TEXTO DE APOIO

AULA 1

Teste de Software

Professor Calebe de Paula Bianchini

EaD | Universidade Presbiteriana Mackenzie

Sumário

TEXTO DE APOIO: QUALIDADE DE SOFTWARE 3

REFERÊNCIAS 11

TESTE DE SOFTWARE

QUALIDADE DE SOFTWARE

No final da década de 1960, em uma mistura de reuniões entre participantes da OTAN e de uma conferência, foi assumido que existia uma Crise do Software. Na verdade, essa crise já existia bem antes daquele momento, mas foi somente naquela multi-conferência que autoridades, profissionais e pesquisadores assumiram sua existência. E o que significava a Crise do Software? De forma muito resumida, todos reconheceram que o projeto, a construção, a implantação e a operação de um software não podiam ser simplificadas em somente uma atividade de programação. Essa verdade estava estampada nos relatórios das organizações presentes naquele encontro: projetos com grandes atrasos (normalmente, acima de um ano), projetos com estouro do orçamento (aqui, estamos falando de milhões de dólares), software com mal funcionamento (os mais diversos possíveis!), projetos de longa data cancelados, dentre outros problemas que foram comentados.

> "Se você quiser conhecer mais sobre essa história, veja o relato pessoal do Prof. E. W. Dijkstra em sua página <https://www.cs.utexas.edu/users/EWD/transcriptions/EWD03xx/EWD340.html>. Ele conta um pouco sobre essa época e dos problemas que enfrentava."

Daquele momento em diante, surgiram movimentos para minimizar os impactos de um software ruim e mal feito. Isso ficou conhecido como Engenharia de Software.

Um esforço constante tem sido feito para que a engenharia de software seja um sucesso. Por um lado, para que um software não seja ruim, ele precisa ser pensado como um produto que atende às necessidades do cliente. Por outro, o processo de construção do software e suas fases devem ser feitos da melhor forma possível, visando construir o produto de software de maneira otimizada, sem desperdício de tempo, dinheiro, ou qualquer outro recurso necessário para sua finalização.

É por isso que você deve conhecer a Engenharia de Software: modelos de processos de desenvolvimento de software, as fases de concepção, modelagem e projeto, construção, teste, implantação, operação, gerenciamento, dentre outras disciplinas que auxiliam uma correta construção do software.

Apesar disso, ainda fica faltando uma disciplina muito importante para que tudo o que foi feito tenha correlação e colabore para a boa construção do produto:

Qualidade de Software. Ela é uma disciplina reconhecidamente importante e integra as demais disciplinas que todos os profissionais da área devem conhecer.

É fácil você perceber que os conceitos de qualidade fazem parte do nosso dia a dia. Na compra de qualquer produto no mercado, nós aplicamos esses conceitos empiricamente: sabor, textura, preço, tamanho, volume, dentre tantas outras decisões que nos auxiliam durante uma compra. Apesar disso, definir qualidade de software, de forma geral, é um desafio que nem os autores mais tradicionais da área tentam fazer. Porém, todos eles concordam: é necessário definir um conceito de qualidade quanto estamos tratando de software.

Outro consenso que também existe na área diz respeito a atender as expectativas do cliente que fará uso do software. Ou seja, se você desenvolver um software que não traz satisfação, não agrega valor ao negócio dele, não transforma seu mundo, tem grandes chances desse software agradar somente a você. Certamente, as expectativas do cliente estão além da satisfação de uso do software, englobando também questões relacionadas ao valor do software e ao prazo de sua construção.

" **Curiosidade:** será que podemos definir a satisfação do cliente como sendo:

expectativa = uso + preço + prazo ?

Essa tríade é chamada de Triangulo de Ferro da qualidade de software e pode ser vista como mostra a Figura 1."

Figura 1 – Triângulo de ferro da qualidade de software

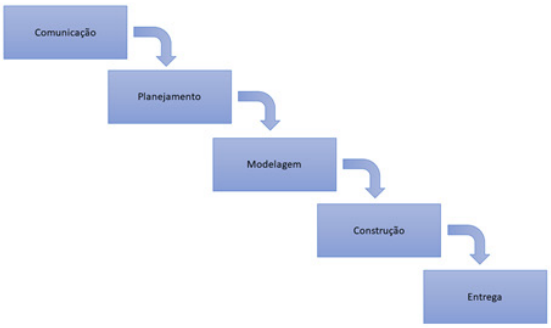
Ou seja, na perspectiva de qualidade de software, se o cliente não estiver satisfeito com o software que está utilizando, nada mais importa. Por outro lado, se o software, de forma geral, fornece um benefício substancial, o cliente se torna mais tolerante a possíveis problemas que o software possua, como falta de desempenho ou falhas ocasionais de confiabilidade.

A escolha de um bom processo de desenvolvimento de software é um dos passos importantes para o sucesso da construção de um software. O processo interfere diretamente em dois dos fatores relacionados à expectativa do usuário: o prazo e o custo. Os modelos de processo existem para ajudar o profissional de engenharia de software a tomar uma decisão sobre qual processo de desenvolvimento de software deve ser escolhido, dando ênfase ao tipo do software solicitado pelo cliente. Cada modelo tem suas próprias características, impactando, portanto, na qualidade do produto gerado.

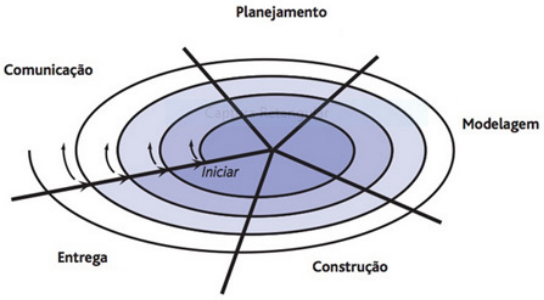
?

"Você lembra dos modelos de processos de desenvolvimento de software estudados anteriormente? Nós separamos alguns que valem a recordação durante seus estudos, conforme mostra a Figura 2.

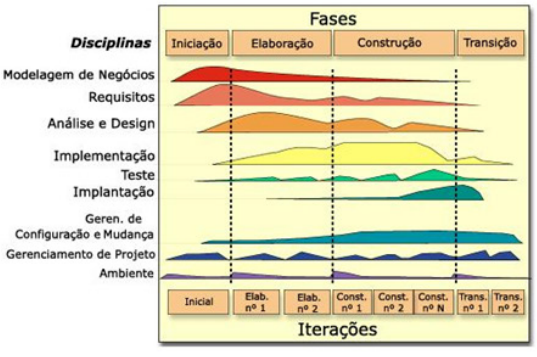
Figura 2 – Modelos de processo de desenvolvimento de software



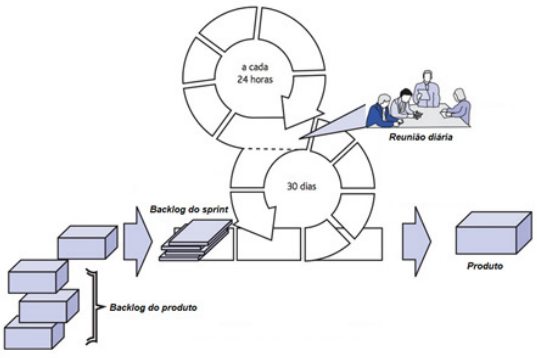
(a) Modelo em cascata



(a) Modelo espiral



(a) Processo unificado



(a) Método ágil Scrum

Uma vez escolhido o processo, ainda são necessários esforços de qualidade para que ele seja realizado e conduzido com sucesso. Isso significa, mais uma vez, que toda a equipe e, muitas vezes, o cliente estão envolvidos ao longo de todo o processo de desenvolvimento de software.

Para vencer os desafios pertinente a este processo, as empresas estudam e buscam certificações de qualidade de processo. De forma geral, o objetivo principal das certificações é garantir que o processo estabelecido seja seguido com o máximo de qualidade. Por isso, muitas dessas certificações oferecem um conjunto de guias ou áreas-chave que direcionam o processo e, por consequência, são utilizados também como medida para o selo de certificação (mediante avaliação). As certificações da ISO-9000 são os exemplos mais conhecidos quando se trata de certificação de processo que visam atender a satisfação do cliente.

Porém, devido à natureza abstrata do software (ou seja, ele não é tangível, e muda bastante de cliente para cliente), as empresas precisam adaptar os processos novamente, devido às novas características de cada software. Uma vez que o processo precisa ser ajustado, outras normas e guias de certificação podem se fazer necessárias. Algumas normas foram criadas para auxiliar as empresas na escolha de quais fases devem fazer parte do processo de desenvolvimento de software. É o caso da ISO-12207 que fornece uma lista de fases que podem ser empregadas para conceber, controlar e melhorar o processo e o ciclo de vida do software.

Quanto aos guias de certificações, novamente eles auxiliam na tomada de decisão quanto à mudança do processo e, como consequência, tendem a facilitar a emissão do selo de certificação (sempre mediante avaliação). Nesse caso, a certificação não só avaliará a empresa quanto à capacidade de seguir o processo escolhido, mas também como ela se adapta às mudanças necessárias provenientes da natureza do software. Nesse caso, a certificação avalia a maturidade da empresa.

As certificações de maturidades mais conhecidas são CMMI e MPS.BR. Elas dividem as certificações em níveis, sendo que a primeira tem cinco níveis e a segunda tem sete níveis. Os níveis mais baixos da certificação dão ênfase para a gerência do processo de desenvolvimento de software, ou seja, se a empresa consegue realmente decidir sobre a adoção de um processo e segui-lo ao longo da construção do software. Os níveis mais altos dessas certificações demonstram que as empresas conseguem, conscientemente, adaptar-se às questões da natureza abstrata do software, bem como tratar outros elementos surpresa decorrentes dos desafios de um processo de desenvolvimento de software.

Adicionalmente às questões relacionadas ao processo, é necessário também

avaliar a qualidade do produto de software. Ou seja, diversos aspectos do software precisam ser avaliados e, agora, não somente as questões relacionadas às necessidades do cliente. É comum fazer uma divisão desses aspectos, observando seu impacto no software, sendo eles (i) internos e (ii) externos.

Os aspectos internos de um software consideram as estruturas utilizadas na construção do produto: algoritmos, estrutura de dados, modularização, portabilidade, manutenibilidade, testabilidade, coesão, acoplamento, dentre outros fatores que podem ser utilizados para observar internamente um produto de software. A avaliação desses aspectos e fatores é realizada pela própria empresa que desenvolve o produto.

Por outro lado, existem os aspectos externos, como facilidade de uso, desempenho, facilidade de instalação, dentre outros, que têm como principal fonte avaliadora o cliente. Mesmo assim, é possível observar esses aspectos ao longo do desenvolvimento de software que, consequentemente, foram avaliadas antes mesmo do cliente.

Alguns desses fatores são bem conhecidos e formaram a ISO-9126 (que foi reaproveitada na ISO-25000):

- Funcionalidade: indica a aderência do software em relação às necessidades declaradas.

- Confiabilidade: indica a quantidade de tempo em que o software fica disponível para uso.

- Usabilidade: indica a facilidade de uso do software.

- Eficiência: indica o quão otimizado está o software com relação ao uso dos recursos do sistema.

- Facilidade de manutenção: indica a facilidade com a qual uma correção pode ser realizada no software.

- Portabilidade: indica a facilidade com a qual um software pode ser transposto de um ambiente para outro.

A partir do momento que você reconhece que qualidade de software será uma disciplina que acompanha todo o processo de desenvolvimento de software, é possível afirmar também que nenhum software é perfeito! Por isso, o software apresentará algum tipo de problema e, portanto, precisamos entender um pouco

sobre a origem desse problema.

Você também já sabe que, se um algoritmo (ou código-fonte) produzir uma resposta que não está em conformidade com o enunciado de um requisito funcional, certamente existe um bug nesse software. Apesar do termo bug estar intimamente ligado ao cotidiano do desenvolvimento de software, ele é genérico o suficiente para representar diversos tipos de problemas de um software. Por isso, ao aplicar os princípios de qualidade de software, é necessário entender o que significa esse *bug* e usar corretamente as definições.

Em qualidade de software, as definições básicas usadas para tratar os problemas de software são:

- Falha *(failure)*: é a demonstração de mau funcionamento de um software, ou seja, é o resultado visível do comportamento de um software que não está em conformidade com o que deveria realizar.

- Defeito *(fault)*: é uma imperfeição, deficiência, omissão ou inconsistência que está presente em um artefato de software, seja este um código-fonte, um modelo, uma especificação, ou qualquer outro artefato relevante do software. Um defeito precisará ser removido, pois sua execução em software (ou a construção derivada dele) levará a uma falha.

- Erro *(error)*: é uma ação humana que produz um resultado incorreto, inserindo um defeito no artefato de software.

"Você sabia que existe um documento normativo para termos e vocabulário da Engenharia de Software? Ele se chama ISO/IEC/IEEE 24765 – Systems and software engineering – Vocabulary, e pode ser acesso por meio do Portal CAPES na página da Biblioteca da Universidade Presbiteriana Mackenzie."

Partindo do pressuposto de que essas definições servem para melhorar o entendimento dos problemas do software e aumentar a comunicação e a proatividade ao longo do processo de desenvolvimento de software, cada empresa tem a liberdade de redefinir (ou aumentar) esses termos para melhor acomodar as dinâmicas dos processos escolhidos.

Mas, mesmo com uma definição própria, a empresa precisa saber a origem da falta de qualidade em um software, pois é importante para a gestão da qualidade de software. Técnicas bem conhecidas podem ser aplicadas visando reduzir a

quantidade de defeitos produzidos no software. De forma geral, a classificação do problema permite à equipe de gestão de qualidade decidir sobre:

- Prevenção de Defeitos: ao perceber que os defeitos possuem uma origem humana, pode-se adotar políticas de incentivo ao treinamento, à especialização e à educação. Dessa forma, os profissionais envolvidos na construção dos artefatos de software diminuem a quantidade de defeitos inseridos (a maioria dos defeitos inseridos são feitos de forma inconsciente).

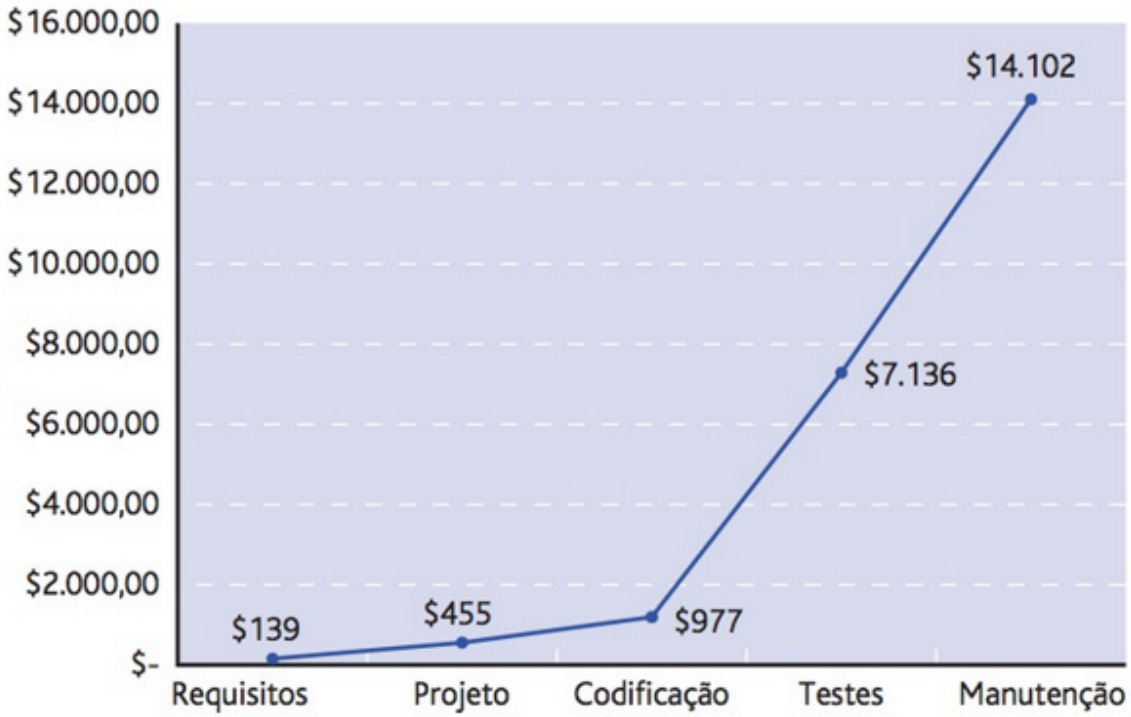
- Redução de Defeitos: sabendo que é impossível construir um software perfeito, e mesmo com o melhor treinamento, defeitos acidentais serão inseridos nos artefatos de software. Por isso, definir como eles podem ser removidos se torna uma prática muito importante. Dentre as técnicas mais conhecidas estão a (i) revisão e o (ii) teste de software.

- Contenção de Defeitos: mesmo com as técnicas de redução, o software ainda pode apresentar algumas falhas ao longo de sua execução. Por isso, entender quais são as partes críticas do software e antecipar uma solução se torna algo importante, principalmente em sistemas complexos e de tempo real. Um exemplo dessas técnicas é estabelecer mecanismos de redundância, balanceamento de carga, dentre outros mecanismos que envolvem também tolerância a defeitos.

Além disso, outras técnicas de garantia de qualidade de software podem ser adotadas para que a quantidade e o impacto dos defeitos sejam minimizados. Essas técnicas são as mais diversas possíveis, como a adoção de modelos de documentação e processo; o uso de padrões de artefatos reconhecidamente bons; o estabelecimento de revisões (e, talvez, auditorias); a prática de testes preventivos, automatizados e em diversos níveis; a coleta e análise dos problemas encontrados e corrigidos; o gerenciamento de mudanças e o estabelecimento de bons processos de evolução de software; a análise e controle de riscos; o estabelecimento de procedimentos de segurança, dentre diversas outras práticas e técnicas que fazem parte de um conjunto de atividades que podem (e devem) ser adotadas como garantia de qualidade de software.

A demora para a remoção de um defeito de um software pode aumentar os custos de sua correção. A Figura 3 mostra um relatório apresentado em 2001 detalhando os custos financeiros para a correção de defeitos nas diversas fases do processo de desenvolvimento de software. Você pode perceber, por exemplo, que o custo da remoção de um defeito na fase de requisitos é baixo e ele aumenta nas fases mais avançadas do processo.

Figura 3 – Custo de correção do defeito ao longo do processo de desenvolvimento de software



Uma boa gestão de qualidade de software, sabendo que esses custos sempre aumentam, tratam do problema com seriedade e utilizam ferramentas para monitorar, analisar e controlar os defeitos que surgem ao longo do processo. Essas ferramentas, no cenário atual, tornaram-se fundamentais para a área de qualidade de software.

REFERÊNCIAS

O texto e as imagens foram elaborados com base nas seguintes referências bibliográficas:

GONÇALVES, P. et al. *Testes de software e gerência de configuração*. Porto Alegre: SAGAH, 2019.

PRESMAN, R.; MAXIM, *B. Engenharia de Software: uma abordagem profissional*. 8. ed. Porto Alegre: AMGH, 2016.

SOMMERVILE, I. *Engenharia de Software. 10.* ed. São Paulo: Pearson, 2018.

ZANIN, A. et al. *Qualidade de Software.* Porto Alegre: SAGAH, 2018.