Introdução a Teste de Software

Herysson R. Figueiredo
herysson.figueiredo@ufn.edu.br

Introdução a teste de *software*

Nesta aula teremos a introdução da nossa disciplina “Testes de Software” e sendo uma aula introdutório trataremos da base teórica da disciplina, assim como os termos relacionados ao assunto. Queremos que você entenda a importância de se testar o software durante todo desenvolvimento e saiba o que é um teste.

Objetivos

- Estudar sobre qualidade de software;
- Entender a importância das atividades de garantia de qualidade e os envolvidos nessas atividades;
- Conhecer as atividades de verificação e validação (V&V) e entender a diferença entre elas;
- Definir teste de software e termos relacionados, como caso de teste, cenário de teste, defeito, erro, falha e engano;
- Entender a importância dos testes nos diferentes modelos de processo de desenvolvimento de software.

Qualidade de *software*

“Você conhece um software de qualidade?

“No que você se baseou para julgar a qualidade do software? O que é qualidade de software? Como será que se faz para atingir a qualidade de um software?”.

Qualidade de *software*

"Qualidade de software é a conformidade com requisitos funcionais e de desempenho explicitamente declarados, normas de desenvolvimento explicitamente documentadas e características implícitas que são esperadas em todo software desenvolvido profissionalmente"

Pressman (PRESSMAN, 2006)

Qualidade de *software*

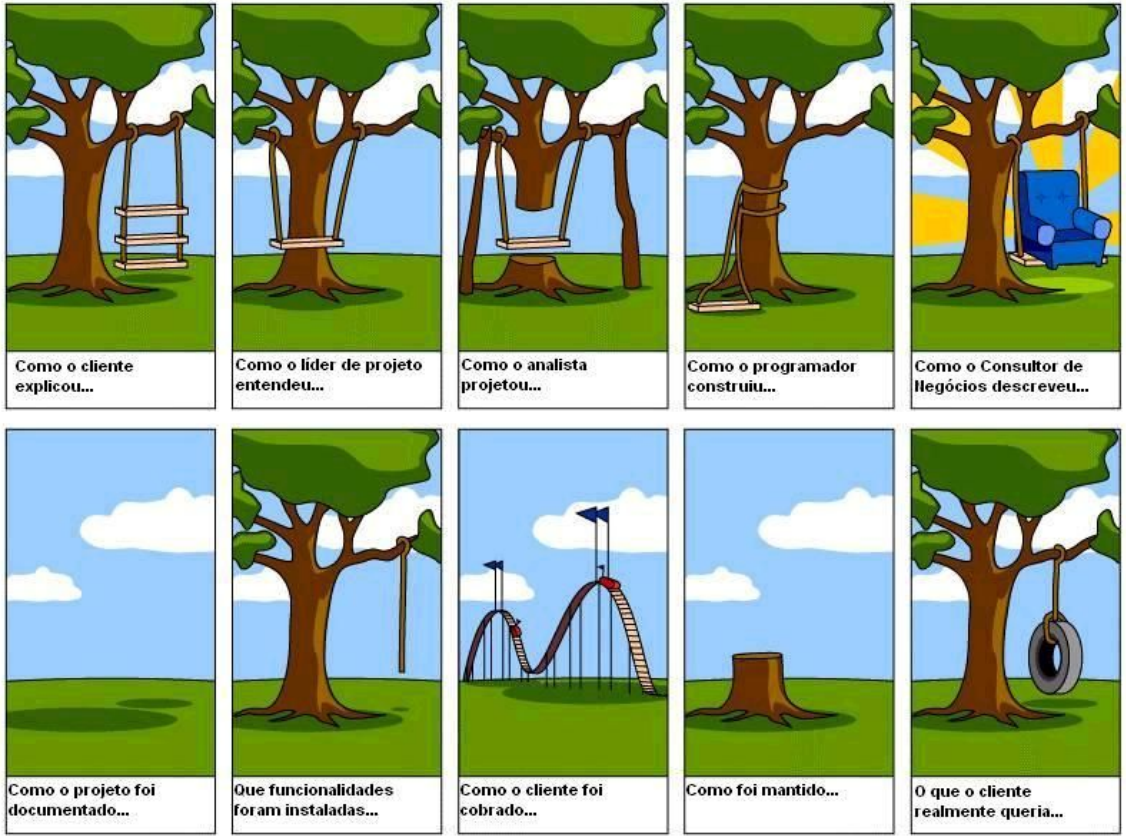
Sempre que falamos de Qualidade de *Software* devemos lembrar nas atividades de **garantia de qualidade de software – SQA (*Software Quality Assurance*)**. Essas atividades foram pensadas para fazer parte de todo processo de desenvolvimento de software e têm como objetivo garantir que requisitos de qualidade de software sejam satisfeitos pelo software

Qualidade de *software*

Requisitos de qualidade: Conjunto completo das **características** da qualidade de um processo, item, material, produto ou serviço, com valor nominal e seus respectivos limites máximo e/ou mínimo de aceitação (tolerâncias).

Requisitos de software: são descrição dos recursos e funcionalidades do sistema alvo. **Os requisitos** transmitem as expectativas dos usuários do produto de **software**.

Qualida


Como o cliente explicou...
Como o líder de projeto entendeu...
Como o analista projetou...
Como o programador construiu...
Como o Consultor de Negócios descreveu...
Como o projeto foi documentado...
Que funcionalidades foram instaladas...
Como o cliente foi cobrado...
Como foi mantido...
O que o cliente realmente queria...

Qualidade de *software*

Pensando de maneira mais ampla, a garantia de qualidade é uma atividade imprescindível para qualquer negócio que produza um produto, seja ele software ou não.

Qualidade de *software*

No contexto das atividades de **garantia de qualidade de software**, vários papéis diferentes do time de desenvolvimento possuem grande importância: engenheiros de software, gerentes de projeto, clientes (quando possível), pessoal de vendas e outros envolvidos fazem parte de um time de SQA.

SQA (*Software Quality Assurance*) = Garantia da qualidade de software.

Qualidade de *software*

O time de SQA deve olhar para o software procurando ter o ponto de vista do cliente, procurando sempre responder perguntas como:

- O desenvolvimento do software foi conduzido de acordo com padrões preestabelecidos?
- As disciplinas técnicas desempenharam seu papel de forma adequada como parte das atividades de SQA?

SQA (*Software Quality Assurance*) = Garantia da qualidade de software.

Qualidade de *software*

O time de SQA deve olhar para o software procurando ter o ponto de vista do cliente, procurando sempre responder perguntas como:

- O software satisfaz critérios de qualidade? A saber, confiabilidade, correção, disponibilidade, eficiência, escalabilidade, flexibilidade, funcionalidade, integridade, interoperabilidade, manutenibilidade, prazo de colocação no mercado, portabilidade, reutilização, segurança, testabilidade e usabilidade?

SQA (*Software Quality Assurance*) = Garantia da qualidade de software.

Qualidade de *software*

A SQA é composta por diversas tarefas associadas a duas frentes distintas – engenheiros de software, que fazem o trabalho técnico, e o time de SQA :

- Engenheiros de *software* aplicam métodos e medidas técnicas sólidas, conduzindo revisões técnicas formais e testes de *software*,
- O time de SQA deve planejar as atividades que farão parte do plano de SQA, supervisionar para garantir que as atividades estão sendo executadas de acordo com o plano, supervisionar a execução das atividades do plano de SQA, registrar como elas são conduzidas e caso necessário, relatar inconformidades.

Qualidade de *software*

“Essa distinção de tarefas ajuda a reforçar na empresa que todos, sem exceção, são responsáveis pela qualidade do produto de *software*.”

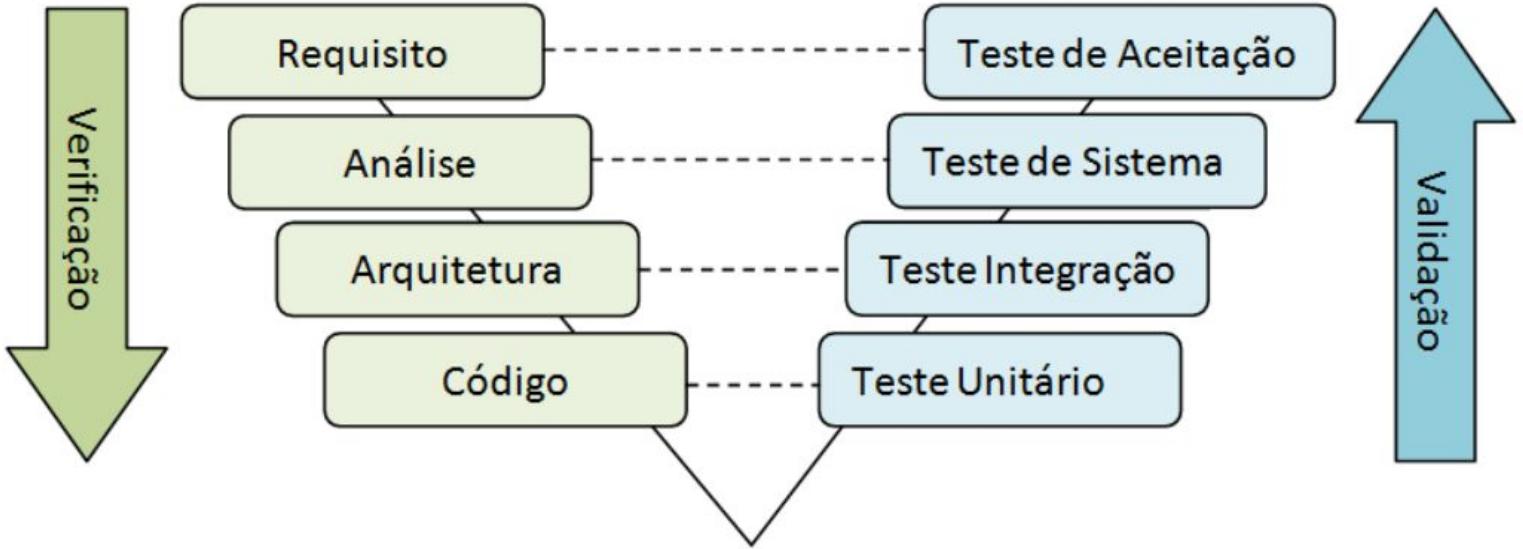
Qualidade de *software*

Do ponto de vista técnico, temos duas atividades de extrema importância para garantir a qualidade de um software: verificação e validação – conhecidas como atividades de V&V.

Qualidade de *software*

VERIFICAÇÃO	Atividades para responder à pergunta: "*Estamos construindo o produto corretamente?*";
VALIDAÇÃO	Atividades para responder à pergunta: "*Estamos construindo o produto certo?*".

Qualidade de *software*



Qualidade de *software*

Além de classificadas como verificação e/ou validação, as atividades V&V são classificadas como estáticas e dinâmicas:

- Atividades estáticas são aquelas que **não dependem** de um artefato executável para serem realizadas. Em outras palavras, não é necessário ter um código fonte para realizar atividades de V&V estáticas.
- Atividades dinâmicas são aquelas que **dependem** de um artefato executável para serem realizadas. Onde é necessário ter um código fonte executável para realizar atividades de V&V dinâmicas.

Artefato de Teste: é a especificação de um conjunto de entradas de **teste**, condições de execução e resultados esperados, identificados com a finalidade de fazer a avaliação de algum aspecto particular de um cenário.

Qualidade de *software*

As atividades de V&V são tipicamente executadas usando uma ou mais das quatro técnicas listadas a seguir:

ANÁLISE	Que é o uso de técnicas ou modelos matemáticos, simulações algoritmos ou procedimentos científicos para determinar se o produto ou artefato está em conformidade com seus requisitos.

Qualidade de *software*

As atividades de V&V são tipicamente executadas usando uma ou mais das quatro técnicas listadas a seguir:

DEMONSTRAÇÃO	É um exame visual do produto de software sendo executado em um cenário específico a fim de identificar se ele está em conformidade com os requisitos. As famosas "demos" são um exemplo dessas demonstrações.

Qualidade de *software*

As atividades de V&V são tipicamente executadas usando uma ou mais das quatro

INSPEÇÃO	É uma examinação visual (comumente envolvendo a manipulação manual dos artefatos) de um artefato não executável, novamente para identificar a conformidade do artefato com seus requisitos. Por sua característica comumente estática, Sommerville (SOMMERVILLE, 2011) salienta que técnicas de análise, como as revisões formais e inspeção, oferece garantia somente entre a correspondência de um artefato com a sua especificação, não sendo adequada para identificar a utilidade do software para o cliente.

Qualidade de *software*

As atividades de V&V são tipicamente executadas usando uma ou mais das quatro técnicas listadas a seguir:

TESTE	É a simulação de um artefato executável com entradas e pré-condições conhecidas e a comparação entre a saída atual e a saída esperada para determinar se o artefato está em conformidade com os requisitos.

Teste de *software*

Teste de Software é a técnica mais fácil de se executar, mesmo em ambientes com processo de desenvolvimento inadequado, no qual não há documentação alguma, há um executável e, mesmo que de forma não adequada, a atividade de teste pode ser aplicada.

“Teste de *software* é o último reduto no qual a qualidade do software pode ser avaliada e defeitos podem ser identificados antes de serem entregues ao usuário final.” (PRESSMAN, 2006)

Teste de *software*

"*Teste é a simulação de um artefato executável com entradas e pré-condições conhecidas e a comparação entre a saída atual e a saída esperada para determinar se o artefato está em conformidade com os requisitos*" (FIRESMITH, 2013).

"Teste é o processo de executar um sistema ou componente sob condições específicas, observando e registrando os resultados, avaliando alguns aspectos do sistema ou componente" (ISO/IEC/IEEE 24765, 2010).

"Teste é o processo de executar um programa ou sistema com a intenção de encontrar erros" (MYERS, 2004).

Teste de *software*

Teste de software – a mentalidade é: se existia algum defeito nas funcionalidades que eu executei durante os testes (ou no código-fonte executado), defeitos esses alcançados pela forma como eu usei o sistema (dados, sequência de execução etc.), esse defeito foi encontrado, reportado e corrigido.

“Uma solução para não ter problema no software é testar todas as possibilidades.”

Dessa forma, nós podemos garantir que todas as funcionalidades foram executadas de todas as formas possíveis e como consequência, todos os defeitos foram encontrados. Isso é o que chamamos de teste exaustivo.

Quanto tempo levaria para ser executado um teste exaustivo?

Teste de *software*

```
1 int exemplo_simples(int j) {
2    j = j - 1;
3    j = j / 30000;
4    return j;
5 }
```

Teste de *software*

No exemplo anterior seria impossível realizar testes por exaustão em um tempo em um tempo plausível. Além da impossibilidade de testar por exaustão, ainda temos o impacto da escolha dos dados corretos, pois um conjunto de dados pode fazer com que o defeito seja identificado, e outro conjunto de dados pode fazer com que o defeito nunca seja identificado.

Teste de *software*

```
1 int exemplo_simples(int j) {
2    j = j - 1;
3    j = j / 30000;
4    return j;
5 }
```

Teste de *software*

Considerando a impossibilidade de testar por exaustão, nós devemos aplicar os critérios de teste, que nos ajudam, entre outras coisas, a definir melhor nossos **casos de testes** e os dados de entrada a fim de identificar defeito.

Casos de teste

De acordo com a norma IEEE 829-2008 (IEEE 829-2008, 2008), **caso de teste** é um conjunto de dados de entradas, condições de execução do sistema e o resultado desenvolvido/projetado para atingir um objetivo específico, como exercitar um caminho específico em um programa ou verificar a concordância do programa com o requisito.

Casos de teste

Casos de teste é uma sequência de passos que devem ser executados no sistema, sendo que os dados de entrada e saída esperada para cada passo são especificados. Os casos de teste devem “direcionar” as ações do testador em uma determinada funcionalidade e fazer com que ele observe se o resultado que ele obtém em cada passo é o resultado esperado, de acordo com os requisitos

Casos de teste

Segundo norma (IEEE 829-2008, 2008) menciona que um **caso de teste** deve conter:

1. IDENTIFICADOR DO CASO DE TESTE	Número único de identificação do caso de teste em questão.
2. OBJETIVO DO CASO DE TESTE	Mostra de forma breve qual é a intenção de teste do caso de teste.
3. ENTRADAS	Todas as entradas que serão consideradas no caso de teste.

Casos de teste

Segundo norma (IEEE 829-2008, 2008) menciona que um **caso de teste** deve conter:

4. SAÍDAS	As saídas esperadas de acordo com as entradas indicadas na execução dos testes.
5. AMBIENTES NECESSÁRIOS OU CONDIÇÕES DE AMBIENTE	Quais ambientes de *hardware e software*, inclusive com versionamentos, que devem ser montados para o caso de teste em questão.

Casos de teste

Segundo norma (IEEE 829-2008, 2008) menciona que um **caso de teste** deve conter:

6. PROCEDIMENTOS ESPECIAIS REQUISITADOS PARA EXECUÇÃO DO CASO DE TESTE	Como o analista de teste deve proceder para a execução do caso de teste em questão. Deve especificar com detalhes o que deve ser testado e a forma que o teste deve ser executado.
7 DEPENDÊNCIA COM OUTROS CASOS DE TESTE	No caso deste caso de teste estar relacionado com outro caso de teste, então, especificar este relacionamento aqui.

Casos de teste

Pensando no exemplo, um caso de teste possível seria:

1. Caso de teste 1: entrada negativa
2. Verificar se entradas negativas são possíveis
3. (E) Digite “- 30000”
4. (S) 0,99
5. Nenhum
6. Nenhum
7. Não

Com esse caso de teste, o testador iria verificar que há um defeito, pois certamente o programa iria falhar em não retornar o resultado esperado.

Teste de *software*

```
1 int exemplo_simples(int j) {
2    j = j - 1;
3    j = j / 30000;
4    return j;
5 }
```

Casos de teste

O que fazer quando o programa falha e não retorna o resultado esperado?

Ao identificar um defeito, o testador deve reportar o defeito de forma a permitir que o time de desenvolvimento identifique facilmente como eliminar o defeito do sistema.

A forma de reportar defeitos depende muito de padrões definidos em cada empresa, ou time de desenvolvimento, e das ferramentas utilizadas para gerenciar o projeto e a atividade de teste, como por exemplo, Jira e Zephyr da Atlassian , Bugzilla ou testLink .

Reportar defeitos

Uma forma de como reportar um defeito é (ISTQB, 2016)

- **Identificador do defeito:** todo defeito deve ter um identificador único para facilitar a comunicação e localização;
- **Descrição do defeito:** uma descrição sucinta do defeito encontrado;
- **Versão do produto:** para saber em qual versão do sistema o defeito foi encontrado;
- **Passos detalhados:** descrição de passos realizados no sistema, incluindo os dados utilizados para encontrar o defeito. *Screenshots* e vídeos podem ser muito úteis também. A ideia é permitir que os desenvolvedores consigam reproduzir o defeito em ambiente de depuração;

Reportar defeitos

Uma forma de como reportar um defeito é (ISTQB, 2016)

- **Data de reporte do defeito:** para facilitar o gerenciamento;
- **Reportado por:** para saber qual o testador identificou o defeito;
- ***Status:*** aqui, diferentes nomes podem ser aplicados, mas o objetivo é permitir que todos os envolvidos saibam se o defeito já foi endereçado, analisado, corrigido, se tem algum desenvolvedor encarregado de arrumar o programa, se o testador já viu a resolução e assim por diante. Um exemplo de lista de possíveis status é: novo/aberto, designado, em verificação, resolvido, fechado, falhado, não será corrigido etc

Reportar defeitos

Uma forma de como reportar um defeito é (ISTQB, 2016)

- **Corrigido por:** Nome do desenvolvedor que corrigiu o defeito;
- **Data de encerramento:** a data que o defeito foi dado como inexistente;
- **Severidade:** para informar quão grave é o defeito no sistema, como por exemplo, bloqueia a versão, crítico, pequeno;
- **Prioridade:** para definir a prioridade em corrigir o defeito: alta, média ou baixa

Cenário de Testes

De acordo com a norma IEEE 829-2008 (IEEE 829-2008, 2008), cenário de teste é um conjunto de casos de teste relacionados, que comumente testam o mesmo componente ou a mesma funcionalidade do sistema.

Cenários de caso de teste também chamados de suítes ou ciclos de teste.

Cenário de Testes

Ao definir esse agrupamento de casos de teste, temos que pensar que eles serão extremamente úteis para testes de regressão (que são testes feitos na fase de manutenção do sistema para garantir que as novas funcionalidades não inseriram defeitos nas antigas), para revalidar mudanças nos casos de teste e também para treinar novos testadores e pessoas que darão suporte ao sistema.

Cenários de caso de teste também chamados de suítes ou ciclos de teste.

Exemplo de caso de teste - Autenticação (*login)*

1. Caso de teste 1 - *Login*
2. Verificar *login* inválido
3. Entradas (E)
4. Saídas (S)
    a. (E) No celular, clique no ícone do aplicativo.
    b. (S) O aplicativo deve abrir e exibir a tela de *login*.
    c. (E) Insira um e-mail inválido, sem o sinal de @ (professor.gmail. com) e clique em "*next*".
    d. (S) O aplicativo deve deixar a borda do campo *login* vermelho e a mensagem "Por favor, digite um *login* válido" deve ser exibida.
    e. (E) Clicar em OK na mensagem.
    f. (S) O cursor de digitação deve estar no campo de *login* e o usuário deve poder corrigir o login anterior.
5. Smartphone com o aplicativo instalado
6. Nenhum
7. Não (afinal, não preciso ter feito nenhuma ação antes para testar o login)

Engano, defeito, erro e falha

Segundo a definição da norma ISO/IEC/IEEE 24765:2010 :

ENGANO (*MISTAKE*)	Uma ação humana que produz um resultado incorreto, como uma codificação ou modelagem errada;
DEFEITO (*FAULT*)	Uma imperfeição ou deficiência em um artefato que faz com que este não esteja em conformidade com os requisitos ou especificações, sendo necessária sua correção ou substituição. Termos como "erro" e "*bug*" comumente são usados para expressar defeitos;

Defeito, erro e falha

Segundo a definição da norma ISO/IEC/IEEE 24765:2010 :

ERRO (*ERROR*)	Resultado incoerente produzido por meio de uma ação no sistema;
FALHA (*FAILURE*)	Incapacidade do *software* exercer a função para a qual foi desenvolvido.

Defeito, erro e falha

Sendo assim, engano introduz um defeito que quando é exercitado, idealmente por meio da execução de um caso de teste, produz um erro no sistema e esse erro, consequentemente, faz com que o sistema falhe.

Atividades

1. Ao seu ver, qual é a importância da qualidade de software em uma empresa?
2. Defina, com suas palavras, o que é Verificação e o que é Validação de software.
3. No contexto de teste de software, o que é um cenário de teste e para quais fins ele pode ser utilizado?

Referências

PRESSMAN, Roger S. Engenharia de Software. Mc Graw Hill, 6 ed, Porto Alegre, 2006.

BRAGA, Pedro Henrique Cacique. Teste de Software. Pearson Education do Brasil. São Paulo. 2016. Disponível na Biblioteca Virtual.

DELAMARO, Márcio; MALDONADO, José Carlos, JINO, Mario. Introdução ao teste de software. Elsevier. Rio de Janeiro. 2007.

PEZZÉ, Mauro; YOUNG, Michal. Teste e Análise de Software. Porto Alegre: Bookman, 2008

ISO/IEC/ IEEE 24765 - Systems and software engineering — Vocabulary , 2010

FIRESMITH, Doanald G. Common System and Software Testing Pitfalls: How to Prevent and Mitigate Them: Descriptions, Symptoms, Consequences, Causes, and Recommendations, Addison-Wesley Professional, 2013

MYERS, Glenford J. The Art of Software Testing. 2004