QUALIDADE DE SOFTWARE

AULA 1

Profª Maristela Weinfurter

CONVERSA INICIAL

FUNDAMENTOS DE QUALIDADE DE SOFTWARE

Nesta aula, vamos falar sobre processos de qualidade para o desenvolvimento de softwares e sobre o produto de software com qualidade.

A qualidade de software muitas vezes é tratada como uma disciplina dentro da engenharia de software, no entanto, não há como falar de engenharia de software sem qualidade e vice-versa.

Sendo assim, a tão falada crise de software dentro da área de engenharia de software de certa forma é o gatilho inicial para os primeiros trabalhos, pesquisas e experimentos em relação à aplicação de processos de desenvolvimento de software com critérios de qualidade, considerando a entrega de produtos que atendam às necessidades das partes envolvidas.

Quando falamos em qualidade nesse contexto, não consideramos apenas o código gerado durante o processo, mas também os requisitos, as métricas utilizadas em gerenciamento, entre outras ferramentas e artefatos.

Há um gasto exorbitante quando um produto é feito sem a observação de critérios de qualidade. Considere, assim, que por várias décadas as funcionalidades de códigos têm estado comprometidas, com falta de usabilidade, requisitos não atendidos, além de problemas de implantação e manutenção. Enfim, diferentes momentos do processo vêm comprometendo o produto e o processo. Em especial, o momento de manutenção e adaptação, que gera um retrabalho considerável, acarretando problemas financeiros e outros problemas que nunca se extinguem para engenheiros e desenvolvedores.

Quando um software fica inativo ou funciona precariamente, clientes são prejudicados, são notas fiscais não são emitidas, pagamentos não são feitos, negócios deixam de ser concretizados,

aplicativos são substituídos por concorrentes – enfim, o prejuízo é grande.

A forma de melhorar (ou pelo menos minimizar) tais situações é através da implantação de uma boa gerência e de um bom controle de qualidade. O trabalho inicia com os requisitos de um software, e se estende até a sua implantação e manutenção. Para isso, temos hoje normas, padrões, técnicas, ferramentas e muitos profissionais especializados em pensar, planejar e executar atividades relacionadas à qualidade de software.

Grandes empresas já estão na estrada por um bom tempo, com vários aparatos de qualidade em seus processos; porém, quando falamos de empresas pequenas, como podemos estabelecer um processo de qualidade, uma vez que em geral faltam recursos financeiros?

Vamos entender, ao longo da caminhada, que nem tudo está atrelado à gerência de qualidade, mas ainda assim os testes mais simples podem ser feitos pela própria equipe. Tudo é uma questão de aplicar o que é minimamente possível.

Portanto, qualidade de software não tem a ver somente com profissionais de qualidade, pois precisamos considerar todos que trabalham e atuam com desenvolvimento de software.

TEMA 1 – CONCEITOS DE QUALIDADE DE SOFTWARE

O termo *qualidade de software* vem sendo lapidado há mais de 30 anos. Ele surgiu em um momento histórico de desenvolvimento de software, justamente para tratar de assuntos inerentes à melhoria, tanto em termos de processos quanto do produto de software. A ideia de qualidade de software ganha maiores proporções nos anos 1990, quando grandes empresas, que faziam uso de vários tipos de sistemas de informações, tais como os ERPs (Enterprise Resource Planning), tiveram que arcar com bilhões de dólares desperdiçados em software, pois eles não entregavam as características e funcionalidades prometidas nos requisitos iniciais (Pressman, 2011).

Como o estudo sobre engenharia de software foi desenhada sobre o desespero de gerentes de projetos de software, analistas de sistemas, engenheiros e programadores, não há como dissociarmos os conceitos de qualidade de software dos fundamentos da disciplina. O debate sobre qualidade de software, por mais abrangente que seja, está intimamente ligado à engenharia de software.

Aliás, vamos voltar um pouco no contexto histórico do software! Aquilo que chamamos de *crise do software* decorre da complexidade inerente ao gerenciamento e ao desenvolvimento de sistemas e aplicações. De forma árdua, considerando o desperdício, com noites sem dormir, finalmente o pessoal de "dev" parou para refletir sobre o que estavam fazendo e como estavam agindo. Mesmo diante de tal episódio, em 1968, levamos mais vinte anos (sim, você leu corretamente, vinte anos!) para perceber que era necessário estabelecer modelos, regras, padrões, critérios e teorias relacionados à qualidade de software.

Finalmente, o nosso pessoal de "dev" olhou ao redor e viu que outras áreas de negócio, especialmente as áreas de engenharia da produção, falavam, disseminavam e utilizavam controle da garantia da qualidade em seus processos fabris. Assim, o engenheiro de software passou a imitar o que havia de melhor nas outras engenharias para a resolução de falhas, erros, projetos não terminados, projetos mal dimensionados e projetos que não tinham um final feliz.

Com esse novo posicionamento na área de desenvolvimento, começaram a aparecer normas e padrões ISO (9000-3, 9126, 15.504, 12.207, 1074, 1298), além de modelos (CMM, CMMI e MPS.BR). Entrelaçada a tais padrões, normas e modelos, a qualidade de software foi passando por um processo evolutivo de detalhamento. Tais detalhamentos fundamentam características importantes, como a preocupação com cultura e ética no processo de engenharia de software, custos de qualidade, melhoria do processo e do produto, segurança de software, verificação, validação, revisão e auditoria, ressignificação do processo de engenharia de requisitos de qualidade de software, caracterização e distinção entre defeitos e erros, gestão da qualidade de software e de processo, além da construção de métricas e ferramentas capazes de dar suporte à qualidade de software. A Figura 1 nos remete a algumas ideias pertinentes aos conceitos fundamentais de qualidade de software.

Figura 1 – Nuvem de palavras sobre itens relacionados ao teste de software

Crédito: master_art/Shutterstock.

Observem que a nuvem de palavras traz duas em evidência: software e testes. Todas as demais são importantes e fazem parte do nosso ecossistema de qualidade de software, mas, conforme vamos compreendendo melhor sobre o que vem a ser a qualidade de software, percebemos que o assunto "testes" tem conquistado grande importância dentro da ideia de qualidade. Isto fica claro tanto no nível teórico quanto no dia a dia de trabalho dentro das empresas. Porém, não apenas os testes são importantes para a área de qualidade de software, pois há duas referências importantes no mundo da engenharia e da computação que tratam do assunto em questão de forma teórico-experimental: IEEE (Institute of Electrical and Electronics Engineering) e ACM (Association for Computing Machinery). O IEEE é responsável por promoção da engenharia de criação, desenvolvimento, integração, compartilhamento e conhecimento aplicado a tudo que se refere à ciência e a tecnologias de eletricidade e informação. A ACM é uma sociedade para educadores, pesquisadores e profissionais de computação. Hoje é a maior sociedade de computação do mundo, fortalecendo a voz coletiva da profissão, com padrões que ajudam no reconhecimento da excelência técnica.

A IEEE e a ACM patrocinaram a criação do guia SWEBOK – "Guide to the Software Engineering Body of Knowledge" (IEE Computer Society, 2022). Esse guia é uma iniciativa para a criação de processos e métodos para a Engenharia de Software, oferecendo uma classificação de suas áreas de conhecimento. Ele divide a Engenharia de Software em dez áreas, entre elas a qualidade. A Figura 2 estabelece algumas subáreas da qualidade de software, segundo o SWEBOK.

Até aqui, falamos de qualidade de software, mas é importante refletir sobre o que é a qualidade. Qualidade, em termos gerais, refere-se às características desejáveis de quaisquer tipos de produtos

ou serviços. Tais características precisam ser medidas e aferidas para que possam ser garantidas. Quando falamos de qualidade de software, não é diferente. Ela envolve processos, ferramentas e técnicas que devem ser utilizados para que tais características sejam alcançadas. A conceituação é tão ampla que às vezes acaba sendo divergente. Nesse contexto, há quem diga que a qualidade de software deve estar em conformidade com os requisitos, considerando ainda níveis excelentes de aptidão para uso. Finalmente, é um aspecto que deve ser orientado para o mercado, ambiente em que o cliente é o árbitro.

Resumidamente, a qualidade de software deve atender aos requisitos estabelecidos com precisão, permitindo que as partes interessadas sejam atendidas em suas expectativas e desejos. Para que isso ocorra, é necessário que o processo de desenvolvimento do software esteja baseado em um plano de garantia da qualidade para todas as etapas do projeto, prevenindo e eliminando possíveis erros e defeitos. Lembramos aqui que a qualidade não é apenas uma fase dentro do ciclo de desenvolvimento de software, mas sim um conceito que nasce desde a primeira ideação do software e se estende até a entrega final, com continuidade, considerando ainda a manutenção posterior à entrega do projeto.

A Figura 2 que traz vários pilares que dão sustentação ao mundo da qualidade de software, o que nos ajuda a compreender a amplitude desse contexto.

Figura 2 – Pilares da qualidade de software

O pilar **Fundamentos da Qualidade de Software** aborda questões relacionadas a atividades de planejamento, execução e melhoria contínua dos processos e do software. É subdividido em:

- engenharia de software, cultura e ética;
- custos e valores em qualidade;
- modelos e características da qualidade;
- melhorias em qualidade de software; e
- segurança de software.

Quando pensamos em **Processos de Gerenciamento de Qualidade de Software**, o nosso foco são técnicas e procedimentos que suportam a identificação de erros em artefatos de software. Como principais subtópicos, temos:

- garantia da qualidade de software;
- validação e verificação; e
- revisões e auditorias.

Dentro do tópico **Considerações Práticas**, trabalhamos com técnicas, procedimentos, medições e com o monitoramento da qualidade do processo e do produto. Os subtópicos são:

- requisitos de qualidade de software;
- caracterização de defeitos;
- técnicas de gerenciamento de qualidade de software; e
- métricas em qualidade de software.

Não menos importante, o tópico **Ferramentas de Qualidade de Software** engloba tudo que pode facilitar a aplicação da qualidade, tanto nos processos de desenvolvimento de software quanto no próprio produto (software) em questão.

Estabelecemos o que é a qualidade em seu formato genérico, analisando a qualidade de software e seus principais pilares. Criar software com qualidade não foi, não é e não será, tão cedo, uma tarefa fácil. Afinal, o processo envolve um conjunto de preocupações técnicas e documentais para que, no final, o produto funcione de forma simples, rápida, eficiente, multiplataforma, adaptável, segura, com falhas tendendo a zero, entre outros aspectos necessários quando usamos um software. **Nossa experiência precisa ser a melhor possível!**

TEMA 2 – PADRÕES DE QUALIDADE DE SOFTWARE

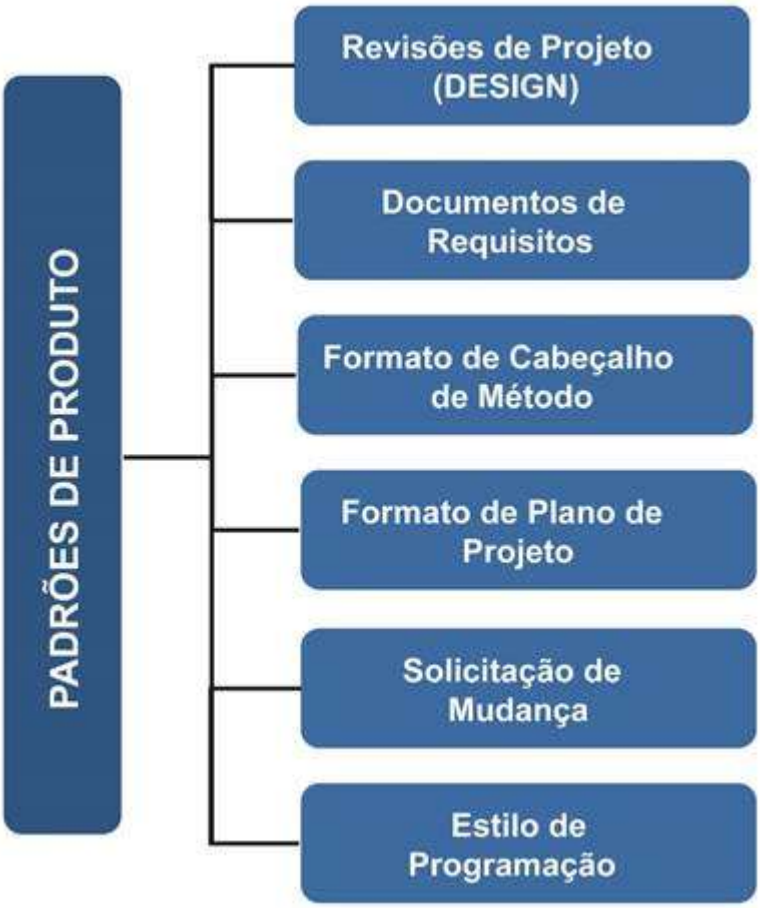
Um dos termos mais utilizados dentro da área de qualidade é o padrão. Padrão, segundo o dicionário, traduz a ideia de que um objeto qualquer pode ser utilizado como modelo para a elaboração de um novo objeto.

Não é diferente na área de software. Padrões de software são importantes porque se baseiam no conhecimento sobre a forma mais adequada e prática para as empresas. Também auxiliam na reutilização da experiência para que os erros do passado sejam evitados. Como a qualidade é algo subjetivo, a utilização de padrões estabelece uma base para a tomada de decisão sobre o nível de qualidade a ser atingido no projeto de software. Além disso, padrões permitem que a continuidade do trabalho por novos engenheiros seja mais tranquila, acarretando menos esforço de aprendizagem para que outros profissionais possam dar prosseguimento às atividades de desenvolvimento de software.

Ao observar essa concepção de padrão, precisamos nos voltar aos padrões estabelecidos dentro da área de qualidade de software, sob dois aspectos.

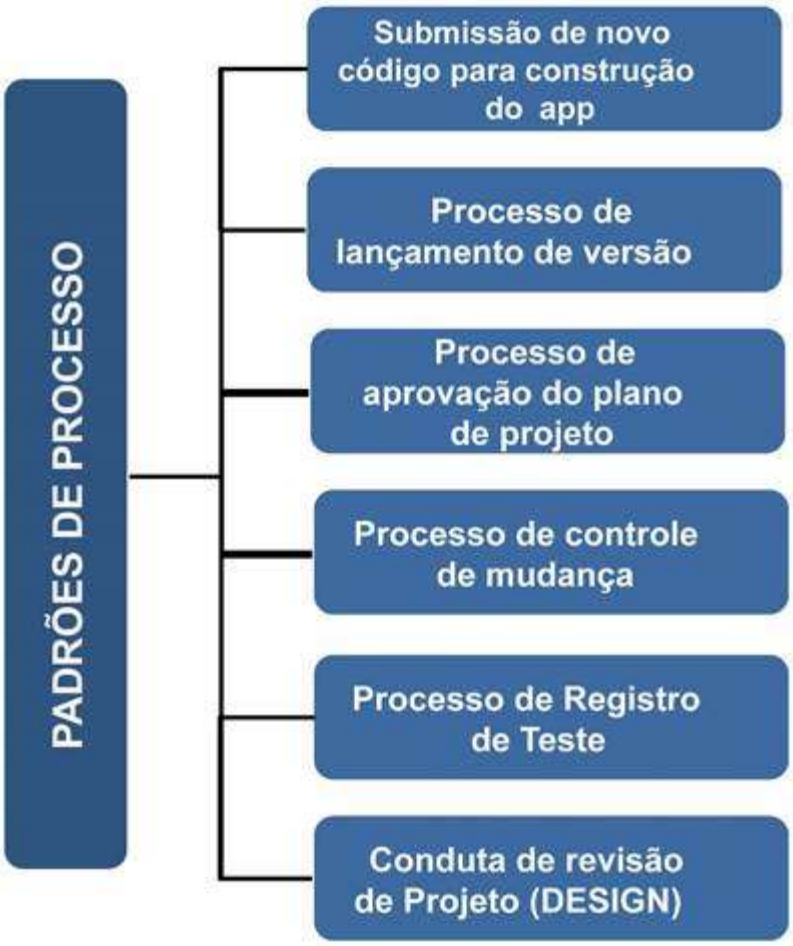
- Padrões de produto (produto = software): estes padrões incluem a documentação, a estruturação dos requisitos, a definição das classes, bem como padrões para a própria codificação dentro de uma linguagem de programação. A figura a seguir trata de alguns exe mplos de padrões de produto.

Figura 3 – Exemplos de padrões de produto



- Padrões de processo (para desenvolver o software): produtos que são desenvolvidos com padrões de qualidade requerem que os processos também tenham garantia de qualidade. Nos padrões de processos, encontramos boas práticas de desenvolvimento, com definições de processos e especificações, validação, e ferramentas de apoio a processos e documentação. A figura a seguir exemplifica padrões relacionados ao processo de desenvolvimento de software.

Figura 4 – Exemplos de padrões de processo



Qualidade de produto e de processo serão inseparáveis em nossa viagem daqui por diante.

Para melhorar a nossa compreensão sobre padrões, vamos partir de um exemplo de aplicação de padrões no desenvolvimento de códigos. Para tanto, vamos utilizar **estilos de programação** dentro de **padrões de produto** (Figura 3). Para exemplificar esse caso, vamos utilizar o PEP 8 – Style Guide for Python. O PEP 8 é um documento que especifica convenções sobre a padronização de códigos escritos em linguagem de programação Python. Ele é muito similar ao PEP7, que foi idealizado para linguagem de programação C. Ou seja, cada linguagem de programação pode ter o seu próprio guia de estilo de codificação.

Voltando ao nosso caso, no início do **PEP 8**, há uma descrição sobre a formatação do código, com três tópicos importantes.

- **Indentação**: nos sugere a utilização do "Python-mode" do Emacs, com quatro espaços por nível de indentação.
- **Tabulações**: sugere que você não misture tabulações e espaços.

- **Comprimento máximo de linhas:** sugere a limitação de 80 caracteres, pois há muitos monitores limitados a 80 colunas. Emacs quebram as linhas em 80 caracteres.
- **Linhas em branco**: funções e definições de classe utilizam duas linhas em branco. Métodos dentro de uma classe são separados com uma linha em branco. Linhas extras para a separação de grupos de funções.
- **Import**: a cláusula **import** deve ser utilizada sempre em linhas separadas.

Vamos utilizar o item 5 da formatação correta de códigos em Python. No Quadro 1, vemos a forma errada e a forma correta. Isto não quer dizer que pela sintaxe a primeira coluna iria retornar algum erro, mas sim que a coluna 2 nos demonstra uma forma mais "limpa" de codificar nessa linguagem de programação.

Quadro 1 – Comparação entre escrita errada e correta na utilização da cláusula import

Forma errada	**Forma correta**
import sys, os	import sys import os

O que vimos até aqui são pequenos exemplos de como a padronização nos encoraja na escrita códigos mais limpos, com a especificação de projetos de forma mais clara e menos ambígua, além do estabelecimento de critérios de reutilização de código, entre tantos outros padrões que nos auxiliam na melhoria contínua do desenvolvimento de software.

TEMA 3 – MEDIÇÃO DE SOFTWARE

Suponha que você baixou um aplicativo no seu smartphone, que foi entregue no prazo, dentro do orçamento, executando de forma correta e eficiente todas as funções especificadas. Isso parece definição de felicidade, correto? Nem sempre! Mas como não?

Vejamos:

- O aplicativo pode ser difícil de entender e difícil de ser operado. Isto poderá levar a custos adicionais de manutenção para torná-lo mais simples! Essa situação implica quase uma "nova" construção de várias partes do aplicativo.

- O aplicativo pode ser induzido a um uso indevido, com os resultados capazes de gerar problemas para a empresa que o desenvolveu, como problemas de segurança com os dados.
- O aplicativo pode ter sido integrado a uma determinada plataforma. No momento de adaptá-lo a outra plataforma, às vezes temos quase que reescrever o código para que funcione adequadamente. Imagine que o aplicativo funcione muito bem no Android, porém, por mais que se tenha optado por um framework multiplataforma, foram utilizadas muitas características de baixo nível para acessar recursos do smartphone. Nessa situação, é provável que uma equipe precise modificar e atualizar determinadas funcionalidades para que a mesma aplicação execute no iOS.
- Para encerrar, alguns usuários gostariam de utilizar essa aplicação apenas na web. Sendo assim, estamos falando de outra equipe, que adapta o mesmo aplicativo para que seja executado adequadamente em outros tipos de equipamentos, como notebooks e desktops.

Compreendemos, assim, que nem tudo está associado a custos, prazos e entregas, pois precisamos garantir um gerenciamento adequado, minucioso, ágil e com qualidade total.

Vamos agora à contextualização de métricas necessárias para processar os critérios de qualidade no desenvolvimento de software.

3.1 CONTEXTUALIZAÇÃO DE MÉTRICAS NA HISTÓRIA

Quando falamos que o tema da qualidade de software vem de longa data, precisamos voltar mais de 30 anos no tempo, muito antes do desenvolvimento do atual ecossistema tecnológico em que estamos imersos.

Mesmo com livros ou artigos anteriores a 1976, vamos nos basear em um framework para o estabelecimento de critérios de qualidade de software, a partir do artigo “Avaliação quantitativa da qualidade do software”, de Boehm, Brown e Lipow. (1976). Trata-se de um clássico da ciência da computação na plataforma da acm. Descreve um estudo para o estabelecimento de uma estrutura conceitual com alguns resultados iniciais importantes para a análise das características da qualidade de software. A hipótese inicial era de que, quando conseguimos dar atenção às características da qualidade do software, obtemos economias significativas nos custos em todo o ciclo de vida do software.

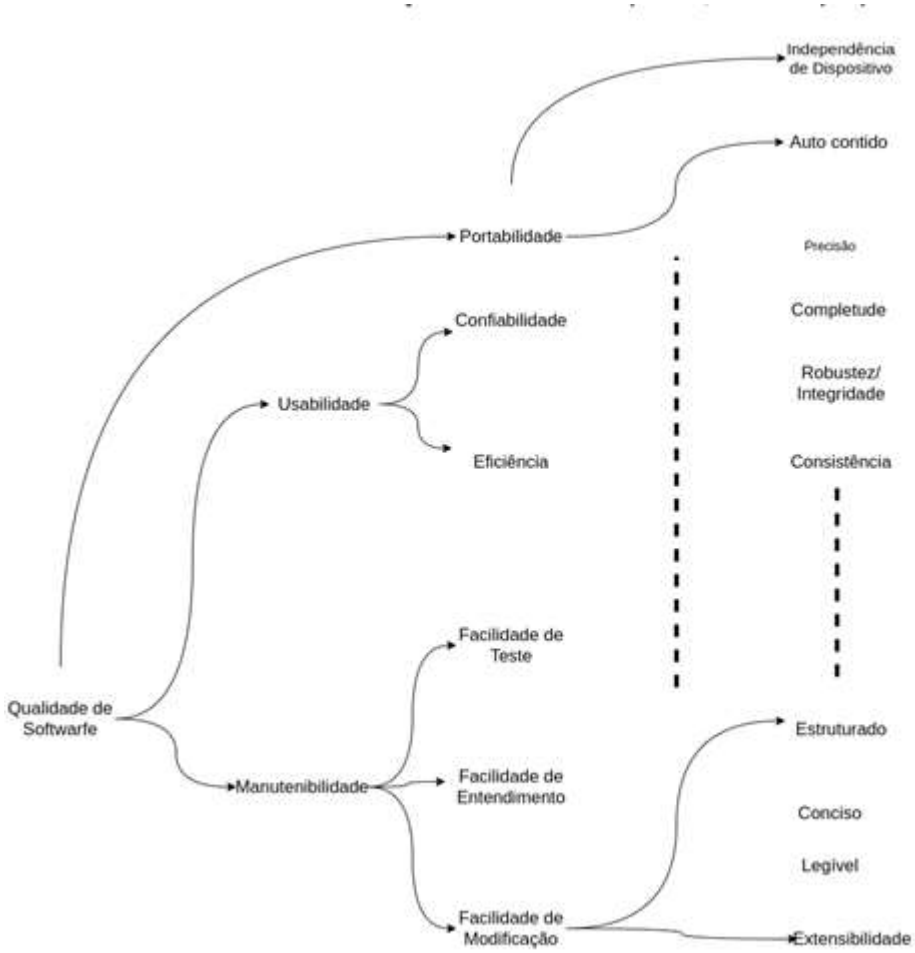
À época, um limitante era a capacidade de avaliação automática e quantitativa da qualidade de software, o que não é mais um problema na atualidade. O artigo traz uma proposta de hierarquia de características, com usos reais para a avaliação da qualidade de software. Com essa hierarquia, foram definidas, classificadas e avaliadas métricas de avaliação de qualidade de software. Um dos benefícios potenciais dessas métricas é a possibilidade de quantificação e a facilidade para a automatização do processo de qualidade.

O estudo definiu, pela primeira vez, uma estrutura clara para questões que por anos foram desafios à qualidade de software. Foi o pontapé para o que veremos adiante. A Figura 5 ilustra esse trabalho brilhante, que deu margem a tantas outras evoluções até o presente momento.

O artigo traz dois pontos importantes e interessantes. Primeiramente, a classificação de 224 erros de software em 13 categorias principais.

- Erros na preparação ou processamento dedados de entrada do cartão.
- Erros de manuseio de fita.
- Erros de manipulação de disco.
- Erros de processamento de saída.
- Erros de processamento de mensagens de erro.
- Erros de interface de software.
- Erros de interface de hardware.
- Erros de interface de base de dados.
- Erros de interface do usuário.
- Erros de computação.
- Erros de indexação e subscrição.
- Erros de procedimentos iterativos.
- Erros de manipulação de bits.

Figura 5 – Árvore das características da qualidade de software



Fonte: Elaborado com base em: Boehm; Brown; Lipow.

Em um segundo momento, foram feitas análises que nos ajudam a compreender os erros de cada fase do desenvolvimento de software. Naquele momento, as fases utilizadas foram:

- definição de requisitos;
- design;
- código e depuração;
- teste de desenvolvimento;
- validação;
- aceitação;
- integração; e
- entrega.

Obviamente, estamos falando de erros remetem aos primeiros sistemas de computadores, que provavelmente executavam em antigos mainframes, ligados à tecnologia daquele momento. No século 21, encontramos muitos pontos em comum, apesar de toda a evolução tecnológica.

Já a análise das fases de desenvolvimento de software não é tão diferente daquilo que fazemos na atualidade, seja em um projeto mais tradicional ou em um modelo ágil.

TEMA 4 – NORMAS E MÉTRICAS

As normas internacionais foram estabelecidas com a ideia de responder a uma pergunta fundamental: qual é a melhor maneira de fazer determinada coisa? Por mais que os fatores pareçam óbvios em relação a pesos e medidas, há 50 anos estamos desenvolvendo uma família de padrões que cobrem quase tudo – construção civil, fabricação de sapatos, redes Wi-Fi, desenvolvimento de software, entre tantas outras áreas do conhecimento.

Tais normas têm por objetivo garantir a confiança dos consumidores em produtos que sejam seguros, confiáveis e de boa qualidade. Envolvem padrões para segurança no trânsito, criação de brinquedos e embalagens seguras, que fazendo do nosso mundo um lugar um pouco mais seguro, sendo reguladas por governos e órgãos especiais ao redor do globo.

Entre as normas internacionais, temos a ISO/IEC 9126-1:2001, que considera a garantia da qualidade de produto de software. Na versão brasileira, temos a NBR 13596. Elas pressupõem métricas, normas de qualidade, características e subcaracterísticas, além de modelos de processos de avaliação e funções de um software.

A ISO/IEC 9126 define a padronização da avaliação da qualidade de software. O título geral da norma é "Engenharia de software - Qualidade do produto", sendo alicerçada em 4 pontos importantes:

- modelo de qualidade;
- métricas externas;
- métricas internas; e
- métricas de qualidade em uso.

Basicamente, propõe atributos de qualidade subdivididos em seis macrocaracterísticas, conforme vemos na figura a seguir.

Figura 6 – Características da ISO/IEC 9126-1:2001

Vamos compreender um pouco cada um dos atributos de qualidade, segundo a ISO/IEC 9126-1:2001. Nossa conceituação se baseia em perguntas, o que facilita o nosso entendimento sobre as características e subcaracterísticas.

1. **Funcionalidade**: as funções do software satisfazem as necessidades do usuário? Subcaracterísticas:
    a. **adequação**: o software faz o que é adequado?
    b. **acurácia**: o software faz o que é proposto de forma correta?
    c. **interoperabilidade**: o software é integrável a outros sistemas?
    d. **segurança**: o software não autoriza acesso a programas e dados?
    e. **conformidade**: o software obedece a normas e leis?
2. **Confiabilidade**: o software é imune a falhas? Subcaracterísticas:
    a. **maturidade**: qual a frequência de falhas por defeitos no software?
    b. **tolerância a falhas**: quando ocorrem falhas, como o software reage?
    c. **recuperabilidade**: é possível recuperar dados em caso de falhas?
3. **Usabilidade**: o software é fácil de ser utilizado? Subcaracterísticas:
    a. **compreensibilidade:** o software é de fácil entendimento em relação ao conceito lógico e sua aplicabilidade?
    b. **apreensibilidade**: é fácil de aprender a usar?
    c. **operacionalidade**: é fácil de operar e controlar?
4. **Eficiência**: o software é rápido e simples? Subcaracterísticas:
    a. **comportamento durante o tempo**: qual o tempo de resposta e de processamento do software?

b. **comportamento em relação aos recursos**: o software utiliza muito recurso (em quanto tempo)?

5. **Manutenibilidade**: o software é facilmente modificado? Subcaracterísticas:
    a. **analisabilidade**: é fácil de encontrar uma falha quando ela ocorre?
    b. **modificabilidade**: o software é facilmente modificável ou adaptável?
    c. **estabilidade**: há riscos de efeitos inesperados quando é feita alguma alteração?
    d. **testabilidade**: o software é facilmente validado após ser modificado?
6. **Portabilidade**: o software pode ser usado em várias plataformas? Subcaracterísticas:
    a. **adaptabilidade**: o software se adapta a ambientes diferentes?
    b. **instabilidade**: é fácil de ser instalado?
    c. **conformidade**: está de acordo com padrões de portabilidade?
    d. **capacidade de substituição**: é fácil de usar em caso de substituição por outro?

O nosso modelo de qualidade segue, em linhas gerais, o que estudamos até aqui. Apresenta várias características que acabam sendo transformadas em métricas, tarefas e metas, que nos ajudam a alcançar a tão almejada qualidade, tanto em nosso produto (software) quanto no processo de desenvolvimento de software.

4.1 MÉTRICAS, TAREFAS E METAS

A métrica de software é uma característica intrínseca ao software. Pode ser um elemento de documentação ou um evento, ou ainda uma meta dentro do processo de desenvolvimento de software. Tais métricas podem abordar certas situações, como linhas de código, código compreensível (código limpo), falhas, erros ou quaisquer outras previsões em relação ao software ou ao processo.

As métricas são comumente divididas em três principais tipos:

- **Métrica em relação ao tempo** que um determinado processo leva para ser concluído. Podemos pensar em tempo gasto no processo de desenvolvimento por engenheiros, além de técnicos e calendário de desenvolvimento.
- **Métrica em relação aos recursos** necessários para que um processo seja executado. Neste caso, podemos mensurar o esforço total em número de pessoas por dia, custos de viagens e alocação de recursos em nuvem.

- **Métrica em relação às ocorrências** de um determinado evento, que normalmente incluem erros, defeitos, inspeções de código, número de alterações de requisitos e número médio de defeito por linhas de códigos modificadas.

Como utilizamos essas métricas? De duas formas:

- para atribuir o valor da qualidade do software; e
- para que o padrão de um produto seja identificado.

Anteriormente, estudamos todas as características da ISO/IEC 9126-1:2001. Percebemos que, dentre elas, existem atributos de software, internos e externos. Por exemplo, utilizamos o atributo de qualidade externa de usabilidade. A usabilidade corresponde diretamente a atributos de qualidade internos, como número de mensagens de erro e tamanho do manual do usuário. O atributo externo reusabilidade se relaciona com árvore de herança e tamanho do programa em número de linhas. Com tais atributos de qualidade, poderíamos criar várias métricas para avaliação e inspeções de códigos com base em características externas.

Assim, percebemos que, por mais que certas normas e padrões auxiliem na garantia da qualidade de software, cada empresa estabelece as suas métricas em função de necessidades e urgências na avaliação dos problemas, de acordo com a maturidade do processo de desenvolvimento de software. Não existe uma fórmula mágica para a criação dos indicadores. Eles normalmente são atrelados às necessidades de cada empresa.

Retornando ao produto, conseguimos identificar métricas de previsão para quantificar atributos internos de um software. Tais atributos se relacionam com linhas de código, número de métodos associados a classes, acoplamento e coesão, entre outras características internas.

Segundo Ian Sommerville (2018), as métricas de produtos podem ser subdivididas em métricas estáticas e dinâmicas. As dinâmicas são obtidas durante a execução de um software. As medidas estáticas são pensadas em termos de projeto e documentação do software. O Quadro 2 descreve algumas métricas estáticas de produtos de software.

Quadro 2 – Métricas estáticas de produto de software

Métrica de Software	**Descrição**

Fan-in / Fan-out	Fan-in: corresponde ao número de métodos que se chama outro método. Quando temos um número alto de fan-in, significa que a outra função está fortemente acoplada ao projeto, de modo que as mudanças terão um efeito dominó. Fan-out: corresponde ao número de métodos que são chamadas por uma função qualquer. Um fan-out elevado sugere que a complexidade geral da função qualquer é alta, podendo causar complexidade da lógica de controle para chamadas da função.
Comprimento de código	Quanto maior o tamanho do código, mais complexo e propenso a erros.
Complexidade ciclomática	Complexidade do controle e compreensão do programa.
Tamanho dos identificadores	Quanto maiores os nomes dos identificadores, mais significativos são. Com isso mais compreensível, é o programa.
Profundidade do aninhamento condicional	Quanto mais aninhamentos de condicionais, mais propenso a erros o código.
Índice Fog	Quanto mais palavras uma documentação contém, mais difícil de compreender o código.

Fonte: Elaborado com base em Sommerville, 2018.

Em geral, as métricas dinâmicas auxiliam na avaliação da eficiência e na confiabilidade do software. Já as métricas estáticas avaliam a complexidade, a compreensibilidade e a manutenibilidade, tanto do software quanto de seus componentes. Além das categorias estáticas e dinâmicas, existem métricas voltadas especificamente para códigos orientados a objetos e componentes, entre outras.

TEMA 5 – CONFIABILIDADE DE SOFTWARE

A garantia da confiabilidade de software tem relação com características apresentadas na ISO 9126. Esse modelo de qualidade faz referência aos requisitos do software, agrupando características como funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade.

Essa discussão está fundamentada na característica de confiabilidade do software. Tais características trazem para si a responsabilidade de garantir ausência de falhas, recuperação de dados após uma possível ocorrência de falhas, emissão de mensagens após uma possível falha, manutenção de padrão de funcionamento após uma possível falha, bem como eficácia no

atendimento e o suporte ao software. Tudo isso está relacionado ao desempenho do software e à sua credibilidade com os usuários.

Desde o surgimento da norma ISO 9126, muitos esforços foram aplicados para que a confiabilidade aumentasse com os usuários, trazendo um novo conceito dentro do processo de desenvolvimento de software, o Site Reliability Engineering (SRE). Esse termo foi cunhado pelo Google, em resposta a situações na empresa que colocavam em evidência a confiabilidade do software.

O SRE veio com a missão de proteger, fornecer e melhorar todas as aplicações para que se tornassem escaláveis, confiáveis e eficientes. Junto aos princípios de SRE, surgem elementos para impulsionar a questão da confiabilidade, como maior credibilidade em testes aliados ao CI/CD (Continuous Integration/Continuous Delivery). O CI/CD é um método que permite que a entrega de aplicações seja automatizada, integrada e contínua, ainda com garantia de qualidade.

A área de SRE estabeleceu uma pirâmide com categorias que vão das mais básicas às mais avançadas. O monitoramento é a base da pirâmide, pois estabelece o primeiro momento dos níveis de serviço que se encaixam para metrificar a aplicação. Quais os principais itens a serem monitorados? Latência, tráfego, erros e I/O. O monitoramento pode ser feito via alertas, tickets ou logs.

Uma vez estabelecido o monitoramento, podemos passar para a definição dos comportamentos esperados pela aplicação. Tais comportamentos são medidos pelas métricas SLI (Service Level Indicator), SLO (Service Level Objective) e SLA (Service Level Agreements). O SLI é uma métrica de negócio capaz, por exemplo, estabelecer quantas vendas foram definidas ou quantos acessos um site obteve. A SLO é uma meta colocada no software com base nos indicadores coletas em uma visão técnica, como em falhas diante de *requests*, disponibilidade de API, ou ainda média de tempo de resposta do software. Finalmente, o SLA é um acordo de nível de serviço definido pelos desenvolvedores de software para garantir que a disponibilidade do serviço não incorra em punições e multas judiciais.

Quando falamos em confiabilidade de software, um dos princípios básicos do SRE é um plano sólido para quaisquer problemas que ocorrem em relação a arquitetura, servidor, performance, plano de retorno, monitoramento, segurança, automação e escalabilidade.

A confiabilidade do software está intimamente relacionada ao momento de deploy de uma nova versão ou release do software, sendo geralmente acompanhada por um profissional DevOps, que pode automatizar muitas das atividades com ferramentas como DataDog, Cloudwatch ou Prometheus.

Normalmente, as interrupções de uso de um software se devem a mudanças em sua versão. A SRE tenta melhorar as suas práticas na administração de serviços que garantem troca de versão ou release de forma automatizada, monitorada e segura.

FINALIZANDO

Garantir a qualidade de software e de seu processo de desenvolvimento não é uma tarefa fácil. Um bom gerenciamento da qualidade de software preocupa-se em garantir que o número de defeitos seja baixo, de acordo com padrões de manutenção, confiabilidade, portabilidade e tantas outras características mensuráveis.

Percebemos também que os padrões de software nos auxiliam na garantia da qualidade, pois propõem melhores práticas e padrões, que tornam a construção do código mais limpa e eficiente.

Outro detalhe importante é que não é apenas o código que demanda atenção, mas também o gerenciamento de todas as mudanças durante o processo de desenvolvimento. É muito comum entregar um resultado diferente daquilo que foi elicitado nos requisitos, mesmo que com mudanças constantes.

Uma prática que vem crescendo dentro do desenvolvimento de software ágil é o que chamamos de “revisões em pares”, feitas sistematicamente sobre o código. A própria equipe que encarregada pelo desenvolvimento analisa em detalhes possíveis erros e omissões, discutindo e revisando o código.

Às vezes, pensamos em estruturas grandes, com pessoal responsável pela garantia da qualidade, mas na maioria das vezes não é assim. Há um número grande de pequenas empresas, em especial startups, que precisam garantir a qualidade do produto com poucas pessoas. Esses profissionais, além de excelentes programadores, precisam treinar o seu olhar de avaliação e inspeção de código para garantir que o código final esteja de acordo com o nível de satisfação desejado.

Finalmente, se a empresa já tem o ideal, que é uma equipe focada em garantia de qualidade, as medições que coletamos quantitativamente, sobre erros e falhas, podem servir de concepção para métricas que auxiliam na inferência sobre a qualidade de nosso produto e de nosso processo.

O caminho é longo, o caminho é cheio de detalhes, mas podemos começar com pequenos passos. A qualidade começa quando pensamos e refletimos sobre o que é possível adotar para garantir que a experiência do nosso usuário seja incrível.

REFERÊNCIAS

BOEHM, B. W.; BROWN, J. R.; LIPOW, M. Quantitative Evaluation of Software Quality. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING. **Proceedings**... Los Alamitos: IEEE Computer Society Press, 1976.

IEE COMPUTER SOCIETY. **Software Engineering Body of Knowledge (SWEBOK)**. 2022. Disponível em: <https://www.computer.org/education/bodies-of-knowledge/software-engineering>. Acesso em: 18 mar. 2022.

PRESSMAN, R. S. **Engenharia de software: uma abordagem profissional**. 7. ed. Porto Alegre: AMGH, 2011.

SOMMERVILLE, I. **Engenharia de software**. São Paulo. Pearson Education do Brasil, 2018.