o software.
- (iv) Determinar o conjunto de documentos para cada público-alvo.
- (v) Descrever o modo de uso de cada documento.

Temos dois modos de uso:

- (i) Modo instrutivo: pode ser usado para aprender sobre o software.
- (ii) Modo de referência: ajuda na busca rápida de algum ponto específico do software.

A norma fornece os requisitos para a inclusão de informação e como descrevê-la e apresentá-la na documentação, para que seja fácil de ler e compreender.

Norma ISO 9127

Descreve a documentação do usuário e a informação de capa do sistema, que são fornecidas para os compradores de pacotes de software.



Segundo a norma, a documentação do usuário fornece ao usuário final toda a informação necessária para instalar e executar o software. Tipicamente, essa documentação, em forma impressa ou eletrônica, vem incluída dentro do pacote de software; ela só estará disponível ao usuário após a compra do pacote de software. A informação de capa está disponível para o possível comprador, externamente ao produto, para que ele possa decidir sobre a aplicabilidade do software, mediante suas necessidades (GUERRA; COLOMBO, 2000, p. 123).

Veja, a seguir, a lista de itens recomendados para o conteúdo de uma documentação de usuário, conforme a norma ISO 9127:

Item da informação	Categoria*
Identificação do pacote	ESS
Conteúdo do pacote	ESS

Descrição funcional do software	ESS
Instalação do software	ESS
Utilização do software	ESS
Informação técnica do software	CON
Teste	OPT
Informação contratual	ESS
Glossário	CON
Índice	CON
Comentários de usuários	OPT

*ESS (Informação Essencial), OPT (Informação Opcional) e CON (Informação Condisional, se necessária).

Quadro 12 - Itens de informação recomendados para um documento de usuário

Fonte: Guerra e Colombo (2000, p. 122).

O objetivo principal da documentação é fornecer ao usuário ajuda e informações para o entendimento do objetivo, das funcionalidades, das características, da instalação, dos direitos e responsabilidades contratuais do software.

A norma ISO 9127 também sugere outro documento, denominado informação de capa. Este documento tem, como objetivo, permitir que os possíveis compradores do software possam avaliar a sua aplicabilidade quanto às necessidades. A seguir, alguns itens de informação recomendados para o conteúdo da Informação de capa:

- Identificação do pacote.
- Propósito e campo de aplicação.
- Ambiente de uso.
- Entrada e saída (restrições).
- Instruções para uso e informação suplementar.
- Informação contratual e contatos do fornecedor.
- Itens fornecidos.
- Padrões e leis e certificação independente.
- Código do produto, preço e marca registrada.

A documentação de usuário do software é considerada o conjunto completo de documentos e que devem estar disponíveis na forma impressa ou online e, também, sempre fornecidos juntos com a instalação para a utilização do sistema.



explorando Ideias

Medidas de eficiência relacionam o nível de eficácia alcançada ao dispêndio de recursos. Recursos relevantes podem incluir esforço mental ou físico, tempo, custos materiais ou financeiros. Por exemplo, a eficiência humana pode ser medida como eficácia dividida pelo esforço humano, eficiência temporal como eficácia dividida pelo tempo ou eficiência econômica como eficácia dividida pelo custo. Se o objetivo desejado for imprimir cópias de um relatório, então, a eficiência pode ser especificada ou medida pelo número de cópias usáveis do relatório impresso, dividido pelos recursos gastos na tarefa, tal como horas de trabalho, despesas com o processo e materiais consumidos.

Fonte: Andrade (2015, p. 49).

CONSIDERAÇÕES FINAIS

Caro(a) aluno(a), chegamos ao final da nossa unidade, em que aprendemos como aplicar normas a requisitos e avaliação da qualidade de software.

Ao longo da unidade, você aprendeu os conceitos do processo de avaliação da qualidade e os requisitos usados para os pacotes de software, para a usabilidade na interface e a documentação de usuário.

Na primeira aula, foi falado sobre o processo de avaliação da qualidade de produto de software. Estudamos que, para uma avaliação mais efetiva e completa do software, é importante utilizar modelos de qualidade que ajudam a estabelecer como será o processo de avaliação e dos requisitos de qualidade.

Na sequência, foi explicada a especificação da avaliação na qualidade de software. Nesta etapa, foi falado que, durante a avaliação do produto de software, temos três conceitos que devem ser analisados: a mediação, a pontuação e o julgamento.

Aprendemos, também, os requisitos usados para pacotes de software. Estes pacotes integram vários componentes do sistema que são disponibilizados aos usuários. Foi explicado como eles podem ser oferecidos e liberados para uso no mercado e como é aplicada a norma NBR ISO/IEC 25051.

Foram estudados os requisitos de usabilidade na interface de software e aprendido como a norma ISO/IEC 92411-11 pode ser usada para mostrar os benefícios de se medir a usabilidade quando avaliamos o desempenho e a satisfação do usuário pelo sistema.

Por fim, foram estudados os requisitos usados para a documentação de usuário. Aprendemos como ele pode ser descrito e como aplicar as normas ANSI/IEEE 1063, e a ISO 9127 para os requisitos mínimos da documentação e as informações para usuários e compradores de software.



1. “A avaliação de produtos de software deve ser objetiva, ou seja, baseada em observação e não em opinião. Também deve ser reproduutiva, de forma que avaliações do mesmo produto, para a mesma especificação de avaliação, executadas por diferentes avaliadores, produzam resultados aceitos como idênticos e repetíveis” (SILVA; ALMEIDA, [2020], on-line). Disponível em: <https://sites.google.com/site/a109iff/teste/iso-14598>. Acesso em: 28 out. 2020.

Considerando o texto anterior, temos que a norma ISO/IEC 14598 está subdividida em seis partes. Descreva a parte 2 e a parte 5 da norma.

2. “Uma avaliação de produto de software tem sido uma das formas empregadas por instalações que permitem ou adquirem software para obtenção de maior qualidade nestes produtos, sejam eles produtos completos ou partes a serem integradas num sistema computacional mais amplo. Para que a avaliação seja mais efetiva, é importante que utilize um modelo de qualidade que permita estabelecer e avaliar os requisitos de qualidade e também que o processo de avaliação seja bem definido e estruturado” (KOSCIANSKI et al., 1999, p. 4). Disponível em: https://www.researchgate.net/publication/326984869_Guia_para_Utilizacao_das_Normas_Sobre_Avaliacao_de_Qualidade_de_Produto_de_Software_-_ISOIEC_9126_e_ISOIEC_14598. Acesso em: 28 out. 2020.

Com base no texto exposto, especificamente, do que trata a norma NBR ISO/IEC 14598-3?

3. “As normas podem ser internacionais, regionais, nacionais e organizacionais em função da sua área de aplicação. Normas nacionais são editadas por uma organização nacional de normas. No Brasil, esta organização é a Associação Brasileira de Normas Técnicas - ABNT. Ela é reconhecida como Foro Nacional de Normalização” (KOSCIANSKI et al., 1999, p. 7). Disponível em: https://www.researchgate.net/publication/326984869_Guia_para_Utilizacao_das_Normas_Sobre_Avaliacao_de_Qualidade_de_Produto_de_Software_-_ISOIEC_9126_e_ISOIEC_14598. Acesso em: 28 out. 2020.



Com base no fragmento de texto apresentado, analise as afirmativas, a seguir, sobre as funções do desenvolvedor relacionados à norma NBR ISO/IEC 14598-3:

- I - Procurar tornar os dados coletados disponíveis para a organização, para seu uso em outros projetos de desenvolvimento.
- II - Procurar rever resultados da avaliação e validade do processo de avaliação, indicadores e medidas aplicadas.
- III - Procurar melhorar os módulos de avaliação, incluir a coleta de dados sobre indicadores extras, de maneira a validá-los para uso posterior, quando necessário
- IV - Procurar melhorar os requisitos e dados de avaliação, incluir a entrevista com usuários e cadastrar indicadores extras, de maneira a validá-los para uso posterior, quando necessário.

É correto o que se afirma em:

- a) I, apenas.
 - b) II e III, apenas.
 - c) I, II e III, apenas.
 - d) I, III e IV, apenas.
 - e) I, II, III e IV.
4. “A ISO é a Organização Internacional de Normalização (ou para Padronização) – do inglês: International Organization for Standardization. Ou seja, ela é a Instituição que reúne as normas de padronização de produtos e empresas na tentativa de manter e garantir a qualidade dos serviços e produtos” (ARDUINI, [2020], on-line). Disponível em: <https://certificacaoiso.com.br/certificacao-iso-as-principais-normas/>. Acesso em: 28 out. 2020.

Considerado o texto exposto, analise as afirmativas, a seguir, sobre as atividades do processo de avaliação segundo a norma NBR ISO/IEC 14598-5.

- I - Estabelecimento dos requisitos da avaliação: atividade que descreve os objetivos da avaliação.



- II - Especificação da avaliação: atividade em que se deve definir o escopo da avaliação e as medidas a serem executadas no produto submetido à avaliação, nos seus vários componentes.
- III - Projeto da avaliação: atividade em que devem ser documentados os procedimentos a serem usados pelo avaliador, para executar as medidas especificadas na fase anterior.
- IV - Execução da avaliação: devem ser obtidos resultados da execução de ações para medir e verificar o produto de software, de acordo com os requisitos de avaliação, como descrito na especificação da avaliação e como planejado no plano.

É correto o que se afirma em:

- a) I, apenas.
 - b) II e III, apenas.
 - c) I, II e III, apenas.
 - d) I, III e IV, apenas.
 - e) I, II, III e IV.
5. “As normas ISO impactam diretamente nos sistemas de gestão. Elas podem parecer totalmente distintas ou incompatíveis, mas é justamente o contrário disso. É possível implementar todos os sistemas de gestão citados acima, sem que um interfira no outro. Isso porque cada um está direcionado para a sua área, mas todos têm em comum a preocupação com a melhoria contínua da empresa” (ARDUINI, [2020], on-line). Disponível em: <https://certificacaoiso.com.br/certificacao-iso-as-principais-normas/>. Acesso em: 28 out. 2020.

Com base no texto apresentado, descreva os conceitos que são usados no processo de avaliação.



UMA ONTOLOGIA PARA O DOMÍNIO DE QUALIDADE DE SOFTWARE COM FOCO EM PRODUTOS E PROCESSOS DE SOFTWARE

Resumo

Qualidade de software envolve diversas perspectivas, sendo as duas principais a qualidade de processo e a qualidade de produtos de software. Ainda que realmente haja aspectos bastante diferentes em cada perspectiva, é possível identificar, também, aspectos comuns. De fato, em ambos os casos, trata-se de entidades que precisam ser mensuradas para ter sua qualidade avaliada. Este artigo apresenta uma ontologia do domínio de Qualidade de Software desenvolvida visando formalizar parcialmente o conhecimento envolvido nesse domínio, focando na captura dos aspectos comuns às perspectivas de qualidade de processos e de produtos de software.

Introdução

É fato que a qualidade dos produtos de software depende fortemente da qualidade dos processos utilizados em seu desenvolvimento e manutenção. Assim, a busca pela qualidade em software passa tanto pela qualidade dos processos quanto pela de produtos de software. No caso de produtos, é necessário avaliar, dentre outros, conformidade a padrões, atendimento a requisitos e consistência com outros artefatos produzidos no processo de software. Em relação a processos de software, o foco deve ser a melhoria contínua. Uma parte importante da garantia da qualidade é a avaliação. A avaliação sistemática da qualidade de processos, seus ativos



(atividades, ferramentas, procedimentos e recursos) e dos artefatos produzidos é essencial para o bom andamento de um projeto e para apoiar a implementação de estratégias de melhoria [Fuggetta 2000]. Contudo, uma avaliação sistemática não pode ser conduzida em bases meramente subjetivas. É necessário identificar características que sejam capazes de indicar a qualidade de uma entidade, medir essas características, analisar os resultados das medições e concluir sobre a necessidade de ajustes. Pode-se notar, portanto, que a medição e a avaliação têm um papel fundamental para se conhecer a qualidade dos processos e produtos de software de uma organização. Entretanto, o estudo acerca da qualidade de software tem sido sempre bastante segmentado de acordo com essas duas perspectivas: qualidade de processos e qualidade de produtos. Por exemplo, há diversas normas e modelos de qualidade, tais como CMMI-SW [SEI 2006], ISO/IEC 12207 [ISO/IEC 2008], ISO/IEC 15504 [ISO/IEC 2003] e MPS.BR [Softex, 2007], focando aspectos relacionados à qualidade de processos de software, enquanto outros focam a qualidade de produtos de software, tais como as normas ISO/IEC 9126 [ISO/IEC 2001] e a ISO/IEC 14598 [ISO/IEC 1999]. Ainda que realmente haja aspectos bastante diferentes em cada perspectiva, é possível identificar, também, aspectos comuns. De fato, em ambos os casos, trata-se de entidades que precisam ser mensuradas para ter sua qualidade avaliada. Assim, é importante tentar prover uma conceituação básica acerca de qualidade de software que se aplique tanto a produtos quanto a processos de software. Isso pode ser útil, por exemplo, para o desenvolvimento e integração de ferramentas de apoio à garantia da qualidade, avaliação e melhoria de processos e produtos de software.

Fonte: Moro e Falbo (2008, p. 37).



eu recomendo!



livro

Qualidade de Software na Prática. Como Reduzir o Custo de Manutenção de Software com a Análise de Código

Autor: Cleuton Sampaio

Editora: Ciência Moderna

Sinopse: a dívida técnica é resultado de baixa qualidade do código-fonte e sempre cobra juros a cada manutenção, na forma de aumento de prazo e custo. Veja como medir, reduzir e administrar a dívida técnica de projetos de software, pela análise de código. Neste livro, são mostradas técnicas e ferramentas open source que permitirão avaliar e controlar a qualidade de seus projetos de software, entre elas: código autodокументado; refatoração; princípios de projeto orientado a objetos; métricas comuns de qualidade de software; cobertura de testes; ferramentas Sonar, PMD, Checkstyle, Findbugs, Cobertura e outras.



livro

Engenharia de Software: Qualidade e Produtividade com Tecnologia Autor: Kechi Hirama

Autor: Elsevier

Editora: Nome da Editora

Sinopse: este livro apresenta uma visão moderna e inovadora do conceito de engenharia de software. Em vez de explorar, à exaustão, tópicos tradicionais da disciplina, como fazem os títulos considerados clássicos da área, a obra dá ênfase aos conceitos requisitados pelos entrantes neste universo, tais como documentação de processos, novas formas de padrões arquiteturais, gerência de projetos OO, gerência de requisitos, e, por fim, o novíssimo manifesto ágil (Scrum) e os projetos e implementação de serviços SOA. A importância do livro se concretiza pelo fato de apresentar todos os temas de maneira direta e totalmente voltada para a realidade dos estudantes e professores brasileiros.





MÉTRICAS DE SOFTWARE

PROFESSORA

Esp. Janaina Aparecida de Freitas

PLANO DE ESTUDO ▼

A seguir, apresentam-se as aulas que você estudará nesta unidade:

- Conceitos básicos de métricas de software
- Framework para métricas de produtos
- Métricas para o modelo de requisitos
- Métricas para o modelo de projetos
- Revisão de software

OBJETIVOS DE APRENDIZAGEM ▼

Entender os conceitos básicos das métricas de software • Expor os frameworks para as métricas de produto • Compreender as métricas utilizadas para o modelo de requisitos de software • Compreender as métricas para o modelo de projetos de software.

INTRODUÇÃO



Olá, alunos(as), nesta unidade, começaremos a entender a importância das métricas de software para a qualidade do produto. Por isso, iniciaremos expondo os principais conceitos relacionados a essas métricas. Uma métrica de software é algo que poderá ser medido no software que foi desenvolvido, desde o tamanho de um sistema em linhas de códigos até o número de defeitos que foram encontrados durante os testes realizados.

Após entender os conceitos básicos das métricas, apresentaremos um framework para a métrica de produto, conhecido como HEART. Este framework utiliza a métrica de experiência com o usuário e ajuda na escolha de métricas que têm a finalidade de medir essa experiência com o software.

Dentre as métricas apresentadas, abordaremos aquela para o modelo de requisitos, a qual se utiliza destes que foram estabelecidos como base para o projeto e que proporciona informações sobre a qualidade do modelo de análise efetuado no início do desenvolvimento do software.

A outra métrica estudada será aquela para o modelo de projeto, a qual proporciona melhor visualização e pode ajudar a evolução do projeto para mais qualidade. É uma métrica importante e que possui muita relevância em relação à obtenção da qualidade.

Por fim, exporemos a importância da revisão de software e como realizar essa revisão, aplicando desde técnicas de inspeção até técnicas de pair programming.

Vamos começar a aprender mais sobre este assunto? Bons estudos!

1 CONCEITOS BÁSICOS DE MÉTRICAS DE software



Já sabemos que entender o que é uma métrica de software se tornou cada vez mais importante no trato com ele. Sendo assim, a partir de agora, trabalharemos com conceitos básicos e que nos ajudarão a prover melhorias em produtos de software.

Uma métrica de software nada mais é do que algo que poderá ser medido no software criado, por exemplo, o tamanho de um produto em linhas de códigos, o número de defeitos encontrados, entre outros (SOMMERVILLE, 2011). Realizar este tipo de tarefa fará com que você descubra e corrija problemas potenciais e que podem se tornar catastróficos. Assim, aprenderemos, aqui, algumas métricas que nos auxiliarão em todo o processo. Começaremos pela métrica de controle e a métrica de previsão.

A métrica de controle suporta processos relacionados ao gerenciamento, ao processo de software em si, por exemplo, o tempo necessário para reparar os defeitos encontrados. Já a métrica de previsão ajuda a prever as características do software e é associada com o software em si e, também, é conhecida como métricas de produto. Podemos citar, como exemplo dessa métrica, o número de atributos e operações de uma classe de objeto.



As métricas de produto são métricas de previsão usadas para medir atributos internos de um sistema de software. O tamanho de sistema, medido em linhas de código, ou o número de métodos associados a cada classe de objeto são exemplos de métricas de produto.

Fonte: Sommerville (2011, p. 56).

No contexto da engenharia de software, poderemos encontrar os termos **medidas**, **métricas** e **indicadores**, que podem se assemelhar, mas que precisam ser diferenciados e entendidos. Em relação à **medida**, proporciona-se uma indicação de quantidade, capacidade ou tamanho de algum atributo ou processo; já a **métrica** está relacionada a medidas individuais, como o número de erros encontrados em um teste de unidade. Por fim, temos o **indicador**, que é uma métrica, ou mesmo uma combinação de métricas, com informações sobre um processo do software.

Bem, agora que vimos um pouco mais desses termos, adentraremos o assunto de como as métricas influenciam os produtos. Durante muito tempo, vêm se desenvolvendo métricas que facilitam o entendimento de um software, cada um com uma visão diferente do que é uma complexidade.

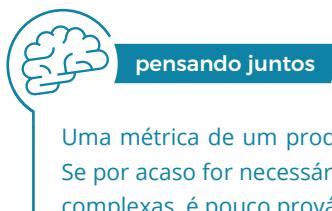
Para compreender melhor, faremos uma analogia em relação a uma moto nova. Ao ser analisada, observadores podem dar ênfase à carroceria, outros, à parte mecânica, e assim por diante. Cada característica é diferenciada, o que tornará complicado dizer qual é o valor mais atraente para a aquisição, por exemplo. Em relação ao software, podemos dizer que é a mesma situação: sabemos que ele deve ser medido e controlado pela complexidade de cada software, e como é complicado obter valor comum de qualidade, devemos, assim, obter medidas diferenciadas de atributos do programa. Desta forma, essas medidas serão utilizadas como indicadores de qualidade independentes dos modelos de requisitos e projeto, ou, nos primeiros estágios do processo de software, fornece algo consistente e objetivo para avaliar a qualidade do software (ANDRADE, 2015).

Temos que a medição é essencial quando o que desejamos é ter uma qualidade do software. Desta forma, podemos expor, aqui, alguns princípios básicos sobre a medição (PRESSMAN, 2011):

- Formulação: a criação de medidas e métricas de software apropriadas para a representação do software considerado.
- Coleção: o mecanismo usado para acumular dados necessários para criar as métricas e fórmulas.

- Análise: a computação das métricas que resultam em informações sobre a qualidade de representação.
- Feedback: recomendações derivadas da interpretação de métricas de produto transmitidas para a equipe de software.

É importante entender, também, que as métricas de software só serão úteis se forem verificadas e validadas e demonstrarem ser importantes para o processo, desta forma, é importante que uma métrica tenha uma propriedade matemática, com intervalos numéricos de avaliação, por exemplo.



Uma métrica de um produto será usada, somente, se ela for clara e fácil de computar. Se por acaso for necessário realizar dezenas de contagens e a execução de computações complexas, é pouco provável que a métrica seja amplamente adotada.

Outro ponto importante é a análise realizada de uma característica do software e que aumenta a ocorrência de peculiaridades; este valor deve aumentar ou diminuir da mesma maneira. Por fim, cada métrica deve ser analisada e validada para ter um valor real de interesse, independentemente de qualquer outro fator que possa surgir.

Desta forma, formular, caracterizar e validar são de extrema importância, assim, sempre que possível, é interessante que você analise dados de forma automática, aplique técnicas válidas bem como diretrizes e recomendações que deverão ser estabilidades para cada métrica.

Existem métricas significativas em qualquer parte do processo de software. Tratam-se das métricas conhecidas como Meta/Questão/Métrica (GQM), nas quais são enfatizados: a necessidade de se estabelecer um objetivo; uma característica do produto que será avaliada; a definição de um conjunto de questões que devem ser respondidas para atingir determinado objetivo e, por fim, identificar as métricas bem formuladas que ajudam a responder a essas questões.

Diversas métricas já foram elaboradas e aplicadas em programas 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 e que devem ser, segundo Pressman (2011, p. 343):



- **Simples e computáveis.** Deverá ser relativamente fácil aprender a derivar a métrica, e sua computação não deve demandar esforço ou tempo fora do normal.
- **Empiricamente e intuitivamente persuasiva.** A métrica deverá satisfazer as ideias do engenheiro 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. Um terceiro independente 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 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 deverão 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.

A maioria das métricas satisfaz aos atributos citados, mas uma ou outra podem não satisfazer, e isso não é sinal de rejeição da métrica. As métricas proporcionam informações úteis e valor distinto, mesmo que não satisfaçam um atributo perfeitamente.





FRAMEWORK PARA MÉTRICAS DE PRODUTOS

FRAMWORK

As métricas relacionadas aos produtos de software são, usualmente, utilizadas para medir atributos de um sistema, como: o tamanho, o número de linhas ou métodos que são associados a classes. Mas não é só esse tipo de métrica relacionada ao software que podemos realizar; também podemos aplicar métricas em relação ao sucesso de um produto, se está como esperado ou não.

Nesse sentido, falaremos um pouco sobre um *framework* que está sendo muito utilizado e que foi desenvolvido pelo Google e pela *Digital Telepathy*. Normalmente, empresas medem só o desempenho do sistema e deixam de lado métricas relacionadas ao sucesso do software – o que é importante, pois entender que o usuário é parte importante do processo poderá trazer benefícios ao seu software.

O *framework* de que falamos é o HEART, que utiliza a métrica de experiência com o usuário. Esse *framework* fornece várias métricas que são centradas no usuário e cuja finalidade é mediar a experiência do usuário com o software (RODDEN; HUTCHINSON; FU, 2010).

Você deve estar se perguntando o que significam as siglas da palavra HEART. Ela vem do inglês, com o conjunto de palavras ***Happiness, Engagement, Adoption, Retention e Task success***. Sabemos que, para o negócio dar certo, devemos considerar o cliente o coração do negócio e, por isso, esse *framework* recebe este nome. Estas cinco palavras formam cinco itens fundamentais que resultam numa métrica. Esta, por sua vez, indica que o usuário está satisfeito; mas, além disso, são

métricas que também auxiliarão a equipe a verificar o desempenho do sistema, o crescimento e o sucesso.

Vamos aprender o que cada item faz, necessariamente, neste *framework*? Começaremos pela palavra **Happiness**. Traduzindo do inglês, podemos dizer que significa “felicidade”, sendo assim, a métrica é 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. Nesta métrica, é importante ter um bom relacionamento entre a equipe de programação com o time de suporte, afinal, é este que sabe das principais reclamações, o que usuários gostam e o que não gostam do sistema. Assim, pode-se realizar *check-lists* e realizar as melhorias do software para aumentar a satisfação do usuário (FIORINI, 2019).

Já a letra E do *framework* trata do **Engagement**, ou traduzindo, engajamento. Esta métrica mede, como o próprio nome diz, o engajamento do usuário em utilizar o software com ele. Mede-se se o usuário está, realmente, utilizando ou não tal função de um software e qual o motivo real. Sabendo disso, é possível alterar o sistema e melhorar as funcionalidades para que o usuário possa voltar a utilizar uma função e, também, melhora a funcionalidade de seu sistema.

Adoption, mais um item do acrônimo HEART, ao ser traduzido, significa adoção. Esta métrica verifica o uso de novas funcionalidade criadas no sistema e quantos novos usuários começaram a utilizar a função criada; ela é importante, pois captura usuários para o sistema. Imagine um usuário utilizando um sistema antigo, o qual já sabe utilizar tudo, mudando para um novo sistema? Claro que ele relutará, mas, ao realizar essa métrica, você poderá verificar quantos usuários antigos estão aceitando o sistema e, também, verificar o porquê de alguns não estarem aceitando (FIORINI, 2019).

Temos, ainda, o item **Retention**, ou ainda, retenção. Nesta métrica, podemos verificar o comportamento de todos os usuários do sistema frente às mudanças que sempre acontecem no software. Para essa métrica, sempre respondemos algumas questões importantes, como: qual a eficácia de um software em segurar um usuário com as mudanças? Qual a frequência de utilização do sistema?

Por fim, o último item, **Task success**, ou seja, sucesso da tarefa. A métrica auxiliará na verificação do quanto é difícil um usuário realizar e terminar uma determinada tarefa no sistema, e se a solução encontrada para determinada função está de acordo com a necessidade do usuário (RODDEN; HUTCHINSON; FU, 2010).

Com todos esses itens, além de você se aproximar do usuário, também otimizará o desempenho, reduzindo e eliminando etapas de construção ou atualização

de funções e, claro, facilitará os processos de criação do sistema.

Ok! Entendemos o que significa cada item da palavra HEART, mas como utilizar este *framework*? Bem, antes de mais nada, veremos como é sua estrutura, que podemos observar na imagem a seguir:

	Goals	Signals	Metrics
Happiness			
Engagement			
Adoption			
Retention			
Task Success			

Figura 1 - Estrutura do *framework* HEART / Fonte: a autora.

Lembrando que esse processo facilita a identificação de métricas a serem utilizadas. Veja que, além dos itens principais do *framework*, possuímos, também, algumas colunas nomeadas como: **Goals (Objetivos)**, **Signals (Sinais)** e **Metrics (Métricas)**.

Entenderemos o que é cada uma dessas colunas. Os **objetivos**, como o próprio nome diz, é expor quais objetivos você quer alcançar no resultado final. Aqui, você deve lembrar de ser bem claro(a) no que quer, para isso, há dicas que podem auxiliar na montagem do *framework*, por exemplo, entender que os objetivos de um projeto podem ser diferentes dos objetivos de software. Você não pode perder muito tempo nesta etapa, encontre as métricas relevantes de forma rápida (RODDEN; HUTCHINSON; FU, 2010).

Os **sinais** são onde os objetivos serão mapeados e que indicarão que você está tendo sucesso no que fez no software. Nos sinais, ainda, devem ser definidos os níveis de dificuldade, a agilidade e o custo de cada um. Como dica, neste item, devemos entender que, talvez, seja mais fácil encontrar falhas do que, propriamente, o sucesso de um sistema.

Em **métricas**, serão definidas as métricas a serem utilizadas. Agora, fica mais fácil, já que definimos os objetivos e devemos levar em consideração as melhores métricas a serem aplicadas. Foque em métricas que lhe aproximem do objetivo final, o que evitárá esforços em relação a implementações desnecessárias.

Em relação aos itens, como devemos aplicar exemplos de métricas? Bem, a respeito do H, *Happiness*, você deve analisar como os usuários se sentem, desta forma, alguns exemplos de métricas que podem ser aplicadas é em relação à satisfação, à percepção de facilidade do uso do sistema, e assim por diante (FIORINI, 2019).

Para o E, *Engagement*, você deve verificar com que frequência os usuários voltam a utilizar seu sistema, desta forma, podemos aplicar algumas métricas que analisem o número de visitas de usuários na semana, o número de ações realizadas no sistema, de relatórios gerados etc.

Em A, *Adoption*, você deve verificar quantos usuários concluem suas tarefas e como é a regularidade da execução delas. Como métricas, podemos aplicar as que verificam atualizações de versões recentes, novos usuários ao sistema etc.

Para o R, *Retention*, devemos verificar a quantidade de usuários que volta a utilizar o sistema. Desta forma, podemos utilizar métricas que verificam o número de usuários do sistema, quantos deixaram de utilizar etc.

E, por fim, o T, *Task success*, que visa verificar se a sua funcionalidade criada no sistema foi satisfeita para, assim, um usuário realizar a sua tarefa de forma clara e fácil. Para isso, podemos aplicar métricas que verificam o sucesso de resultados obtidos ou métricas relacionadas a erros existentes ao finalizar uma ação no sistema (FIORINI, 2019).

A seguir, temos um exemplo de HEART aplicado e que facilitará o entendimento deste *framework*. Vamos observar o quadro a seguir:

	Goals	Signals	Metrics
Happiness	Usuários acham o sistema útil e fácil de usar.	Responder pesquisas avaliando o serviço e deixando feedbacks.	Ranking de satisfação e número de avaliações.
Engagement	Usuários continuam a executar e a utilizar as atividades.	Gastando tempo no sistema.	Tempo de utilização e frequência média de sessões.

	Goals	Signals	Metrics
Adoption	Novos usuários utilizam o sistema.	Criando novos usuários. Usando features.	Número de registros. Adoção de features.
Retention	Usuários voltam para utilizar o software.	Ativos no aplicativo.	Taxa de erros.
Task success	Usuários completam objetivos de forma rápida e fácil.	Completando tarefas.	Número de quedas.

Quadro 1 - Exemplo de HEART / Fonte: a autora.

Analizando a montagem do *framework*, é realmente importante e útil ter métricas que facilitam a qualidade do software e a qualidade de experiência com o usuário. Por isso, entenda e reflita cada passo para a sua criação.



Como vimos, existem vários frameworks que auxiliam a realização de métricas em produtos de software. O link a seguir apresenta um artigo responsável pelas métricas relacionadas à linha de produto de software. Conecte-se e adquira mais conhecimentos sobre as métricas de produtos de software.

<https://www.devmedia.com.br/conheca-o-sdmetrics-framework/31697>





MÉTRICAS PARA O MODELO DE REQUISITOS

Trabalhar com métricas atinge desde níveis fáceis até os mais difíceis, depende de como são aplicadas. Por isso, é importante que todas as fases do desenvolvimento do software sejam compreendidas desde o início, e uma dessas fases é o levantamento de requisitos, que passaremos a tratar como modelo de requisitos.

O modelo de requisitos é onde estes são formulados e é estabelecida uma base para o projeto. Assim, as métricas de produto proporcionam informações sobre a qualidade do modelo de análise (PRESSMAN, 2011). Existem poucas métricas que analisam as especificações, mas há aquelas que podem ser adaptadas para realizar uma estimativa de projeto e, claro, aplicá-las.

Podemos começar pela métrica baseada em função, utilizada para medir a funcionalidade que o sistema fornecerá. Essa métrica pode ser empregada para estimar um custo ou trabalho, para projetar, codificar e testar determinado software; também, para prever o número de erros durante um teste e, por fim, prever o número de componentes ou linhas projetadas do sistema.

Pontos de função são derivados de medidas calculadas por valores de domínio de informações, que são definidos em diversas formas, como o número de entradas externas (EEs), as quais são originadas de um usuário e transmitidas de outra aplicação. Temos, também, o número de saídas externas (EOs), onde cada saída é formada por dados de uma aplicação que fornece informações para usuários. Já o número de consultas externas (EQs) são aquelas definidas como

entradas que resultam em respostas imediatas de um software. No número de arquivos lógicos internos (ILFs), cada arquivo lógico de dados de um aplicativo é mantido por meio de entradas externas; por fim, no número de arquivos de interfaces externos (EIFs), cada arquivo de interface reside fora da aplicação, mas fornece informações usadas na aplicação.

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

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

Em que os F_i são fatores de ajustes de valores baseados em respostas para algumas questões.



explorando Ideias

Os fatores de ajuste de valor são usados para fornecer uma indicação da complexidade do problema. Devemos questionar:

1. O sistema requer salvamento (backup) e recuperação confiável (recovery)?
2. São necessárias comunicações de dados especializadas para transferir informações para a aplicação ou da aplicação?
3. Há funções de processamento distribuído?
4. O desempenho é crítico?
5. O sistema rodará em um ambiente operacional existente e intensamente utilizado?
6. O sistema requer entrada de dados online?
7. A entrada online de dados requer que a transação de entrada seja composta em múltiplas telas ou operações?
8. Os ILFs são atualizados online?
9. As entradas, as saídas, os arquivos ou as consultas são complexos?
10. O processamento interno é complexo?
11. O código é projetado para ser reutilizável?
12. A conversão e a instalação estão incluídas no projeto?
13. O sistema é projetado para múltiplas instalações em diferentes organizações?
14. A aplicação é projetada para facilitar a troca e o uso pelo usuário?

Fonte: a autora.

Para entendermos melhor como se dá a métrica por pontos de função, representaremos um modelo simples de um diagrama de fluxo de dados, de um software chamado CasaSegura. Assim, analisemos a imagem a seguir:

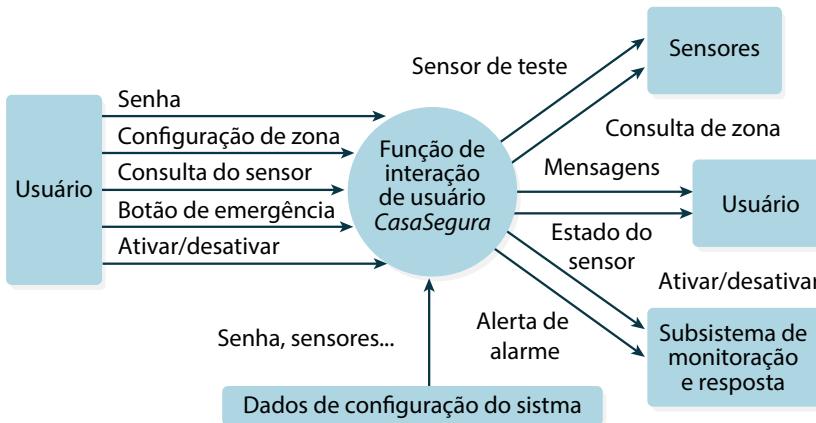


Figura 2 - Modelo de fluxo de dados software CasaSegura / Fonte: Pressman (2011, p. 545).

Analizando o diagrama, podemos verificar algumas informações importantes, como três entradas externas, as quais são a senha, o botão de emergência e ativar/desativar. Temos, ainda, duas consultas externas, sendo consulta de zona e consulta de sensor. É identificado um arquivo de configuração de sistema, que é o ILF e, também, são apresentadas duas saídas externas: mensagens e estado do sensor. Por fim, temos, como EIFs, o sensor de teste, a configuração de zona, o ativar/desativar e o alerta de alarme.

A partir destes dados, podemos computar os pontos de função, como podemos ver na imagem:

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

Figura 3 - Computando pontos de função / Fonte: Pressman (2011, p. 546).

Com base no valor encontrado, pode-se estimar o tamanho total relacionado à função de interação de usuário do software CasaSegura.

Podemos, também, citar as métricas para qualidade de especificação, em que determinadas características são usadas para avaliar a qualidade do modelo de requisitos, como totalidade, exatidão, entendimento, disponibilidade, modificação, rastreabilidade, peculiaridade, repetibilidade, consistência interna e externa, brevidade, precisão e reutilização.



É inegável que todo projeto tenha, como essencial iniciativa, a qualidade, por isso, para a construção de um projeto, é importante ter medidas, padrões ou métricas que satisfaçam a essa qualidade. Com isso em mente, analisaremos algumas métricas importantes relacionadas a modelos de projetos, que proporcionam melhor visualização e podem ajudar a evolução do projeto para mais qualidade.

Podemos começar com a **métrica de projeto da arquitetura**, cujos focos são a arquitetura do programa e a eficácia dos módulos presentes dentro da arquitetura do software. Esta técnica não requer conhecimento do funcionamento dos módulos ou componentes de software, mas trata as medidas de complexidade para realizar a métrica.

As **medidas de complexidade de projeto de software** são a complexidade estrutural, a complexidade de dados e a complexidade de sistema. À medida que cada valor de complexidade aumenta, a complexidade global da arquitetura do

software também tende a aumentar. Isto nos leva a um fato importante, em que temos mais probabilidade de o trabalho de integração e de teste também aumentar.

Outra métrica para os modelos de projetos é a **métrica para projeto orientado a objeto**. Este é importante e, normalmente, difere um projetista experiente de um novato. Em um projeto de software orientado a objetos, normalmente, um projetista sabe caracterizar um sistema de maneira que implemente o que o cliente necessita. Quando aumentamos a complexidade desses sistemas é que vemos a diferença entre os projetistas, por este motivo, há características importantes que um projeto orientado a objetos deve ter, as quais abordaremos a partir de agora.

A princípio, citaremos a característica **tamanho**, que é definida em termos de população, volume, comprimento e funcionalidade. Em relação à **população**, contam-se classes ou operações; que também são medidas em relação ao volume, com a diferença de serem coletadas em algum instante de tempo. Para o **comprimento**, é medida a cadeia de elementos de projetos interconectados; para a funcionalidade, está relacionado a um valor fornecido ao cliente por uma aplicação.

Já para a característica **complexidade**, é medida em relação às características estruturais, verificando como classes se comportam e se relacionam com outras classes. Na característica **acoplamento**, são analisadas as conexões, ou seja, as colaborações entre classes e mensagens trocadas entre os objetos.

Na característica **suficiência**, devemos analisar se um componente do projeto é suficiente para expor todas as propriedades do objeto de domínio da aplicação. Já na **totalidade**, são considerados vários pontos de vista sobre algo, se há uma implicação indireta no grau com que a abstração ou o componente pode ser reutilizado, diferentemente da suficiência, que compara a abstração do ponto de vista da aplicação corrente.

Para a característica **coesão**, podemos dizer que é como em tudo que fazemos: deve-se analisar se todas as operações estão funcionando em conjunto, se foram projetados de maneira correta, a fim de atingir um objetivo final. Devemos entender que a **coerência** faz parte do domínio do problema ou projeto, e deve ser bem analisado. Já na **similaridade**, é analisado o grau de similaridade entre duas ou mais classes, em relação à estrutura, à função, ao comportamento ou à finalidade.

Por fim, temos as características **originalidade** e **volatilidade**. Na originalidade, 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. Na volatilidade, é medida a possibilidade de que uma alteração venha a ocorrer, já que, em um projeto, podem ocorrer diversas alterações em seu curso normal. Temos que as métricas de produto para sistemas do tipo orientado a objetos podem ser aplicadas não só ao modelo de projeto, mas também ao modelo de requisitos.

Agora, aprenderemos um pouco sobre as métricas orientadas à classe, conhecidas como o conjunto de métricas cK. Nestas, são levadas em consideração a hierarquia de classes e as colaborações entre as mesmas. Uma classe encapsula dados e a função manipula dados, cada uma destas características é utilizada como a métrica cK, ou média *Chidamber e Kemerer* (PRESSMAN, 2011).

Na verdade, esse é um conjunto de três métricas conhecidas como métricas de software orientado a objetos, cujas características explanaremos a partir de agora. Começando com **métodos ponderados por classe**; aqui, o número de métodos e sua complexidade são indicadores razoáveis do trabalho para implementar e testar uma classe. Para este caso, quando mais o número de métodos cresce, mais ela tende a ser específica de uma aplicação, o que limita a sua reutilização, e é por isso que os métodos ponderados por classe devem ser mantidos os mais baixos possíveis, senão, deve ser realizada uma abordagem mais consistente.

A métrica **extensão da árvore de herança (DIT)** analisa o comprimento máximo desde o nó até a raiz da árvore. A medida que a métrica DIT cresce, é possível que classes herdem muitos métodos, causando dificuldades. Mas, por outro lado, grandes volumes de DIT implicam muitos métodos que podem ser reutilizados.

Para a métrica **resposta para uma classe (RFC)**, o conjunto de resposta de classe indica um conjunto de métodos que podem ser executados em resposta a determinada mensagem recebida por um objeto. No caso, o RFC, é o número de métodos no conjunto de respostas; se o RFC aumenta, também aumenta o trabalho em relação aos testes assim como a complexidade do projeto.

Bem, vimos, assim, algumas métricas do conjunto cK de métricas, que são importantes, pois trabalham com a análise de projetos orientados a objetos, analisando métodos, classes e funções que podem trazer problemas a um software. Na próxima aula, estudaremos a revisão de software.

REVISÃO DE SOFTWARE

Podemos dizer que revisar um software é uma das ações mais importantes a se realizar já que estamos analisando criticamente algo. Desta forma, o processo de revisão pode ser estabelecido como uma avaliação crítica de um software, de um processo, de um objeto ou classe (ANDRADE, 2015).

Dentro das revisões de software que podemos realizar, encontramos algumas que são mais utilizadas, como: inspeções, *walkthroughs* (passo a passo), *pair programming* (programação em pares), que são formas de processo de revisão.

Podemos destacar que as revisões visam à qualidade de um processo ou de um produto e são consideradas filtros de problemas e inconsistências que podem estar presentes em todo o processo de programação e desenvolvimento do software. Qualquer produto pode ser submetido a revisões, desde o princípio do ciclo de vida, estabelecendo a forma de avaliação da qualidade, com aspectos técnicos que dependem de cada produto.

Dentre as técnicas de revisão, começaremos a discutir a técnica de inspeção de software, que é um tipo de revisão aplicada a todos os artefatos presentes e que analisa praticamente tudo, sendo considerado rigoroso na sua inspeção (ANDRADE, 2015).

Com dito, o processo de revisão pode ser realizado em todo artefato presente. Para entendermos melhor, analise a imagem a seguir, em que cada inspeção é realizada entre cada artefato, verificando cada possível problema existente.

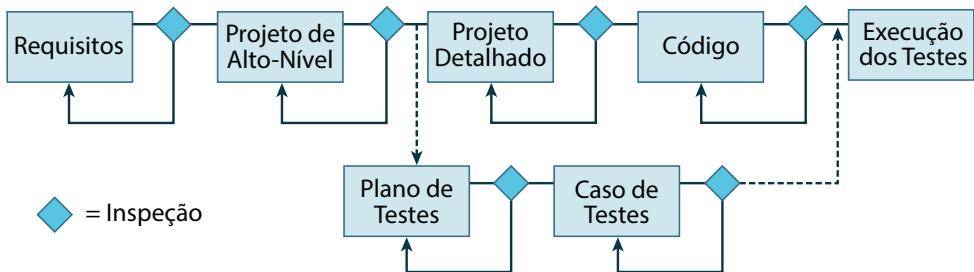


Figura 4 - Revisão por inspeções / Fonte: a autora.

Cada inspeção deve analisar erros, defeitos e falhas possíveis de acontecer. Além, é claro, da possível omissão de algum item importante, como um item relacionado à funcionalidade, ao desempenho do sistema, aos atributos, entre outros. Mas a inspeção, ainda, vai muito além da análise da ambiguidade de requisitos ou qualquer outra característica do sistema; a inconsistência de possíveis requisitos que geram conflitos; alguma informação estranha que foi inserida no código e que não é necessária, ou mesmo a inserção de requisitos em locais incorretos do código ou documento.

Realizar a inspeção de um software gerará diversos benefícios, como a diminuição do esforço, que é um dos principais problemas encontrados no desenvolvimento de um software. Evitar o retrabalho é importante, por isso, detectar os erros ou problemas em fases iniciais de construção do software facilitará toda a correção dos erros encontrados, diminuindo o esforço da equipe (ANDRADE, 2015). Gera-se, assim, outro benefício, que é a produtividade. Equipes podem se concentrar em realizar outras ações em projetos de software que possuem poucos erros, já que foram inspecionados, melhorando o tempo de entrega de cada produto.

Todas essas características resultam, ainda, na redução do custo de implementação. Observamos, assim, que o objetivo da inspeção sempre será melhorar a qualidade dos componentes do software, removendo o que há de ruim, pensando sempre nas próximas fases de implementação, resultando em ganhos ao final de cada projeto.

Outro tipo de inspeção realizada é o tipo conhecido como *walkthrough*, ou travessia. Esse tipo é como fosse uma reunião entre a equipe com a finalidade de analisar a qualidade do produto em relação ao desenvolvimento do software. Podemos, analogamente, comparar com o ensaio de câmeras na televisão, por exemplo. Isto trará a prevenção de possíveis erros e cada um saberá o que deve ser feito.

Por fim, temos a programação em pares, ou ainda, *pair programming*, este conceito não é muito difícil de entender e é muito utilizado. Aqui, o código fonte é compartilhado entre dois programadores e ambos trabalham programando e analisando o código criado, ao mesmo tempo e no mesmo equipamento.

Como característica principal dessa técnica, os programadores se intercalam nas ações, ou seja, ora um programador codifica, ora outro analisa o trabalho e, depois, realizam a troca de papéis, analisando as sintaxes, os métodos e tudo que pode ser otimizado e simplificado. Com isso, a técnica possibilita a troca de informações e conhecimento entre os pares de programadores, o que melhora a implementação. Temos diversas vantagens em relação à aplicação dessa técnica, como a aplicação de código coletivo e de padrões de projeto e, claro, o rodízio de pessoal na execução das atividades do projeto de implementação.

Em relação ao rodízio de pessoas, por exemplo, todos possuirão a mesma visão em relação ao que está sendo implementado no programa, já que os programadores trocam de lugar e favorecem a troca de conhecimento e a prática de propriedade coletiva do código, em que todos possuem acesso e conhecimento sobre ele.

Fora que, com a programação em par, há um ganho quanto à qualidade do produto que está sendo implementado, devido à atenção depositada na codificação, a aplicação de padrões bem definidos bem como a diminuição da quantidade de erros unitários ou de qualquer outro e, claro, devido à programação em par, a refatoração do código diminui, já que a análise do código é praticamente realizada ao mesmo tempo em que é codificada.

Claro que, dentro da realização de revisão de softwares, planeja-se o que se deve fazer, o que deve ser revisado, quais os resultados esperados. E isso deve ser feito pelo engenheiro de software, que também define quem deve realizar as revisões em todo o software, com todas as pré-condições a serem analisadas. Mas como isso deve ser organizado? Bom, pode utilizar *check-lists* para a execução das análises, para que nada fique fora do controle e da análise; além disso, é importante gerar documentos que informem o que foi alterado. A revisão é importante, já percebemos, e, quando realizadas desde o início de codificação, temos um ganho principal em relação ao custo total do projeto.

CONSIDERAÇÕES FINAIS

Caro(a) aluno(a), chegamos ao final da nossa unidade, em que estudamos os conceitos básicos sobre as métricas de software que ajudam a promover a melhoria contínua dos produtos de software. Uma métrica é algo que poderá ser medido no programa desenvolvido, desde o tamanho de um sistema em linhas de códigos até o número de defeitos que foram encontrados durante os testes realizados.

Na segunda aula, foram apresentados *frameworks* para a métrica de produto. O *framework* apresentado foi o HEART, que utiliza a métrica de experiência com o usuário e que auxilia na escolha de métricas cuja finalidade é medir a experiência do usuário com o software.

Na terceira aula, estudamos as métricas utilizadas para o modelo de requisitos de software. São métricas que utilizam os requisitos estabelecidos como base para o projeto e que proporcionam informações sobre a qualidade do modelo de análise.

Na quarta aula, falamos sobre as métricas para o modelo de software que proporcionam melhor visualização e podem ajudar a evolução do projeto para mais qualidade.

Por fim, falamos sobre a revisão de software e sua importância. Técnicas e métricas são essenciais para a garantia e a obtenção de qualidade, desta forma, sempre procure entender o que cada uma necessita e o que cada uma faz.

Depois destes conhecimentos adquiridos ao longo da unidade sobre as métricas, você está preparado(a) para seguir adiante. Preparado(a)? Então, continue sua leitura e bons estudos!



1. Em se tratando de projetos de software, devemos destacar a importância de métricas em sua construção. Uma métrica trará resultados como aperfeiçoamento de processos, produtividade, conformidade com o escopo do projeto e, claro, melhorar a qualidade do produto final.

Considerando o exposto sobre as métricas, podemos afirmar que a definição correta de métrica é:

- a) Algo que pode ser medido no software.
 - b) Indicação de um número máximo de linhas.
 - c) Padrão de elaboração de um projeto.
 - d) Realização da exclusão de linhas repetidas.
 - e) Análise do comportamento final do software.
2. Trabalhar com métricas em projetos de software se torna importante à medida que se necessita de mais qualidade no sistema. Existem métricas que atingem níveis fáceis e difíceis, no caso do gerente de projeto, ele deve conhecer todo o projeto e realizar uma parametrização do ambiente, para que, assim, saiba quais métricas possam ser executadas para atingir um bom nível de qualidade.

Considerando o trecho apresentado, sobre métricas de software, analise as afirmativas a seguir:

- I - Métricas de produto proporcionam informações sobre a qualidade do modelo que será analisado.
- II - Pontos de função são derivados das funções presentes no sistema, que são calculadas por valores de informações inseridas no programa.
- III - O cálculo de ponto por função é composto, somente, pelo número de saídas externas e pelo número de arquivos de interface externos.
- IV - Métrica baseada em função é utilizada para medir a funcionalidade que o sistema fornecerá.



É correto o que se afirma em:

- a) Apenas I e II.
 - b) Apenas I e IV.
 - c) Apenas I.
 - d) Apenas II, III e IV.
 - e) Nenhuma das alternativas está correta.
3. Todo projeto de software, uma hora ou outra, passará por algum tipo de métrica, com a finalidade de obter qualidade final. Para a construção de um projeto, é importante ter medidas, padrões ou métricas que satisfaçam a qualidade. Dentre as métricas aplicadas, podemos destacar a conhecida como cK, que é um conjunto delas.

Considerando o exposto sobre métricas cK, analise as afirmativas a seguir e assinale (V) para Verdadeiro e (F) para Falso.

- () Nos métodos ponderados por classe, o número de métodos e sua complexidade são indicadores razoáveis do trabalho para implementar e testar uma classe.
- () No método extensão da árvore de herança, o conjunto de resposta de classe quer dizer: um conjunto de métodos que podem ser executados em resposta a determinada mensagem recebida por um objeto.
- () No método resposta para uma classe, analisa-se o comprimento máximo de métodos e classes, desde o nó até a raiz da árvore. À medida que a métrica cresce, é possível que classes herdem muitos métodos, causando dificuldades.

A sequência correta para a resposta da questão é:

- a) V, V, F.
- b) F, F, V.
- c) V, F, F.
- d) F, F, F.
- e) V, V, V.



4. Em se tratando de software, podemos destacar a importância de realizar a revisão de Software. O processo de revisão é uma análise crítica do software, de um processo, de uma classe ou um objeto. Existem alguns tipos de revisões de software e podemos destacar as técnicas de inspeção e de pair programming. Considerando a técnica de pair programming, explique como é o funcionamento dela.
5. “As métricas de produto são métricas de previsão usadas para medir atributos internos de um sistema de software. O tamanho de sistema, medido em linhas de código, ou o número de métodos associados a cada classe de objeto são exemplos de métricas de produto” (SOMMERVILLE, 2011, p. 454).

Considerando o texto apresentado, descreva a técnica de inspeção de software.



MEDIÇÕES E MÉTRICAS DE SOFTWARE

A medição de software preocupa-se com a derivação de um valor numérico ou o perfil para um atributo de um componente de software, sistema ou processo. Comparando esses valores entre si e com os padrões que se aplicam a toda a organização, você pode ser capaz de tirar conclusões sobre a qualidade do software ou avaliar a eficácia dos métodos, das ferramentas e dos processos de software.

Por exemplo, digamos que uma organização tem a intenção de introduzir uma nova ferramenta de teste de software. Antes de introduzir a ferramenta, em um determinado momento você registra o número de defeitos de software descobertos. Essa é uma baseline de avaliação da eficácia da ferramenta. Depois de algum tempo usando a ferramenta, esse processo é repetido. Se mais defeitos forem encontrados durante o mesmo período, depois que a ferramenta foi introduzida, você pode decidir se ela fornece suporte útil para o processo de validação de software.

O objetivo a longo prazo de medição de software é usá-la no lugar de revisões para fazer julgamentos sobre a qualidade de software. Usando a medição de software, um sistema poderia, idealmente, ser avaliado usando uma variedade de métricas e, a partir dessa medição, deduzir um valor para a qualidade do sistema. Se o software atingir o limiar de qualidade requerido, então ele poderia ser aprovado sem revisão. Quando apropriado, as ferramentas de medição também podem realçar áreas do software que poderiam ser melhoradas. No entanto, estamos ainda longe dessa situação ideal e não há sinal de que as avaliações automatizadas de qualidade venham a se tornar realidade em um futuro próximo.

Uma métrica de software é uma característica de um sistema de software, documentação de sistema ou processo de desenvolvimento que pode ser objetivamente medido. Exemplos de métricas incluem: o tamanho de um produto em linhas de código; o índice Fog que é uma medida da legibilidade de uma passagem de texto escrito; o número de defeitos relatados em um produto de software entregue, e o número de pessoas/dia requerido para desenvolver um componente de sistema.

As métricas de software podem ser métricas de controle ou métricas de previsão. Como os nomes sugerem, as primeiras suportam os processos de gerenciamento e



as outras o ajudam a prever as características do software. As métricas de controle são geralmente associadas com os processos de software. Exemplos de métricas de controle ou de processos são o esforço médio e o tempo necessário para reparar os defeitos relatados. Métricas de previsão são associadas com o software em si e, por vezes, são conhecidas como ‘métricas de produto’. São exemplos de métricas de previsão: a complexidade ciclomática de um módulo, o comprimento médio dos identificadores em um programa e o número de atributos e operações associadas com as classes de objeto em um projeto.

As métricas de controle e de previsão podem influenciar a tomada de decisão de gerenciamento. Os gerentes usam métricas de processo para decidir se devem ser feitas alterações no processo; as métricas de previsão são usadas para ajudar a estimar o esforço necessário para fazer as alterações no software.

Existem duas maneiras para o uso das medições de um software de sistema:

1. Para atribuir um valor aos atributos de qualidade de sistema. Ao medir as características dos componentes de sistema, bem como sua complexidade ciclomática e, em seguida, agregar essas medições, você pode avaliar os atributos de qualidade do sistema, como a manutenibilidade.

2. Para identificar os componentes de sistema cuja qualidade não atingiu o padrão. As medições podem identificar componentes individuais com características que se desviem da norma. Por exemplo, você pode medir componentes para descobrir aqueles com a mais alta complexidade. Esses são mais propensos a conter bugs porque a complexidade os torna mais difíceis de entender.



Infelizmente, é difícil fazer medições diretas de muitos dos atributos de qualidade de software. Os atributos de qualidade como manutenibilidade, comprehensibilidade e usabilidade são atributos externos relacionados com os desenvolvedores e usuários que experimentam o software. Eles são afetados por fatores subjetivos, como a experiência e a educação do usuário e, portanto, não podem ser medidos objetivamente. Para fazer um julgamento sobre esses atributos, você deve medir alguns atributos internos do software (como tamanho, complexidade etc.) e assumir que estão relacionados com as características de qualidade com as quais você se preocupa.

Os atributos internos de software, como a complexidade ciclomática de um componente, são medidos com o uso de ferramentas de software que analisam o código-fonte do software. As ferramentas open source disponíveis podem ser usadas para fazer essas medições. Embora a intuição sugira que poderia haver um relacionamento entre a complexidade de um componente de software e o número de falhas observadas em uso, é difícil demonstrar objetivamente que é este o caso. Para testar essa hipótese, você precisa de dados de falha de um grande número de componentes e de acesso ao código-fonte do componente para análise. Poucas empresas fizeram um compromisso a longo prazo para a coleta de dados sobre seu software, portanto, dados de falha raramente são disponibilizados para análise.

Fonte: Sommerville (2011, p. 480-481).



eu recomendo!



livro

Engenharia de Software: Uma Abordagem Profissional

Autor: Roger S. Pressman e Bruce R. Maxim

Editora: McGraw-Hill Education

Sinopse: o livro trata de assuntos relacionados à engenharia de software, desde o princípio da história e as formas de construção, com seus modelos, até chegar à melhoria a partir da qualidade e revisão de software. É uma ótima edição, que trata da engenharia do século XXI e, também, de assuntos relacionados à segurança de software e de informações sobre a construção de aplicativos móveis.



filme

O Jogo da Imitação

Ano: 2014

Sinopse: o filme acontece durante a Segunda Guerra Mundial, quando o governo britânico pretendia montar uma equipe a fim de quebrar um enigma, o código que os alemães utilizavam para enviar mensagens aos seus submarinos, e o grande Alan Turing faz parte deste grupo que pretende quebrar esse código. Na época, Turing é um matemático lógico com grande foco no trabalho, a princípio, ele não era o líder da equipe, mas se tornou a principal peça para a resolução do problema. O objetivo era construir uma máquina que permitisse a análise das informações e de todas as possibilidades que poderiam ser codificadas. O código foi quebrado em 18 horas e, assim, os ingleses começaram a conhecer as mensagens enviadas.





eu recomendo!



conecte-se

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

Este artigo visa efetuar uma abordagem da mensuração de softwares por meio 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.

https://www.researchgate.net/publication/272906572_Uma_analise_de_métricas_de_software_orientadas_a_funcao_e_sua_aplicacao_no_desenvolvimento_orientado_a_objetos



GARANTIA DA QUALIDADE DE SOFTWARE

PROFESSORA

Esp. Janaina Aparecida de Freitas

PLANO DE ESTUDO ▼

A seguir, apresentam-se as aulas que você estudará nesta unidade:

- Conceitos de garantia da qualidade de software
- Elementos de garantia da qualidade de software
- Abordagens formais de SQA
- Estatísticas da garantia da qualidade de software
- Confiabilidade de software.

OBJETIVOS DE APRENDIZAGEM ▼

Entender os conceitos usados na garantia de qualidade de software

- Entender os elementos trabalhados na garantia da qualidade de software
- Aprender as abordagens formais da SQA
- Compreender as estatísticas usadas na garantia da qualidade de software
- Compreender a confiabilidade de software.

INTRODUÇÃO



Olá, alunos! Nesta unidade, aprenderemos, um pouco mais, como garantir a melhor qualidade ao software que está sendo criado. Começaremos apresentando os principais conceitos usados na garantia da qualidade de software.

Dando sequência, serão apresentados elementos trabalhados na garantia dessa qualidade. Dentre esses elementos, estão os padrões que são essenciais na construção do software e, normalmente, feitos por organizações de padronização, como a ISO (Organização Internacional de Padronização) e o IEEE (Instituto de Engenheiros Eletrotécnicos e Eletrônicos).

Outro ponto abordado, nesta unidade, será a apresentação das abordagens formais da SQA, em que são definidas uma sintaxe e uma semântica mais rigorosas para todas as linguagens de programação existentes e as criadas futuramente, além da realização de uma rigorosa abordagem da especificação dos requisitos de software.

Por fim, mostraremos um pouco das estatísticas da garantia de qualidade de software e como obter a confiabilidade de software. A confiabilidade, diferentemente de outros fatores de qualidade, pode ser medida diretamente e estimada usando dados históricos e de desenvolvimento.

Estes tópicos são importantes e essenciais para a aplicação adequada de técnicas que trazem, ao final do projeto de software, qualidade e confiabilidade por parte do cliente.

Preparado(a) para continuar? Então, vamos seguir em frente! Boa leitura e bons estudos!

1

CONCEITOS DE GARANTIA DA QUALIDADE DE software

Garantir a qualidade de um software requer que, antes, seja entendido o que significa essa qualidade. Desta forma, Guerra e Colombo (2009, p. 17) citam que:



A Norma NBR ISO 9000 sobre terminologias de gestão e garantia da qualidade define qualidade como: A totalidade das características de uma entidade que lhe confere a capacidade de satisfazer às necessidades explícitas e implícitas.

Na manufatura, a qualidade pode não atingir os 100%, o que não podemos aplicar à qualidade de software. Tolerâncias não podem ser aplicadas aqui, já que, dependendo do propósito do software, o mesmo pode ser essencial para a segurança ou para a vida de pessoas. Mesmo assim, em relação à construção, é difícil escrever especificações de software que sejam tão completas e precisas; e vai além, cada cliente ou desenvolvedor pode realizar interpretações dos requisitos de forma diferente, com isso, é complicado dizer se um programa cumpre o que foi proposto de forma completa e precisa (SOMMERVILLE, 2011).



No desenvolvimento de software, a qualidade de um projeto engloba o grau de atendimento às funções e características especificadas no modelo de requisitos.

Isso acontece porque, geralmente, os requisitos são especificados por diversos *stakeholders* e cada um deles pode verificar o sistema de uma forma diferente, podendo perceber que um sistema pode ter baixa qualidade, enquanto o outro percebe que a qualidade está boa. Além disso, há características que são impossíveis de serem medidas, por exemplo, a manutenibilidade de um sistema. Sendo assim, a avaliação da qualidade se tornou um processo subjetivo, em que a equipe que realizar o gerenciamento da qualidade é quem julga se determinado objetivo foi alcançado.

Para este julgamento, algumas perguntas são lançadas e respondidas, por exemplo, se o software foi devidamente testado, se é útil, se está bem estruturado e compreensível, entre outras. Decidir se o software oferece ou não a funcionalidade desejada parte da análise dos testes e dos registros de testes realizados, para verificar se foram feitos de forma correta ou não. Quando falamos em qualidade de software, devemos, também, verificar alguns atributos que fazem essa qualidade de ser atingida, os quais são atributos de confiança e importantes para o sistema, como podemos verificar no Quadro 1.

Segurança	Compreensibilidade	Portabilidade
Proteção	Testabilidade	Usabilidade
Confiabilidade	Adaptabilidade	Reusabilidade
Resiliência	Modularidade	Eficiência
Robustez	Complexidade	Capacidade de armazenamento

Quadro 1 - Atributos de qualidade de software / Fonte: Sommerville (2011).

Para a qualidade, devemos, então, definir um plano de qualidade e definir quais atributos são mais importantes para o software que está sendo construído. Assim, você pode definir, no plano, que um atributo é crítico e, assim, os desenvolvedores trabalham da melhor forma em cima deste atributo.

A qualidade de software está relacionada à qualidade do processo de desenvolvimento, é um processo de fabricação que envolve configurar e operar máquinas envolvidas no processo de codificação. Desta forma, não temos dúvida de que o processo de desenvolvimento influencia a qualidade de software e que bons processos são mais fáceis de serem analisados.

Garantir a qualidade de um software e demandar o seguimento de padrões de ações que são exigidas para garantir essa qualidade. A este processo é dado o nome de *Software Quality Assurance* – SGA (Garantia de Qualidade de Software) que possui uma série planejada de atividades de apoio que auxiliam na confiança ao programa.

Como padrões de ações, podemos citar: (i) aplicação de métodos técnicos; (ii) realização de revisões; (iii) atividade de testes de software; (iv) aplicação de padrões e procedimentos; (v) processo de controle de mudanças; (vi) mecanismos de medição; (vii) anotação e manutenção de registros (ANDRADE, 2015).

Alguns aspectos positivos da garantia da qualidade podem ser mencionados, como: o software terá menos defeitos, terá mais confiabilidade e satisfação do cliente, os custos de manutenção podem ser reduzidos. Por fim, conseguimos entender a importância da qualidade no produto final e como isso gera maior índice de relacionamento com os clientes.



pensando juntos

Embora seja tentador criar medidas quantitativas para os fatores de qualidade aqui citados, também podemos criar uma lista de verificação simples dos atributos que dão a sólida indicação de que o fator está presente.





2 ELEMENTOS DE GARANTIA da qualidade de software



Ter qualidade é essencial, já entendemos isso; mas garantir que a qualidade de software se mantenha durante seu ciclo de vida requer que algumas atividades sejam aplicadas.

Dentre as atividades, podemos começar destacando a aplicação de padrões. Eles são 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).

O papel dos padrões aplicados é garantir que sejam seguidas as etapas para a construção do software e, claro, a SQA (*Software Quality Assurance*), ou seja, a garantia de qualidade de software deve certificar que os padrões aplicados sejam seguidos e que os produtos estejam em conformidade com ele (PRESSMAN, 2011). Outro item importante são as revisões e as auditorias. As revisões são atividades cuja finalidade é verificar o que foi feito a fim de buscar erros, é uma atividade de controle da qualidade. Já as auditorias têm a finalidade de assegurar se tudo relacionado à qualidade está sendo aplicado e seguido de forma correta.

Na atividade teste, que também faz parte da obtenção da qualidade, tem-se, como finalidade, descobrir erros que possam acontecer. O papel da SQA é garantir que todos os testes sejam planejados antes de serem aplicados, para que, assim, sejam executados de forma eficiente e consigam atingir os seus objetivos.

Devemos entender que não se trata apenas de encontrar os erros, mas analisar erros e defeitos e como eles acontecem, para que os engenheiros de software possam adequar o sistema, esta atividade é chamada de coleta e análise de erros/defeitos. Além de tudo isso, realizar o gerenciamento de mudanças é essencial, nesta atividade, a SQA garante que práticas adequadas sejam aplicadas, pois mudanças podem ser negativas no projeto. São várias atividades e uma das mais importantes é a administração da segurança. Sabemos que, conforme a tecnologia cresce, mais crimes digitais acontecem, desta forma, toda empresa precisa ter uma política de segurança que proteja os seus dados. A SQA garante que sejam aplicadas as tecnologias necessárias para a segurança adequada (PRESSMAN, 2011).

Continuando com a segurança, temos uma atividade conhecida como proteção. Sabemos que é possível trabalhar com muitos tipos de software, inclusive os que envolvem vidas humanas, assim, o impacto dos erros pode ser terrível. Por isso, a SQA, aqui, é responsável por avaliar os problemas e aplicar etapas que os reduzem.

Temos ainda, a atividade de administração de riscos, em que a SQA garante que as atividades de gestão sejam conduzidas adequadamente e que planos de contingência tenham sido estabelecidos.

Quando falamos em garantia de qualidade de software, devemos entender que ela, além das atividades citadas anteriormente, é composta por tarefas, metas e métricas. As tarefas tratam do planejamento, da supervisão, da manutenção de registros, análise e relatórios. Isso, claro, é realizado por meio de um plano de SQA para um projeto, que deve ser preparado de forma correta, já que identificará diversas características, como revisões a serem realizadas. Podemos citar, também, como tarefa, revisar as atividades de engenharia de software para verificar a sua conformidade com a gestão da qualidade; realizar a auditoria de produtos de software, garantindo que desvios de trabalho de software e produtos sejam documentados e tratados de acordo com um procedimento documentado; e também, o acompanhamento de erros até que eles sejam resolvidos (PRESSMAN, 2011).

Dentre as metas, os atributos e as métricas, podemos citar a qualidade dos requisitos. Estes devem ser analisados, criteriosamente, para assegurar que a equipe de desenvolvimento consiga aplicá-los, e que, no final, obtenha-se alto nível de qualidade.

Temos, ainda, a qualidade do projeto e a qualidade do código. Na qualidade do projeto, a equipe de software deve garantir que os requisitos e o próprio projeto estejam de acordo com o que foi solicitado. Na qualidade do código, estes devem estar em conformidade com os padrões de codificação, o que facilitará a manutenção futura.

Por fim, podemos citar a eficácia do controle de qualidade, cuja equipe deve aplicar recursos para obter mais qualidade final, analisando recursos, aplicando testes e averiguando se tudo está sendo alocado de maneira efetiva. Assim, a qualidade do software aumentará, o cliente ficará satisfeito e a empresa terá diversos benefícios.



A qualidade de software é uma tarefa importante e atingida por meio da prática e da competência, quando aplicada por parte dos profissionais. Além disso, ela é obtida por meio de revisões técnicas, estratégias de testes, melhor controle do artefato de software e, também, aplicação de padrões e análise de mudanças feitas. Ainda, a obtenção da qualidade pode ser definida pelos atributos realizados com qualidade e média, aplicando-se diversos itens, como índices e métricas importantes.

Com o passar do tempo, foi visto que eram necessárias abordagens mais formais para garantir a qualidade de software. Por isso, definiu-se uma sintaxe e uma semântica mais rigorosas para todas as linguagens de programação existentes e as criadas futuramente; além disso, foi realizada uma rigorosa abordagem da especificação dos requisitos de software.

Pressman (2011, p. 393), cita que:



Se o modelo de requisitos (especificações) e a linguagem de programação podem ser representados de maneira rigorosa, deve ser possível aplicar uma prova matemática da correção para demonstrar a adequação exata de um programa às suas especificações.

Provar que programas são corretos, por sua vez, são tentativas novas e estão associadas ao uso de conceitos de programação estruturada, por isso, foi criado um processo global de melhoria contínua de qualidade, o qual é conhecido como TQM (*Total Quality Management*). O TQM é resultado de diversos níveis entre pessoas envolvidas que, aqui, são formados por professores, estudantes e funcionários, com extensão em projetos à comunidade, tudo voltado a criar abordagens para a melhoria de SQA.

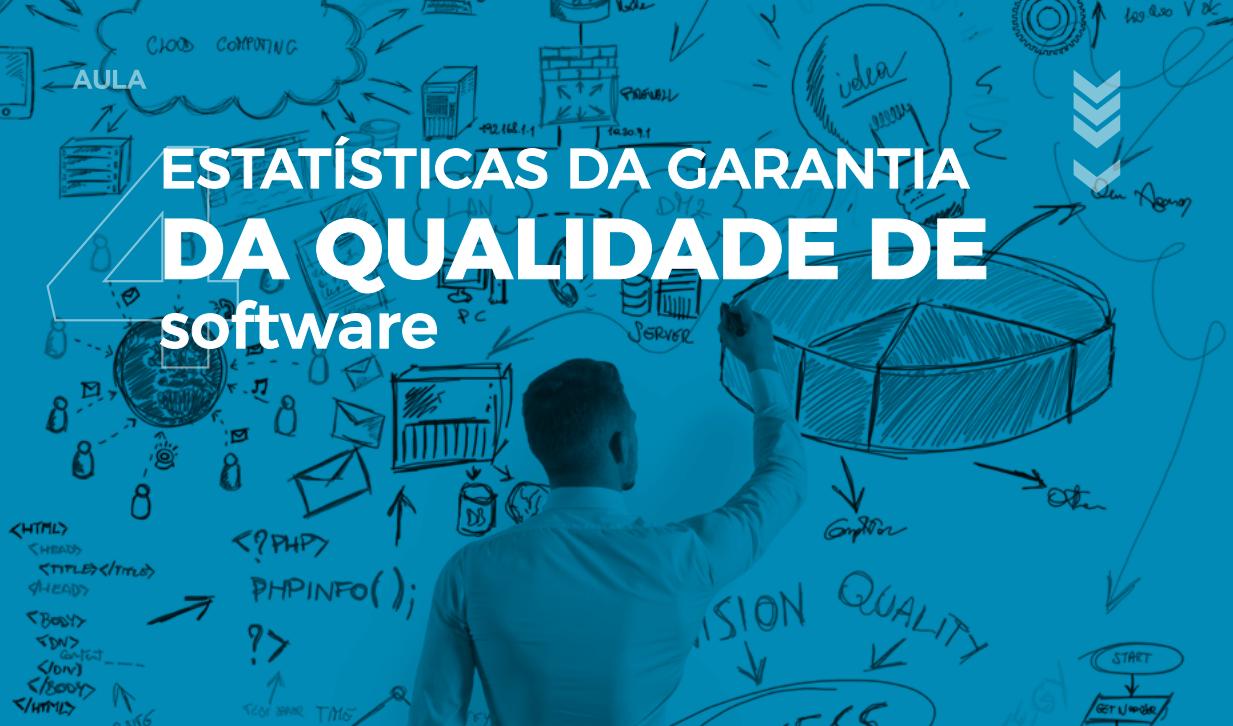


explorando Ideias

Existem 14 pontos de efetiva aplicação do TQM:

1. O objetivo é formar alunos da melhor qualidade que assumirão posições significativas na sociedade.
2. Ter gerentes que se tornem líderes para a mudança.
3. Abolir a classificação e os efeitos prejudiciais da classificação dos alunos.
4. Fornecer experiências de aprendizagem para criar um desempenho de qualidade.
5. Minimize o custo total da educação, trabalhando para melhorar o relacionamento com as fontes de alunos e a qualidade dos alunos que entram em seu sistema.
6. Consistentemente, nos esforçamos para melhorar o serviço prestado aos alunos.
7. Instituir treinamento prático para todos: professores, administradores e alunos.
8. Institua liderança ao invés de “gerenciamento do chefe”.
9. Crie um ambiente livre de coerção e medo.
10. Incentive o ensino em equipe e, ao fazê-lo, elimine as barreiras entre professores ou departamentos.
11. Elimine slogans competitivos, exortações e programas de retirada, uma vez que eles geram relacionamentos adversários.
12. Elimine os padrões de trabalho (cotas) bem como a aderência ao modelo de “curva normal”.
13. Mude o foco na educação de quantidade para qualidade e, ao fazê-lo, remova as barreiras que privam alunos, professores e administradores escolares de seu orgulho pela obra.
14. Envolva todos na transformação da escola em um ambiente de qualidade.

Fonte: a autora.



ESTATÍSTICAS DA GARANTIA DA QUALIDADE DE Software

A estatística da garantia da qualidade de software está em ascensão com o objetivo de tornar mais quantitativa a análise dessa qualidade. A estatística implica algumas etapas que devem ser seguidas, como: devem ser coletadas e classificadas informações sobre erros; deve-se associar cada erro ou defeito a uma causa; deve ser utilizado o princípio de Pareto, em que 80% dos defeitos podem ser associados a 20% de todas as possíveis causas e, assim que as causas forem identificadas, prossegue-se com a correção dos problemas que provocam os erros.

Este conceito é considerado importante para a criação de uma gestão da qualidade em que as mudanças são realizadas para melhorar elementos do processo que introduzem os erros no sistema. Para entendermos um pouco melhor, Pressman (2011, p. 393) cita um exemplo genérico:



Para ilustrar o uso de métodos estatísticos para a engenharia de software, suponhamos que uma organização de engenharia de software reúna informações sobre erros e defeitos por um período de um ano. Alguns desses erros são revelados à medida que o software é desenvolvido.

São encontrados outros (defeitos) após o software ter sido liberado para os usuários finais. Embora centenas de problemas diferentes sejam encontrados, todos podem ser associados a uma (ou mais) das seguintes causas:

- Especificações incompletas ou errôneas
- Má interpretação da comunicação do cliente
- Desvio intencional das especificações
- Violação dos padrões de programação
- Erro na representação de dados
- Interface inconsistente de componentes
- Erro na lógica de projeto
- Testes incompletos ou errôneos
- Documentação imprecisa ou incompleta
- Erro na tradução do projeto para linguagem de programação
- Interface homem-máquina ambígua ou inconsistente

Dentre as causas citadas, caso haja problemas em relação à má interpretação da comunicação do cliente, para realizar a correção deste problema, é preciso implementar técnicas de reunião de requisitos para melhorar a qualidade da comunicação do cliente e das especificações. É importante notarmos que a realização da ação corretiva se concentra em causas que sejam importantes para a vida do software, as técnicas de estatística da garantia da qualidade para software demonstram fornecer um aperfeiçoamento substancial da qualidade.

Há uma técnica conhecida como **seis sigma** para a engenharia de software e, conforme Pressman (2011, p. 393), “Seis Sigma é a estratégia para a estatística da garantia da qualidade mais utilizada na indústria atual”. Essa estratégia é uma metodologia rigorosa e disciplinada que utiliza a análise estatística e de dados para a média e para melhorar o desempenho operacional de uma empresa por meio da identificação e da eliminação de defeitos em processos de fabricação de software.

Segundo Pressman (2011, p.394), a metodologia seis sigma define três etapas essenciais:



- Definir as necessidades do cliente e os artefatos passíveis de entrega, bem como as metas de projeto, através de métodos bem definidos da comunicação com o cliente.
- Medir o processo existente e seu resultado para determinar o desempenho da qualidade atual (reunir métricas para defeitos).
- Analisar as métricas para defeitos e determinar as poucas causas vitais.

Caso haja uma gestão de qualidade, o modelo sugere mais duas etapas adicionais, que são melhorar o processo por meio da eliminação das causas fundamentais dos defeitos e controlar o processo para garantir que trabalhos futuros não reintroduzam as causas dos defeitos.

Caso uma organização desenvolver uma gestão de qualidade e não aperfeiçoar uma existente, são adicionadas algumas etapas, como: projetar o processo para evitar as causas fundamentais dos defeitos e atender às necessidades do cliente; verificar se o modelo de processos evitará defeitos e atenderá às necessidades desse cliente.



explorando Ideias

Os primeiros trabalhos sobre confiabilidade de software tentaram extrapolar a matemática da teoria da confiabilidade de hardware para a previsão da confiabilidade de software. A maioria dos modelos de confiabilidade relacionada com hardware tem, como base, falhas devido a desgaste e não falhas devido a defeitos de projeto. Em hardware, as falhas devido a desgaste físico (por exemplo, os efeitos decorrentes de temperatura, corrosão, choque) são mais prováveis do que uma falha relacionada ao projeto. Infelizmente, o contrário é verdadeiro para software. Na realidade, todas as falhas de software podem ser associadas a problemas de projeto ou de implementação; o desgaste não entra em questão.

Fonte: Pressman e Maxim (2016, p. 338).



CONFIABILIDADE DE SOFTWARE

5



Quando aplicadas as técnicas para a qualidade do software, claramente, temos mais confiabilidade. Esta é um elemento fundamental e pode ser medida diretamente e estimada usando dados históricos e de desenvolvimento; e mais, é definida de forma estatística, como a probabilidade de realizar operações de um programa de computador, sem falhas, em dado ambiente por determinado espaço de tempo (PRESSMAN, 2011).

Por exemplo, imagine que um programa tenha a confiabilidade de 0,999, depois de ser executado mais de oito horas seguidas de processamento, ou seja, se o programa tiver que ser executado 1000 vezes e precisar de um total de oito horas de processamento, é provável que ele opere, sem falhas, por 999 vezes, o que é algo muito bom.

Em se tratando de confiabilidade, é importante entender o significado de falha, que nada mais é do que a falta de conformidade com os requisitos de software. Há falhas que podem ser problemáticas e que podem ser corrigidas em segundos, mas também há outras que podem levar muito mais tempo para a correção, chegando a meses.

Outro ponto que deve ser entendido é que a correção de determinadas falhas pode, também, resultar no surgimento de outros erros, os quais poderão resultar em outras falhas e, por isso, cada item deve ser analisado, corretamente, desde o início da construção do software (PRESSMAN, 2011). Para auxiliar nisso, existem medidas de

confiabilidade e disponibilidade que exploram a teoria da confiabilidade de hardware. A maioria dos modelos de confiabilidade estão relacionados com o hardware e têm, como base, falhas devido ao desgaste e não falhas devido aos defeitos do projeto.

Como citamos, em hardware, as falhas que acontecem devido ao desgaste físico são causadas por uma falha relacionada ao projeto, já todas as falhas de software podem ser associadas a problemas de projeto ou de implementação.

Continuando com a confiabilidade de software, temos um item que está relacionado e retorna mais confiabilidade: a chamada proteção do software. Esta é uma atividade relacionada à garantia e se concentra na identificação e na avaliação de potenciais problemas que podem afetar, negativamente, um software e provocar falhas em um sistema.

Sabemos que a maioria das atividades podem ser especificadas no projeto de software e, com isso, os problemas podem ser identificados e controlados para a sua eliminação. Uma vez identificados problemas no nível de sistemas, técnicas de análise são utilizadas para atribuir valores de gravidade e a sua probabilidade de ocorrência. Para um software ser efetivo, ele deve ser analisado no contexto de todo o sistema (PRESSMAN, 2011).

Para prever uma cadeia de eventos com chances de causar problemas, podem ser realizados análises técnicas, como análise de árvore de falhas, lógica em tempo real e diversos outros. Uma vez que problemas são identificados e analisados, requisitos relacionados com a proteção podem ser especificados para o software, ou seja, podem conter uma lista de eventos indesejáveis e as respostas desejadas pelo sistema para esses eventos.



Proteção do software é uma atividade da garantia da qualidade de software que se concentra na identificação e na avaliação de potenciais problemas que podem afetar negativamente um software e provocar falha em todo o sistema. Se os problemas podem ser identificados precocemente na gestão de qualidade, as características para eliminar ou controlar estes problemas podem ser especificadas no projeto de software (PRESSMAN, 2011, p. 395).

Uma discussão a respeito da proteção de um software vai muito além do que imaginamos, por isso devem ser analisados e estudados casos e técnicas que auxiliam a, cada vez mais, haver proteção do sistema e dos dados, a fim de obter a qualidade final do software.



explorando Ideias

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.

Fonte: Pressman e Maxim (2016, p. 344).



CONSIDERAÇÕES FINAIS

Prezado(a) aluno(a)! Chegamos ao final da última unidade do nosso livro. Estudamos, nesta unidade, a garantia de qualidade. Aprendemos, um pouco mais, como obter mais qualidade em relação ao software.

Foram apresentados conceitos iniciais e elementos da garantia da qualidade de software. As duas primeiras aulas foram importantes para entendermos como devem ser realizadas atividades para garantir a qualidade.

Após isso, falamos sobre abordagens formais de SQA e estatísticas a respeito da garantia da qualidade de software. Por fim, falamos um pouco mais de como aplicar técnicas que resultam na confiabilidade de software.

Ao final desta unidade, você será capaz de entender a importância dessa qualidade e de como aplicar seus conhecimentos para que ela seja garantida. Esperamos que, a partir do conhecimento adquirido, você procure pesquisar e estudar além do conteúdo citado no livro, pois sempre surgem coisas novas para otimizar o trabalho e melhorar a qualidade.

Portanto, para um futuro profissional de engenharia de software, fica a dica: sempre estude, pesquise e procure testar novas técnicas, ferramentas e abordagens até encontrar a que funciona melhor para você, para sua equipe ou empresa que você trabalha. Inove sempre.

Desejo sucesso em toda a sua caminhada em busca de conhecimento!!



1. “Com a constante demanda gerada pela vida moderna, cada vez mais os computadores passam a integrar a rotina diária e a produção de software vem tendo um aumento constante. Exigência por qualidade estende-se também à área de software e pode ser considerada o centro das atenções para o desenvolvimento de software” (ANDRADE, 2015, p. 88).

Considerando o exposto sobre a qualidade de software, podemos afirmar que a definição correta de qualidade é:

- a) Normas que podem ser utilizadas no desenvolvimento de sistemas de gerenciamento.
 - b) Processo de fabricação que envolve configurar e operar máquinas em um processo.
 - c) A totalidade de características de um produto que satisfaz às necessidades declaradas.
 - d) Manuais organizacionais e planos individuais obtidos a partir de um modelo principal.
 - e) Certificados de processos definidos em conformidade com normas estabelecidas.
2. “Pode-se definir qualidade de produto de software como a conformidade a requisitos funcionais e de desempenho declarados explicitamente, padrões de desenvolvimento claramente documentados e as características implícitas que são esperadas de todo software desenvolvido profissionalmente. As definições de qualidade podem variar em alguns aspectos, porém o aspecto que se refere à satisfação do cliente ou usuário não deve ser esquecido” (GUERRA; COLOMBO, 2009, p. 135).

Considerando o trecho do texto apresentado sobre qualidade de software, analise as afirmativas a seguir:

- I - O papel dos padrões é garantir que sejam seguidas etapas para a construção do software e, claro, a garantia de qualidade de software.
- II - As revisões são atividades que têm a finalidade de verificar todo o código, buscando comentários incoerentes e código mal indentado.



III - O papel da SQA é garantir que todos os testes sejam planejados antes de serem aplicados para que, assim, consigam atingir os seus objetivos.

IV - A SQA garante que sejam aplicadas as tecnologias necessárias para ter segurança adequada.

É correto o que se afirma em:

- a) Apenas I e II.
 - b) Apenas I e IV.
 - c) Apenas I.
 - d) Apenas I, III e IV.
 - e) Nenhuma das alternativas anteriores está correta.
3. “Os fundamentos do gerenciamento de qualidade foram estabelecidos pela indústria manufatureira em um esforço para melhorar a qualidade dos produtos em produção. Como parte disso, eles desenvolveram uma definição de ‘qualidade’, baseada na conformidade com uma especificação detalhada e na noção de tolerâncias” (SOMMerville, 2011, p. 346).

Considerando o exposto e, a respeito das estatísticas da garantia de qualidade de software, analise as afirmativas a seguir e assinale (V) para Verdadeiro e (F) para Falso.

- () A estatística implica algumas etapas, como: coletar e classificar informações sobre erros; associar cada erro ou defeito a uma causa, utilizar o princípio de Pareto.
- () A metodologia seis sigma, em uma de suas etapas, define as necessidades do cliente e os artefatos passíveis de entrega bem como as metas de projeto, por meio de métodos bem definidos da comunicação com o cliente.
- () A técnica seis sigma é uma metodologia rigorosa e disciplinada que utiliza a análise estatística e de dados para medir e melhorar o desempenho operacional de uma empresa, por meio da identificação e eliminação de defeitos.

É correto o que se afirma em:



- a) V, V, F.
 - b) F, F, V.
 - c) V, F, F.
 - d) F, F, F.
 - e) V, V, V.
4. Alcançar a qualidade em um projeto de software requer que sejam aplicadas e entendidas técnicas específicas, as quais trabalham na busca de erros, falhas e diversos outros problemas. As aplicações dessas técnicas resultam em muita confiabilidade do software, e que podem ser medidas e estimadas. Considerando a confiabilidade de software alcançada, explique como a confiabilidade pode ser medida, expondo suas principais características.
5. “As técnicas de estatística da garantia da qualidade para software demonstraram fornecer um aperfeiçoamento substancial da qualidade. Em alguns casos, as organizações de software atingiram a redução de 50% por ano nos defeitos após a aplicação dessas técnicas” (PRESSMAN, 2011, p. 63).

Considerando o texto apresentado, descreva a técnica conhecida como seis sigma.



ELEMENTOS DE GARANTIA DA QUALIDADE DE SOFTWARE

A garantia da qualidade de software engloba um amplo espectro de preocupações e atividades que se concentram na gestão da qualidade de software e que podem ser sintetizadas da seguinte maneira:

Padrões. O IEEE, a ISO e outras organizações de padronizações produziram uma ampla gama de padrões para engenharia de software e os respectivos documentos. Os padrões podem ser adotados voluntariamente por uma organização de engenharia de software ou impostos pelo cliente ou outros interessados. O papel da SQA é garantir que padrões que tenham sido adotados sejam seguidos e que todos os produtos resultantes estejam em conformidade com eles.

Revisões e auditorias. As revisões técnicas são uma atividade de controle de qualidade realizada por engenheiros de software para engenheiros de software. Seu intuito é o de revelar erros. Auditorias são um tipo de revisão efetuado pelo pessoal de SQA com o intuito de assegurar-se de que as diretrizes de qualidade estejam sendo seguidas no trabalho de engenharia de software. Por exemplo, uma auditoria do processo de revisão pode ser realizada para garantir que as revisões estejam sendo realizadas de maneira que conduza à maior probabilidade de descoberta de erros.

Testes. Os testes de software são uma função de controle de qualidade com um objetivo principal — descobrir erros. O papel da SQA é garantir que os testes sejam planejados apropriadamente e conduzidos eficientemente de modo que se tenha a maior probabilidade possível de alcançar seu objetivo primário.

Coleta e análise de erros/defeitos. A única forma de melhorar é medir o nosso desempenho. A SQA reúne e analisa dados de erros e defeitos para melhor compreender como os erros são introduzidos e quais atividades de engenharia de software melhor se adequa para sua eliminação.

Gerenciamento de mudanças. As mudanças são um dos aspectos mais negativos de qualquer projeto de software. Se não forem administradas apropriadamente, podem gerar confusão, e confusão quase sempre leva a uma qualidade inadequada. A SQA garante que práticas adequadas de gerenciamento de mudanças tenham sido instituídas.



Educação. Toda organização de software quer melhorar suas práticas de engenharia de software. Um fator fundamental para o aperfeiçoamento é a educação dos engenheiros de software, seus gerentes e outros interessados. A organização de SQA assume a liderança no processo de aperfeiçoamento do software e é um proponente fundamental e patrocinador de programas educacionais.

Gerência dos fornecedores. Adquirem-se três categorias de software de fornecedores externos de software — pacotes prontos, comerciais (por exemplo, Microsoft Office, oferecidos ao usuário em embalagens), um shell personalizado que fornece um esqueleto básico, personalizado de acordo com as necessidades do comprador e software sob encomenda que é projetado e construído de forma personalizada a partir de especificações fornecidas pela empresa-cliente. O papel do grupo de SQA é garantir software de alta qualidade por meio da sugestão de práticas específicas de garantia da qualidade que o fornecedor deve (sempre que possível) seguir, e incorporar exigências de qualidade como parte de qualquer contrato com um fornecedor externo.

Administração da segurança. Com o aumento dos crimes cibernéticos e novas regulamentações governamentais referentes à privacidade, toda organização de software deve instituir políticas que protejam os dados em todos os níveis, estabelecer proteção através de firewalls para as aplicações da Internet (WebApps) e garantir que o software não tenha sido alterado internamente, sem autorização. A SQA garante o emprego de processos e tecnologias apropriadas para ter a segurança de software desejada.



Proteção. O fato de o software ser quase sempre um componente fundamental de sistemas que envolvem vidas humanas (por exemplo, aplicações na indústria automotiva ou aeronáutica), o impacto de defeitos ocultos pode ser catastrófico. A SQA pode ser responsável por avaliar o impacto de falhas de software e por iniciar as etapas necessárias para redução de riscos.

Administração de riscos. Embora a análise e a redução de riscos sejam preocupação dos engenheiros de software, o grupo de SQA garante que as atividades de gestão de riscos sejam conduzidas apropriadamente e que planos de contingência relacionados a riscos tenham sido estabelecidos.

Além de cada uma dessas preocupações e atividades, a SQA trabalha para garantir que atividades de suporte ao software (por exemplo, manutenção, suporte on-line, documentação e manuais) sejam realizadas ou produzidas tendo a qualidade como preocupação dominante.

Fonte: Pressman e Maxim (2016, p. 365).



eu recomendo!



livro

Tecnologia da Informação: Qualidade de Produto de Software

Autor: Ana Cervigni Guerra e Regina Maria Thienne Colombo

Editora: PBPQ Software

Sinopse: o livro trata de assuntos relacionados à engenharia de software, em especial, a qualidade de software. Apresenta conceitos fundamentais sobre essa qualidade, 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 realizada a avaliação de um software. Estes conceitos são importantes para quem pretende entrar no mercado de trabalho.



filme

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.





eu recomendo!



conecte-se

Uma Abordagem de Garantia de Qualidade de Processos e Produtos de Software com Apoio de Gerência de Conhecimento na Estação TABA

O artigo apresenta uma abordagem de garantia de qualidade de processos e produtos de software com o apoio da Gerência de Conhecimento na Estação TABA, um ambiente de desenvolvimento de software orientado à organização.

https://www.researchgate.net/publication/228347581_Uma_Abordagem_de_Garantia_de_Qualidade_de_Processos_e_Produtos_de_Software_com_Apoio_de_Gerencia_de_Conhecimento_na_Estacao_TABA

Priorização de Requisitos e Avaliação da Qualidade de Software Segundo a Percepção dos Usuários

O artigo traz uma abordagem metodológica da priorização dos requisitos de software e a avaliação da qualidade do produto de software, segundo a percepção dos usuários.

<http://revista.ibict.br/ciinf/article/download/1308/1486/1965>



Caro(a) aluno(a),

Chegamos ao final do nosso estudo sobre Qualidade de Software. Espero que você tenha gostado e conseguido entender os conceitos importantes que fazem parte dessa qualidade.

Estudamos os conceitos sobre a qualidade e apresentamos os conceitos e suas definições. Foi importante você compreender o que é qualidade e porque ela é importante para a empresa e os clientes e como está presente no nosso dia a dia. Também falamos sobre as definições de qualidade de produtos, assim como a importância da qualidade nos processos de desenvolvimento do software.

Na sequência, estudamos os modelos de melhoria e avaliação de processo de software e, depois, que a qualidade de um software está diretamente relacionada à qualidade do seu processo. Assim, foi estudado o modelo de melhoria da maturidade de processo de software CMMI (Capability Maturity Model Integration), o modelo qualidade de processo de software: Modelo MPS.BR, a norma de qualidade de software ISO/IEC 12207: processos de ciclo de vida de software, a qual estabelece uma estrutura comum para tais processos. Por fim, aprendemos a norma ISO/IEC 15504 – SPICE.

Estudamos os requisitos e a avaliação da qualidade de software, os conceitos, o processo de avaliação da qualidade, o processo de avaliação da qualidade de produto de software, aqueles usados para pacotes de software e como são oferecidos e liberados para uso no mercado e, também, como é aplicada a norma NBR ISO/IEC 25051.



Também vimos os requisitos de usabilidade na interface de software e como a norma ISO/IEC 92411-11 é usada para esclarecer os benefícios de medir a usabilidade, quando pensamos em desempenho e satisfação do usuário. Na quarta unidade, aprendemos a importância das métricas de software para a qualidade do produto, foram expostos os principais conceitos relacionados às métricas de software e a importância da revisão dele, além de como realizar essa revisão, aplicando desde técnicas de inspeção até técnicas de pair programming.

No final, vimos como garantir a melhor qualidade ao software que está sendo criado, apresentamos os principais conceitos usados na garantia da qualidade de software e conhecemos os padrões que são essenciais na construção dele, os quais, normalmente, são feitos por organizações de padronização, como a ISO (Organização Internacional de Padronização) e o IEEE (Instituto de Engenheiros Eletrotécnicos e Eletrônicos). Assim como as abordagens formais da SQA, em que são definidas uma sintaxe e uma semântica mais rigorosas para todas as linguagens de programação existentes e para as criadas futuramente.

Espero ter alcançado o objetivo da disciplina em relação aos conceitos que envolvem a qualidade de software. Desejo que a utilização do que foi apresentado, aqui, seja adotado e lhe garanta sucesso profissional. Obrigada pela atenção em todas as unidades e até a próxima!

ABNT. **NBR ISO 8402**. Gestão da qualidade e garantia da qualidade – Terminologia. Rio de Janeiro: ABNT, 1994.

ABNT. **NBR ISO 9126**. Qualidade de Produto de Software. Rio de Janeiro: ABNT, 2011.

ABNT. **NBR ISO/IEC 20000-1:2018**. Tecnologia da informação – Gestão de serviço. Parte 1: Requisitos do sistema de gestão de serviço. Rio de Janeiro: ABNT, 2018.

ANDRADE, M. **Qualidade de Software**. 1. ed. Rio de Janeiro: Seses, 2015.

ARDUINI, C. Certificação ISO: Quais São as Principais Normas e Como Estão Associadas? **Templum Consultoria**, [2020]. Disponível em: <https://certificacaoiso.com.br/certificacao-iso-as-principais-normas/>. Acesso em: 28 out. 2020.

AZEVEDO, R. Melhoria de Processo de Software, por onde começar? **Lennit**, São Paulo, 7 jun. 2014. Disponível em: <https://www.leanti.com.br/artigos/18/melhoria-de-processo-de-software,-por-onde-comecar.aspx>. Acesso em: 26 out. 2020.

CROSBY, P. B. **Qualidade**: Falando Sério. São Paulo: McGraw-Hill Brasil, 1990.

DEVMEDIA. Melhoria do Processo de Software Brasileiro. **Devmedia**, 2014. Artigos. <https://www.devmedia.com.br/melhoria-do-processo-de-software-brasileiro/29915>. Acesso em: 26 out. 2020.

DIAS, R. P. Os Processos de Software. **Contexto Delimitado**, 20 ago. 2019. Disponível em: <https://medium.com/contexto-delimitado/os-processos-de-software-56a2e70fddfb>. Acesso em: 26 out. 2020.

DUARTE, K. C.; FARBO, R. de A. **Uma Ontologia de Qualidade de Software**. Vitória: UFES, 2006.

FILHO, R. C. S.; ROCHA, A. R.; TRAVASSOS, G. H. O Uso de Projetos-Piloto para Avaliação da Efe-
tividade da Melhoria de Processos. In: SIMPÓSIO BRASILEIRO DE QUALIDADE DE SOFTWARE
– SBQS'06, 5., 2006, Vila Velha. **Anais** [...]. Vila Velha: SBQS'06, 2006.

FIORINI, M. Technology and Business Model Foresight for Border and Coastal Surveillance Systems. **International Journal of e-Navigation and Maritime Economy**, v. 13, p. 117-127, 2019. Disponível em: <https://www.researchgate.net/publication/338357326>. Acesso em: 28 out. 2020.

FREITAS, J. A. de. **Projeto, Implementação e Teste de Software**. Maringá: Unicesumar, 2019.

GUERRA, A. C.; COLOMBO, R. M. T. **Tecnologia da Informação**: Qualidade de Produto de Sof-
tware. [S. I.]: PBQP Software, 2000.

IEEE. **IEEE Std 1063-2001**. IEEE Standard for Software User Documentation. New York: IEEE, 2001.

- ISO. **ISO 9127:1988.** Information processing systems – User documentation and cover information for consumer software packages. Geneva: ISO, 1988.
- ISO. **ISO 9241-11:2018** – Ergonomics of human-system interaction – Part 11: Usability: Definitions and concepts. Geneva: ISO, 2018.
- ISO. **ISO/IEC TR 15504-1:1998** – Information technology – Software process assessment – Part 1: Concepts and introductory guide. Geneva: ISO, 1998.
- JURAN, J. M.; DEFFEO, J. A. **Fundamentos da Qualidade para Líderes.** 1. ed. Porto Alegre: Bookman, 2015.
- KOSCIANSKI, A. et al. **Guia para Utilização das Normas sobre Qualidade de Produto de Software ISO/IEC 9126 e 14598.** [S. l.: s. n.]: maio 1999. Disponível em: https://www.researchgate.net/publication/326984869_Guia_para_Utilizacao_das_Normas_Sobre_Avaliacao_de_Qualidade_de_Produto_de_Software_-_ISOIEC_9126_e_ISOIEC_14598. Acesso em: 28 out. 2020.
- MENDES, R. **Modelagem e Avaliação do CMMI no SPEM para Definição de um Meta-Processo de Software.** 2005. 83f. Trabalho de Conclusão de Curso (Curso de Ciência da Computação) – Centro de Informática, Universidade Federal de Pernambuco, Recife, 2005.
- MORO, R. D.; FALBO, R. de A. Uma Ontologia para o Domínio de Qualidade de Software com Foco em Produtos e Processos de Software. In: WORKSHOP ON ONTOLOGIES AND META-MODELING IN SOFTWARE AND DATA ENGINEERING – WOMSDE, 3., 2008, Campinas. **Anais** [...]. Campinas: WOMSDE, 2008. Disponível em: https://www.researchgate.net/publication/229042029_Uma_Ontologia_para_o_Dominio_de_Qualidade_de_Software_com_Foco_em_Produtos_e_Processos_de_Software. Acesso em: 28 out. 2020.
- PADILHA, A. M. **MAPS 1550:** Uma Metodologia de Avaliação de Processo de Software para o EXPSEE Baseado na ISO/IEC15504. 2000. 176f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Computação, Universidade Estadual de Campinas, Campinas, 2000.
- PIRES, F. Modelo CMM de Qualidade de Software. **Informativo Técnico** nº 70 – Revista de Informação e Tecnologia. Campinas: Unicamp, jan./fev. 2000.
- PRESSMAN, R. S. **Engenharia de Software:** uma abordagem profissional. 7. ed. Porto Alegre: AMGH, 2011.
- PRESSMAN, R. S.; MAXIM, B. **Engenharia de Software:** uma abordagem profissional. 8. ed. Porto Alegre: McGraw-Hill Brasil, 2016.
- RETAMAL, A. M. CMMI – Quem precisa dele? **Revista Clube Delphi**, ed. 62, maio 2005. Disponível em: <https://www.devmedia.com.br/artigo-clube-delphi-62-cmmi/13856>. Acesso em: 26 out. 2020.

REZENDE, D. A. **Engenharia de Software e Sistemas de Informação**. 3. ed. Rio de Janeiro: Brasport, 2005.

ROCHA, A. R. *et al.* Fatores de Sucesso e Dificuldades na Implementação de Processos de Software Utilizando o MR-MPS e o CMMI. *In:* WORKSHOP DE IMPLEMENTADORES MPS.BR, 1., 2005, Brasília. **Anais** [...]. Brasília: Softex-MPS, 2005.

RODDEN, K.; HUTCHINSON, H.; FU, X. Measuring the User Experience on a Large Scale: User-Centered Metrics for Web Applications. *In:* INTERNATIONAL CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, 28., 2010, Atlanta. **Proceedings** [...]. Atlanta: ACM CHI, 2010. Disponível em: https://www.researchgate.net/publication/221516100_Measuring_the_User_Experience_on_a_Large_Scale_User-Centered_Metrics_for_Web_Applications. Acesso em: 28 out. 2020.

SILVA, F. R. D. da; ALMEIDA, S. A. de. ISO/IEC 14598. **Instituto Federal de Educação, Ciência e Tecnologia Fluminense Campus Campos-Centro**, Rio de Janeiro, [2020]. Normas de Produto. Disponível em: <https://sites.google.com/site/a109iff/teste/iso-14598>. Acesso em: 28 out. 2020.

SOFTEX. **Guia Geral MPS de Software**. [S. l.]: Softex, jan. 2016. Disponível em: https://www.softex.br/wpcontent/uploads/2016/04/MPS.BR_Guia_Geral_Software_2016-com-ISBN.pdf. Acesso em: 26 out. 2020.

SOMMERVILLE, I. **Engenharia de Software**. 8. ed. São Paulo: Pearson Education, 2011.

UNIDADE 1

1. B.
2. E.
3. A.
4. Qualidade do processo: deve garantir que o projeto realize todos os processos necessários para atender aos requisitos. Auditar os processos de acordo com os padrões e procedimentos previamente estabelecidos, utilizando métodos e procedimentos para comparar o que foi realizado com o que deve ser feito. Qualidade de produto: descobrir os defeitos gerados ao longo do projeto e eliminar suas causas, por meio de testes e revisões, utilizando check-lists e casos de uso de testes comparando o que foi feito pelo que é esperado pelo cliente.
5. Os fatores de qualidade são classificados em duas grandes categorias: fatores que podem ser medidos diretamente (por exemplo, defeitos revelados durante testes) e fatores que podem ser medidos apenas indiretamente (por exemplo, usabilidade ou facilidade de manutenção).

UNIDADE 2

1. 1. Acarreta a empresa construir uma:
 - Infraestrutura com processos eficazes, que sejam utilizáveis e consistentes e que possam ser aplicados em toda a empresa.

- Cultura corporativa que usa os métodos, os procedimentos e as práticas dos negócios para que perdurem na empresa.

2. C.
3. B.
4. Os níveis de maturidade e suas áreas de processo são: processos fundamentais (aquisição, gerência de requisitos, desenvolvimento de requisitos, solução técnica, integração, instalação e liberação do produto). Processo organizacionais (gerência de projeto, adaptação do processo para gerência de projeto, análise de decisão e resolução, gerência de riscos, avaliação e melhoria do processo organizacional, definição e desempenho do processo organizacional, gerência quantitativa do projeto, análise e resolução de causas, inovação e implantação na organização). Processos de apoio: garantia da qualidade, gerência de configuração, validação, medição, verificação, treinamento.
5. O processo de adaptação define atividades que são necessárias para a norma se adaptar melhor às tarefas desenvolvidas pela empresa, em seus projetos. Esta adaptação ocorre com base nos fatores que diferenciam uma empresa de outras, por exemplo, estratégias de aquisição, os modelos de ciclo de vida do projeto, as características do software e a cultura organizacional.

UNIDADE 3

1. 1. Parte 2: planejamento e gestão, etapa de apoio ao processo de avaliação de produto e apoio ao processo de avaliação de produto e estão relacionadas ao suporte e à gestão da avaliação corporativa ou departamental, fornecendo exemplos de medidas que correspondem a uma subcaracterística.
 - . Parte 5: processo para avaliadores: etapa de requisitos e orientações para o processo de avaliação de projeto na situação de incluir a avaliação de terceira parte.
2. Especificamente, a norma NBR ISO/IEC 14598-3 trata do processo de avaliação para desenvolvedores de software e é aplicada em todas as fases do ciclo de vida de desenvolvimento. O foco dessa norma é a seleção de indicadores que preveem a qualidade do produto final, por meio da medição da qualidade de produtos intermediários.
3. 3. C.
4. 4. E.
5. 5. Os conceitos num processo de avaliação são:
 - Medição: aplicação de uma medida de qualidade de software a um produto específico. A partir desta ação, é gerado o relatório qualitativo de avaliação.
 - Pontuação: a partir das métricas e medidas obtidas, as informações são transformadas em um sistema numérico preestabelecido de pontuação. A partir

desta transformação, é gerado outro relatório qualitativo de avaliação.

- Julgamento: a partir do relatório qualitativo, passamos à emissão de um juízo sobre o nível de qualidade do produto de software.

UNIDADE 4

1. A.
2. B.
3. C.
4. A técnica de pair programming não é muito difícil de entender e é muito utilizada. Aqui, o código fonte é compartilhado entre dois programadores, e ambos trabalham programando e analisando o código criado, ao mesmo tempo e no mesmo equipamento. Como característica principal dessa técnica, os programadores se intercalam nas ações, ou seja, ora um programador codifica, ora outro analisa o trabalho, depois, realizam a troca de papéis, analisando as sintaxes, métodos e tudo que pode ser otimizado e simplificado. Com isso, a técnica possibilita a troca de informações e conhecimento entre os pares de programadores, o que melhora a implementação.
5. A técnica de inspeção de software é um tipo de revisão aplicado a todos os artefatos presentes, analisa praticamente tudo, sendo considerado rigoroso na sua inspeção. Como dito, o processo de revisão pode ser realizado em todo artefato presente.

UNIDADE 5

1. C.
2. D.
3. E.
4. A confiabilidade pode ser medida diretamente e estimada usando dados históricos e de desenvolvimento; e mais, é definida de forma estatística, como a probabilidade de realizar operações de um programa de computador, sem falhas, em dado ambiente por determinado espaço de tempo.
5. Seis sigma é a estratégia para a estatística da garantia da qualidade mais utilizada na indústria atual. Essa estratégia é uma metodologia rigorosa e disciplinada que utiliza a análise estatística e de dados para a média e para melhorar o desempenho operacional de uma empresa, por meio da identificação e da eliminação de defeitos em processos de fabricação de software.



anotações