QUALIDADE DE SOFTWARE

Aline Zanin

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Qualidade de *software*

Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Conceituar qualidade de *software*.
- Identificar os benefícios da qualidade de *software*.
- Aplicar os conceitos de qualidade.

Introdução

Ao adquirir um produto ou serviço, o cliente deseja que esse produto seja entregue em pleno funcionamento, de acordo com aquilo que foi solicitado e livre de falhas. No contexto de *software,* essa realidade é exatamente a mesma: quando o cliente solicita um *software*, cria-se uma expectativa e, por isso, faz-se necessário garantir que o *software* atenda a ela. O produto deve ser entregue em funcionamento e deve atender aos requisitos solicitados pelo cliente.

O responsável pela garantia dessa entrega eficaz é o setor de qualidade de *software*. A qualidade de *software* atua em dois segmentos distintos: qualidade de produto e qualidade de processo. Neste capítulo, você vai estudar os conceitos fundamentais desses dois segmentos, bem como conhecer os benefícios da qualidade de *software* e como aplicá-la nos projetos de desenvolvimento de sistemas.

Conceitos

A palavra qualidade parece ter um significado bastante óbvio, contudo, trata-se de um elemento complexo e de difícil mensuração, dado que a qualidade é relativa e pode assumir diversos valores, de acordo com a pessoa que a observa. Nesse sentido, a qualidade do *software* pode ser medida de acordo com o quanto ele está em conformidade com o que o cliente solicitou. Ou seja, mensura-se se o *software* está em conformidade com os requisitos funcionais

e não funcionais especificados explicitamente pelo cliente, conforme leciona Basu (2015).

Exemplo

Façamos uma analogia: suponha que você tenha preferência por chocolate do tipo meio amargo, e seu irmão tenha preferência por chocolate ao leite; ao chegar em casa, se seu pai trouxer um chocolate ao leite, muito provavelmente seu irmão dirá que esse chocolate é de alta qualidade, enquanto você dirá que o chocolate não tem qualidade, porque não atende ao seu gosto, à sua expectativa.

Embora a qualidade do *software* seja uma prática recente, que ganhou evidência com o advento da tecnologia, a qualidade, no geral, é uma preocupação bem antiga. Existem registros de que há mais de quatro mil anos os egípcios estabeleceram um padrão de medida para realizar seus trabalhos de forma apurada, chamado de cúbito, que equivalia ao tamanho do braço do faraó reinante, conforme Koscianski e Soares (2007).

Entretanto, mesmo a qualidade sendo um conceito tão antigo, os projetos de *software* contam com vários desafios para entregar o *software* em perfeito funcionamento, devido a uma série de fatores — em especial, a complexidade. Isso porque construir um sistema envolve diversas habilidades, como comunicação e interpretação, para conseguir entender o que o cliente deseja, além de habilidades específicas para as etapas de programação, análise da qualidade, entre outras, que são de grande complexidade.

Dada a multidisciplinariedade envolvida no processo de desenvolvimento de *software* e a complexidade de todo o processo, o segmento de qualidade se divide em duas áreas relacionadas, porém distintas: qualidade de processos e qualidade de produtos. A área de qualidade de processos trata da organização sistemática dos processos da empresa, visando ao melhor andamento dos projetos de desenvolvimento de sistemas, otimizando o tempo, tornando os processos repetitivos e evitando problemas em situações críticas para os projetos, por exemplo: estimativa, custo, entrada e saída de recursos humanos.

Ao buscar garantir a qualidade de um *software*, estamos diante do desafio de estabelecer uma cultura de não tolerância a erros, por meio de processos que objetivem inibir ou impedir falhas, segundo Bartié (2002). É a área de qualidade de processos que se responsabiliza pela definição da metodologia ou

do ciclo de vida de *software* que a equipe utilizará, podendo optar, por exemplo, por trabalhar com métodos ágeis ou métodos tradicionais. Independentemente da metodologia escolhida, o importante é que um processo seja seguido, para que a equipe tenha um padrão para se basear. Isso melhora a comunicação e influencia na qualidade dos produtos desenvolvidos pela equipe.

A área de qualidade de produtos tem por objetivo garantir a qualidade do produto tecnológico gerado durante o ciclo de desenvolvimento. Para esse fim, são realizadas atividades com o objetivo de estressar as funcionalidades do sistema, identificando o comportamento dele nesse contexto. Essas atividades são chamadas de testes de *software*. Essa área possui algumas divisões, sendo a mais importante a subdivisão entre testes que fazem uso do código-fonte do programa, chamados de caixa branca, e testes que não fazem uso do código-fonte do programa, chamados de caixa preta. Além disso, os testes podem ser feitos de forma manual ou automatizada e fazendo uso das mais diversas técnicas, conforme Bartié (2002).

A eficiência de um processo de testes é afetada diretamente por alguns fatores, que devem ser considerados para evitar problemas nas organizações, aponta Bartié (2002): falta de planejamento das atividades de testes, ausência de testes que validem funcionalidades antigas e ausência de processos de automação de testes.

Uma terminologia bastante importante e comum na área de testes de qualidade de *software* é o conceito de *bug*, que abrange erros, defeitos e falhas. Por curiosidade, o termo *bug* corresponde à palavra inseto em inglês e começou a ser utilizado justamente quando um inseto causou uma falha em um equipamento. É importante entender que os termos erro, defeito e falha se referem a coisas distintas. Defeito é um comportamento inesperado de um produto. O defeito está em uma parte do produto e, em geral, refere-se a uma funcionalidade que está implementado no código de maneira incorreta. Erro é aquilo que foi cometido pelo programador e que gerou um código defeituoso, enquanto a falha se dá quando o programa defeituoso é executado e interfere no funcionamento do sistema para o usuário final. Falhas também podem ocorrer por fatores externos ao programa, como corrupção de bases de dados ou invasões de memória por outros programas, conforme Koscianski e Soares (2007).

Considere, por exemplo, o código a seguir, conforme Koscianski e Soares (2007):

```
a = input ();
c = b/a;
d = a ? b/a : 0;
```

A linha de código que calcula o valor para a variável c pode apresentar um problema, dado que não é feita uma verificação para validar se o valor de a é 0. Dessa forma, pode acontecer um erro ao tentar realizar uma divisão por zero. O comportamento anormal do programa, que provavelmente gera um *bug* ou uma interrupção da execução, é provocado pela divisão b/a. Em um primeiro momento, podemos dizer que essa linha de código é defeituosa.

Existe, entretanto, outra hipótese: o defeito pode estar na rotina input (). Imagine que a especificação dessa rotina estabeleça que ela não deve jamais retornar um valor nulo. Nesse caso, o erro foi cometido pelo programador responsável por essa rotina. Essa segunda hipótese é bastante razoável, pois, para a maioria dos programas, não é recomendado preceder cada operação de divisão com um teste if. Nesse caso, um erro cometido pelo programador na rotina input fez com que o programa apresentasse um defeito ao executar a divisão de a por b.

Benefícios

Inúmeros são os benefícios que as empresas podem ter ao demandar uma atenção especial para a área de qualidade de *software*. Os benefícios vão muito além de valores financeiros, podendo estar relacionados inclusive com evitar transtornos legais ou preservar vidas.

Exemplo

Empresas que desenvolvem sistemas de semáforos têm uma responsabilidade muito grande, uma vez que um erro de programação pode causar um grande acidente de trânsito ou, na melhor das hipóteses, um transtorno no trânsito.

É por isso que o controle de qualidade é totalmente importante e benéfico nos dias atuais. A qualidade de *software* possui diversos benefícios que ajudam os usuários a não sofrerem com falhas de *software* e as empresas a oferecerem produtos melhores, impedindo até mesmo grandes catástrofes. Vamos verificar alguns dos benefícios da qualidade de *software*, com base em Basu (2015):

1. **Economiza dinheiro —** quanto dinheiro um projeto de *software* defeituoso lhe custa? Custa usuários e clientes. E é bem sabido que, quanto mais tempo leva para um *bug* ser detectado em seu *software*, mais difícil e caro é consertá-lo. Ao empregar testes de controle de qualidade durante o processo de desenvolvimento do *software*, você economizará tempo e dinheiro após a implantação.
2. **Impede emergências corporativas catastróficas —** com o *software* corporativo, as apostas são ainda maiores. *Bugs* no *software* corporativo podem levar a blecautes do sistema, falta de dados e falhas de comunicação. Se você estiver empregando *software* em toda a empresa ou lidando com informações confidenciais, é melhor ter certeza de que o *software* funcionará exatamente como ele precisa funcionar. Não há margem para erro.
3. **Inspira a confiança do cliente —** ao tornar o teste de *software* de controle de qualidade uma prioridade clara para o desenvolvimento de *software*, você está enviando uma mensagem para seus clientes de que deseja que o *software* deles seja o mais bem-sucedido possível. Isso é extremamente importante quando você busca oferecer qualidade e criar relacionamentos de longo prazo.
4. **Mantém o nível de experiência do usuário elevado —** está se tornando cada vez mais claro hoje em dia que a experiência do usuário pode quebrar ou impulsionar um negócio. Se o *software* estiver com problemas ou estiver lento, isso impedirá a experiência do usuário com o produto. Má experiência do usuário resulta em insatisfação e frustração. A boa experiência do usuário, que você obtém quando testa meticulosamente um produto de *software*, resulta em um usuário satisfeito — que tem muito mais probabilidade de recomendar o produto e sua empresa a outras pessoas.
5. **Traz mais lucro —** se você está criando um *software* para comercializar ou vender, investir em qualidade de *software* significa que você pode vender seu produto a uma taxa maior. Não há nada pior do que um usuário irritado que pagou por um produto que não funciona.
6. **Aumenta a satisfação do cliente —** relacionado ao ponto anterior, esse benefício está focado na reputação que a satisfação do cliente traz à sua empresa, não apenas no lucro. Ao oferecer um *software* de qualidade, que funciona quando e como você quer que ele funcione, você estará aumentando sua reputação, produzindo clientes satisfeitos. Não sobrecarregue a paciência dos seus clientes com um *software* defeituoso,

que você precisa consertar constantemente. Dê-lhes qualidade desde o início, e eles lhe recompensarão com lealdade.

7. **Promove organização, produtividade e eficiência —** o que você menos deseja é o caos de um *software* defeituoso, uma comunicação frenética e correções apressadas. Ser organizado com o teste de controle de qualidade, desde o início de sua estratégia de desenvolvimento, permitirá que você trabalhe em paz e seja mais produtivo com seu tempo. Ao utilizar metodologias ágeis, nas quais os desenvolvedores de *software* criam e entregam pequenos trechos de um produto em uma linha do tempo clara, você pode começar a testar o *software* à medida que ele é criado, em vez de sempre esperar até o final.

Existe uma regra que vigora desde os princípios do teste de *software*, chamada regra 10 de Myers. Essa regra estipula que o custo de encontrar um defeito no sistema aumenta 10 vezes a cada etapa do processo em que esse erro avançar, conforme aponta Myers (1979). Essa regra é mais aplicada para o contexto de projetos com ciclo de vida em cascata, em que as etapas são executadas sequencialmente, dado que, em equipes ágeis, não existe essa divisão exata de etapas. A Figura 1 ilustra essa regra.

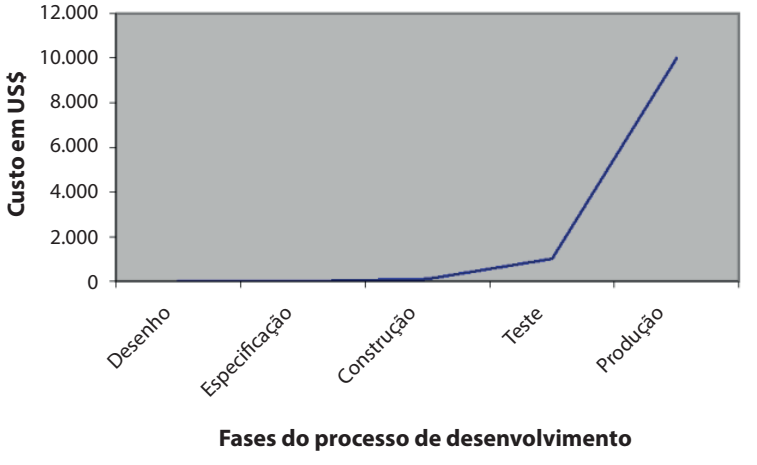

Figura 1. Regra 10 de Myers.
Fonte: MPT… (2010, p. 8).

Aplicando a qualidade de *software*

Conforme vimos anteriormente, a área de qualidade de *software* pode ser dividida em qualidade de produtos e qualidade de processos e é uma importante área no desenvolvimento de sistemas. Para a garantia da qualidade de processos, as empresas podem buscar certificações. Essas certificações fazem com a empresa seja, de certa forma, obrigada a seguir um processo que implemente a qualidade de processos de forma prática. As principais certificações são CMMI e MPS.BR.

O **CMMI**, sigla para *Capability Maturity Model Integration*, é um modelo de maturidade utilizado para medir a maturidade dos processos de uma empresa. Ele serve também como um guia para a melhoria dos processos das organizações. Esse modelo classifica as empresas em cinco níveis, conforme Koscianski e Soares (2007), sendo eles:

- Inicial: processos caóticos; em geral, não existe um ambiente propício para o desenvolvimento de *software*.
- Gerenciado: nesse nível, já é perceptível uma melhoria em relação ao nível 1. Aqui já existem requisitos gerenciados e processos planejados, medidos e controlados.
- Definido: nesse nível, a maturidade da empresa já está em um nível considerável. Aqui os processos são caracterizados e entendidos.
- Gerenciado quantitativamente: nesse nível, os processos são controlados usando métodos estatísticos e outras técnicas quantitativas.
- Otimizado: nesse nível, os processos são continuamente melhorados com base em análises feitas pela equipe.

O **MPS.BR** — sigla para Melhoria do Processo de *Software* Brasileiro — também é um modelo de maturidade, similar ao CMMI e com o mesmo objetivo. O MPS.BR foi proposto no Brasil e é utilizado por empresas brasi-leiras. Esse modelo considera sete níveis, sendo eles:

- A: processo em otimização.
- B: processo gerenciado quantitativamente.
- C: processo definido.
- D: processo largamente definido.
- E: processo parcialmente definido.
- F: processo gerenciado.

- G: processo parcialmente gerenciado.

Já para a qualidade de produto, diversas são as técnicas que podem ser empregadas. A área de testes de *software* é bastante abrangente e completa, sendo sempre recomendado que o trabalho de testes seja feito em conjunto com o desenvolvedor e durante o desenvolvimento do sistema, isto é, sem esperar que o sistema seja concluído para, então, testar. Vamos falar aqui de alguns tipos de testes e como aplicá-los, com base em Graham et al. (2008).

Teste unitário — é um tipo de teste realizado diretamente no código fonte do programa em teste. Nesse tipo de teste, são criadas funcionalidades de testes que validam o comportamento dos métodos ou funções implementadas pelo programador. No teste unitário, são identificados os primeiros erros e, quando obtido sucesso nos testes (ou seja, quando não forem localizadas falhas), as funcionalidades principais do programa devem estar funcionando. Um exemplo de técnica para teste unitário é o *TDD — test driven development —*, que tem por objetivo o desenvolvedor criar primeiro o código de teste, para depois criar a funcionalidade e, então, executar o teste que valida essa funcionalidade.

Teste funcional — é um tipo de teste realizado após o desenvolvimento do sistema ou de um módulo do sistema. O teste funcional pode ser automatizado ou manual. No caso de teste automatizado, são utilizados ***scripts* de teste**, que simulam o comportamento de um usuário no sistema, por exemplo, clicando em botões e inserindo valores em campos, e verificam a resposta do sistema para esses comportamentos. Para testes automatizados, esses *scripts* são executados automaticamente, e os defeitos localizados são sinalizados também de forma automática. No caso de testes manuais, os *scripts* são substituídos por casos de teste. Caso de teste é um documento em que é descrita a ação do usuário e a resposta esperada do sistema para ela. Nesse caso, a execução do caso de teste é feita manualmente, e o registro dos defeitos também.

Teste de aceitação — esse teste é realizado junto com o cliente e visa a verificar se o sistema que foi ou está sendo construído é realmente o que foi solicitado.

Teste exploratório — esse tipo de teste é muito comum, mais rápido de ser realizado e se torna uma alternativa para projetos com cronograma apertado. Nesse tipo de teste, o testador usa a sua experiência para navegar no sistema visando a localizar erros. A principal vantagem desse tipo de testes é localizar defeitos que não seriam localizados nos demais testes; por exemplo, casos em que o usuário tem um comportamento incomum, como clicar duas vezes em um botão de salvar. Para organizar esses testes e garantir que as funcionalidades principais sejam testadas, é comum criar um *checklist*, que

nada mais é do que uma lista que registra os principais pontos que devem ser testados nas telas do sistema.

Referências

BARTIE, A. *Garantia da qualidade de software*. Rio de Janeiro: Elsevier, 2002.

BASU, A. *Software quality assurance, testing and metrics*. India: PHI Learning, 2015.

GRAHAM, D. et al. *Foundations of software testing*: ISTQB certification. United Kingdom: Cengage Learning EMEA, 2008.

KOSCIANSKI, A.; SOARES, M. *Qualidade de software*: aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software. 2. ed. São Paulo: Novatec, 2007.

MPT – Melhoria de processo de teste brasileiro. 2010. Disponível em: <http://www.mpt.org.br/wp-content/uploads/2010/12/MPT_BR_Nivel_1_v_2.2.pdf>. Acesso em: 23 jan. 2019.

MYERS, G. J. *Art of software testing*. New York: John Wiley & Sons, 1979.

Leitura recomendada

LEWIS, W. E. *Software testing and continuous quality improvement*. Boca Raton: CRC Press, 2016.

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.

Conteúdo:

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS