INSTITUTO FEDERAL
Santa Catarina
Câmpus Gaspar

Teste de Software

Prof. Marcos Rodrigo momo, M.Sc.
marcos.momo@ifsc.edu.br

Gaspar, maio 2021.

Roteiro

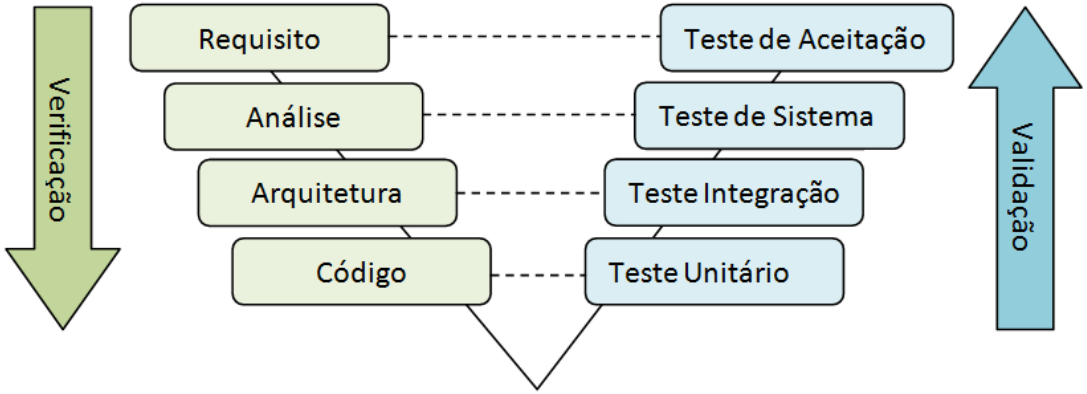
- Gravação
- Testes durante o ciclo de vida do desenvolvimento do software
- Atividades teóricas

Teste durante o ciclo de vida do software

- **Modelos de Desenvolvimento de Software**
  - **Modelo V**
  - **Modelos iterativos de desenvolvimento**
  - **Teste dentro de um modelo de ciclo de vida**
- Níveis de Teste
  - Teste de Componente (ou Unidade)
  - Teste de Integração
  - Teste de Sistema
  - Teste de Aceite
- Tipos de Teste: o alvo do teste
  - Teste de Função (Teste funcional)
  - Teste de características do produto de software (testes não funcionais)
  - Teste de estrutura/arquitetura do software (teste estrutural)
  - Teste relacionado a mudanças (teste de confirmação e regressão)
- Teste de Manutenção

Modelos de Desenvolvimento de Software

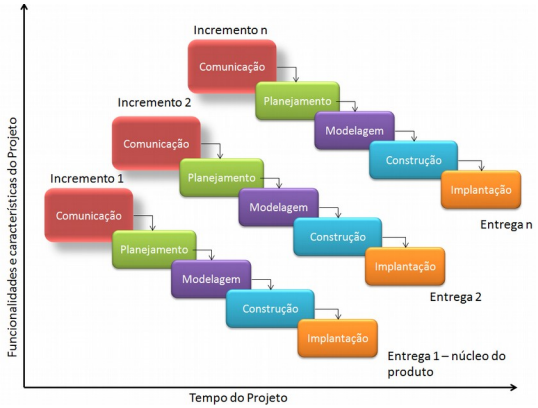
- Modelo V (fases do desenvolvimento x níveis dos testes)
  - Pode ter mais, menos ou diferentes níveis de desenvolvimento e teste, dependendo do projeto e do produto.

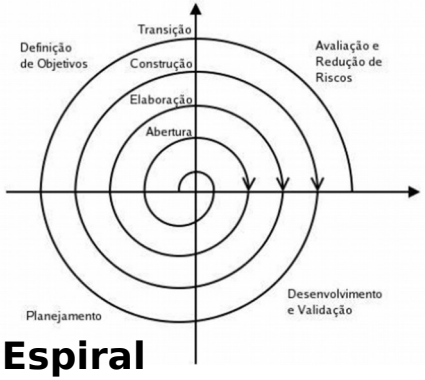


Modelos Iterativos de Desenvolvimento de Software

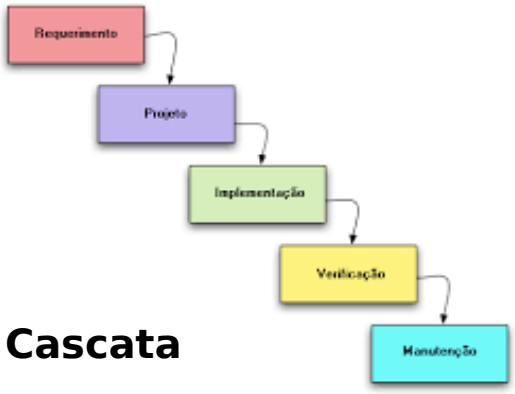
- Modelos **iterativos** de desenvolvimento (RAD, metodologias ágeis, prototipagem)
  - O produto resultante de uma iteração **pode ser testado** em vários níveis como parte do seu desenvolvimento.

Incremental





Espiral

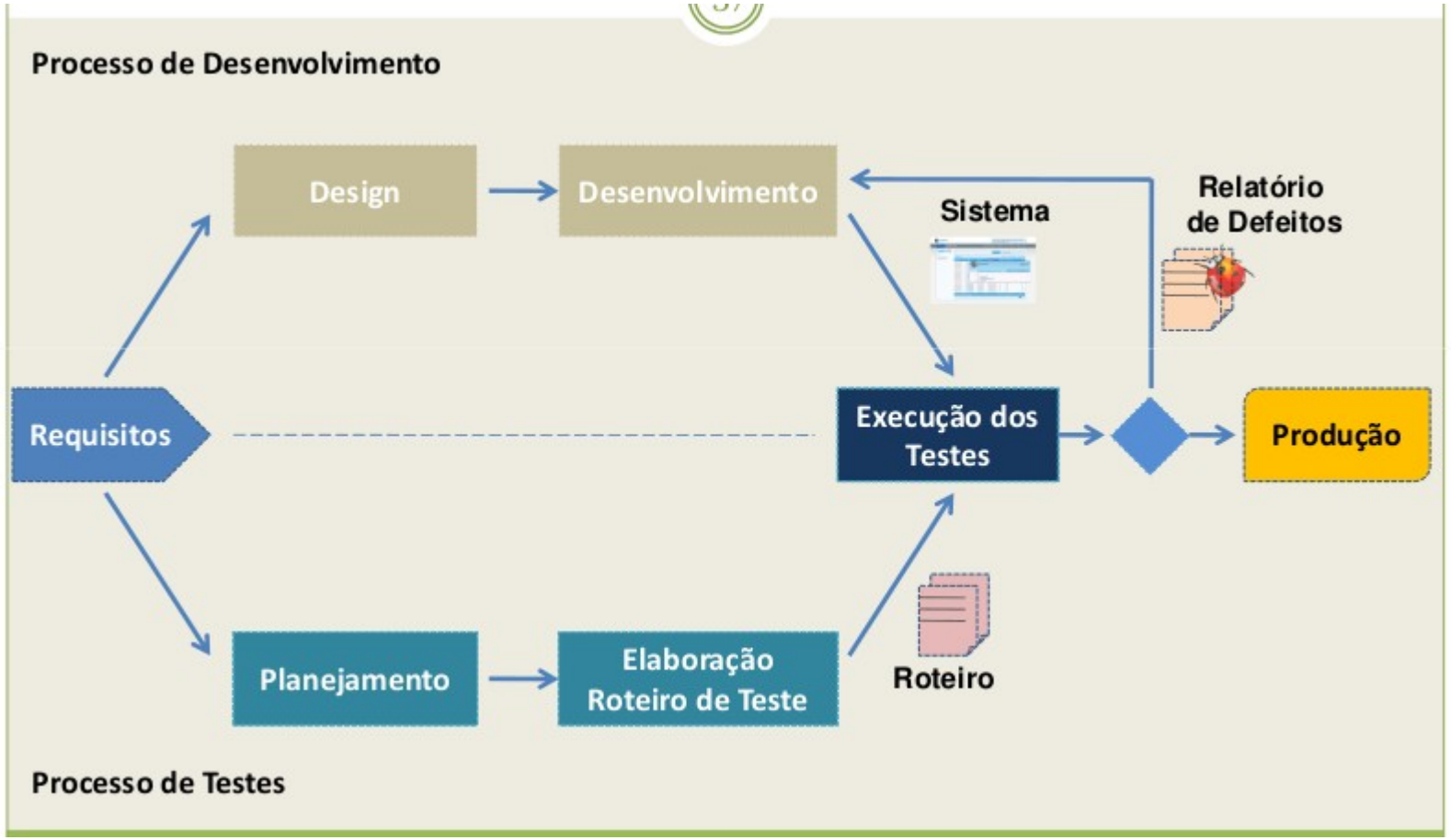


Cascata

Teste dentro de um modelo de ciclo de vida

- Características para um bom teste para **qualquer** modelo de ciclo de vida:
  - Toda **atividade do desenvolvimento** há uma **atividade de teste** correspondente.
  - Cada **nível de teste** tem um **objetivo** específico daquele nível.
  - A **análise e modelagem do teste** para um dado nível de teste devem começar **durante a atividade de desenvolvimento correspondente**.
  - Testadores devem se envolver na revisão de documentos o mais cedo possível utilizando as primeiras versões disponíveis ao longo ciclo de desenvolvimento.

Processo de Testes x Processo de Desenvolvimento



Teste durante o ciclo de vida do software

- Modelos de Desenvolvimento de Software
  - Modelo V
  - Modelos iterativos de desenvolvimento
  - Teste dentro de um modelo de ciclo de vida
- **Níveis de Teste**
  - **Teste de Componente (ou Unidade)**
  - **Teste de Integração**
  - **Teste de Sistema**
  - **Teste de Aceite**
- Tipos de Teste: o alvo do teste
  - Teste de Função (Teste funcional)
  - Teste de características do produto de software (testes não funcionais)
  - Teste de estrutura/arquitetura do software (teste estrutural)
  - Teste relacionado a mudanças (teste de confirmação e regressão)
- Teste de Manutenção

Níveis de Teste

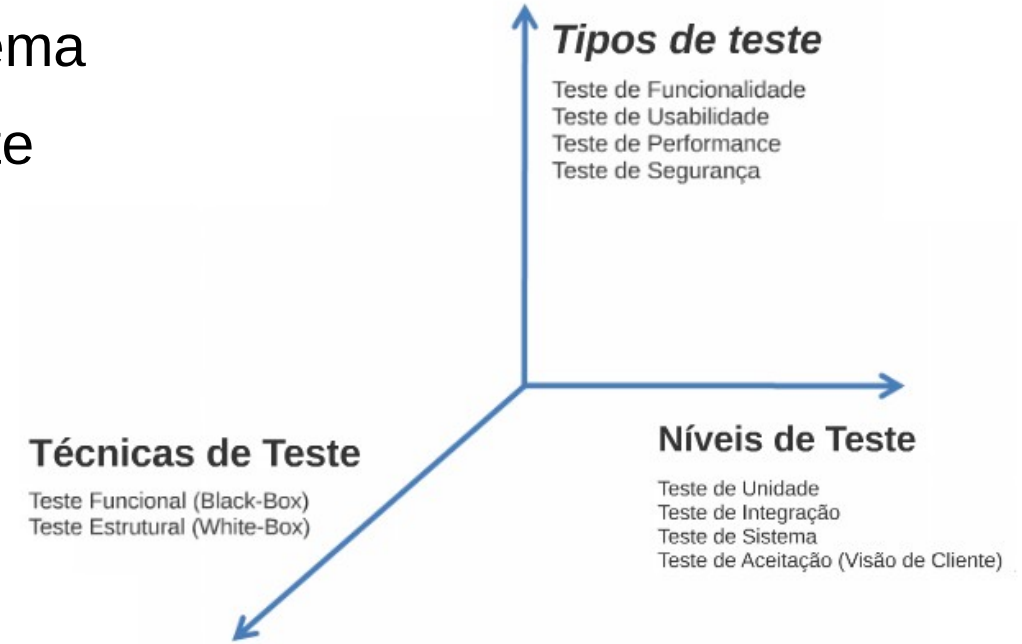
- Para cada nível de teste, os seguintes aspectos podem ser identificados:
  - os objetivos do teste;
  - os produtos de trabalho utilizados como referência para derivar os **casos de testes** (ex.: **base de teste**);
  - o objeto do teste (o que está sendo testado – componente de software, sistema ...);
  - defeitos e falhas típicas a se encontrar;
  - testes automatizados (*"harness"*);
  - ferramentas de suporte;
  - abordagens e responsabilidades específicas.

Casos de teste: conjunto de valores de entrada, precondições de execução, resultados esperados e pós-condições de execução

Base de teste: todos os documentos a partir dos quais os requisitos de um determinado componente de Software podem ser inferidos. Documentação na qual os casos de testes estão baseados.

Níveis de Teste

- Quais são?
  - Teste de Componente (ou Unidade)
  - Teste de Integração
  - Teste de Sistema
  - Teste de Aceite



Teste de Componente (ou Unidade)

- Procura defeitos e verifica o funcionamento do software (ex.: módulos, programas, objetos, classes, etc.) que são testáveis separadamente;
- Pode ser feito isolado do resto do sistema, **dependendo do contexto do ciclo de desenvolvimento** e do sistema.
- Controladores (“drivers”) e simuladores (“stubs”) podem ser usados.
- Pode incluir teste de funcionalidade e características específicas não funcionais como:
  - comportamento dos recursos (ex.: falta de memória);
  - testes de robustez (capacidade do sistema funcionar mesmo em condições anormais);
  - teste estrutural (cobertura de código);
- Casos de teste são derivados dos produtos de trabalho como, por exemplo, especificação de componente, modelagem do software ou modelo de dados.

Teste de Componente (ou Unidade)

- Tipicamente, teste de componente **ocorre com acesso ao código que está sendo testado** e no **ambiente de desenvolvimento**, assim como um *framework* de teste de unidade ou ferramenta de depuração "*debugging*".
- Na prática, **envolve o programador do código**.
- **Defeitos são normalmente corrigidos assim que são encontrados** sem registrar formalmente os incidentes.
- Uma abordagem no teste de componente consiste em preparar e automatizar os casos de testes **antes de codificar:** teste antecipado ou desenvolvimento dirigido a teste.
  - Esta abordagem é essencialmente iterativa e é baseada em **ciclos de elaboração de casos de testes**.
  - À medida que são construídas e integradas pequenas partes do código, são executados testes de componente até que eles passem.

Teste de Integração

- É caracterizado por **testar as interfaces** entre os componentes, **interações de diferentes partes de um sistema**, como o sistema operacional, arquivos, hardware ou interfaces entre os sistemas.
- Pode haver mais que um nível de teste de integração, que pode ser utilizado em **objetos de teste** de tamanho variado.
- Por exemplo:
  - Teste de integração de componente testa interações entre componentes de *software* e é realizado após o teste de componente.
  - Teste de integração de sistemas testa interação entre diferentes sistemas e pode ser realizado após o teste de sistema. Neste caso a área de desenvolvimento pode controlar apenas um lado da interface, de forma que mudanças podem causar instabilidades.
- Quanto maior o escopo da integração, maior a dificuldade de isolar as falhas para componentes ou sistemas específicos, fato que pode representar um aumento no risco.

Objetos de teste: Componente ou sistema a ser testado

Teste de Integração

- Visando reduzir o risco de encontrar defeitos tardiamente, a integração deve, preferencialmente, **ser incremental e não "big bang"**.
- **Teste de características não funcionais** específicas (por exemplo, performance) pode ser incluído nos testes de integração.
- A cada estágio da integração, os testadores concentram somente na integração propriamente.
  - Por exemplo, o módulo A está sendo integrado com o módulo B o foco é a **comunicação entre os módulos, não suas funcionalidades**. Tanto testes funcionais quanto estruturais podem ser utilizados.
- Idealmente, os testadores devem compreender a arquitetura e influenciar no planejamento da integração.
  - Se o teste de integração for planejado antes que os componentes ou sistemas estejam prontos, eles podem ser preparados visando um teste mais eficiente.

Teste de Sistema

- Se refere ao comportamento de todo do sistema / produto definido pelo escopo de um projeto ou programa de desenvolvimento.
- O ambiente de teste deve corresponder **o máximo possível ao objetivo final, ou o ambiente de produção**, para minimizar que os riscos de falhas específicas de ambiente não serem encontradas durante o teste.
- Podem ser baseados em:
  - especificação de riscos e/ou de requisitos,
  - processos de negócios,
  - casos de uso, dentre outras descrições de alto nível do comportamento e interações
- Deve tratar **requisitos funcionais e não funcionais** do sistema. Os requisitos podem estar como um texto ou diagramas.
- Em requisitos funcionais deve inicialmente utilizar a técnica baseada em especificação mais apropriada (caixa-preta) de acordo com a característica do sistema a ser testado.
  - Por exemplo, uma tabela de decisão pode ser criada por combinações de efeitos descritos em regras de negócio. Em seguida, técnica baseada na estrutura (caixa-branca) pode ser utilizada para avaliar a eficácia do teste com respeito ao elemento estrutural, assim como estrutura do menu ou página web. (Veremos no Capítulo 4.)
- Uma **equipe de teste independente** é frequentemente responsável pelo teste de sistema.

Teste de Aceite

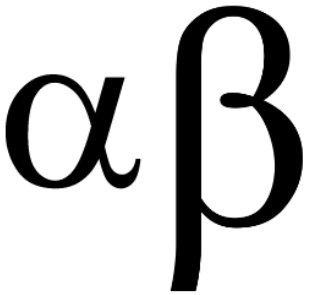
- Frequentemente é de **responsabilidade do cliente ou do usuário do sistema**; os interessados (stakeholders) também podem ser envolvidos.
- O objetivo é estabelecer a confiança no sistema, parte do sistema ou uma característica não específica do sistema. Procurar defeitos não é o principal foco do teste de aceite. Teste de aceite pode **avaliar a disponibilidade do sistema para entrar em produção**, apesar de não ser necessariamente o último nível de teste.
- Pode ser realizado em mais de um único nível de teste, por exemplo:
  - Um pacote (COTS) de *software* ter um teste de aceite quando é instalado ou integrado.
  - Teste de aceite de usabilidade de um componente pode ser feito durante o teste de componente.
  - Teste de aceite de uma nova funcionalidade pode vir antes do teste de sistema.

Formas de Teste de Aceite

- **Teste de Aceite de Usuário**: Normalmente verifica se o sistema está apropriado para o uso por um usuário com perfil de negócio.
- **Teste Operacional de Aceite**: O aceite do sistema pelo administrador dos sistemas inclui: Teste de *Backup/Restore*; Recuperação de Desastre; Gerenciamento de Usuário; Tarefas de manutenção; Checagens periódicas de vulnerabilidades de segurança.
- **Teste de aceite de contrato e regulamento**: é realizado verificando-se algum critério de aceite incluso em contrato na produção de *software* encomendado. ***O critério de aceite deve ser definido quando o contrato é assinado***. Teste de aceite de regulamento é quando se verifica a necessidade de adesão a algum regulamento de acordo com outras normas (ex.: segurança, governamental, legislação).

Teste de Aceite

- **Alfa e Beta Teste (ou teste no campo)**: desenvolvedores de *softwares* comerciais ou pacotes, muitas vezes precisam obter um *feedback* de clientes em potencial existente no mercado antes que o *software* seja colocado à venda comercialmente.
  - **Alfa Teste** é feito no “site” da organização em que o produto foi desenvolvido.
  - **Beta Teste**, ou teste no campo, é feito pelas pessoas em suas próprias localidades.
  - Ambos os testes são feitos pelos clientes em potencial e **não pelos desenvolvedores do produto**.
- Organizações podem utilizar outros termos como Teste de Aceite de Fábrica e Teste de Aceite no “site”, para sistemas que são testados antes e após terem sido movidos ao “site” do cliente.



Quadro resumo dos Níveis/Fases de Teste

Nível/Fase	Foco	Erros que podem ser revelados	Observação
Teste de Unidade/Componente	Menores unidades de um *software* (funções, procedimentos, métodos, classes)	Erros simples de programação, estruturas de dados incorretas, algoritmos mal implementados ou incorretos.	Pode ser feito durante a implementação e pelo próprio desenvolvedor.
Teste de Integração	Construção da estrutura do software	Interfaces entre as unidades	Feitos após testar as unidades. Executados pela equipe de desenvolvimento que conhece a estrutura interna do *software*.
Teste de Sistema	O Sistema	Erros de funções e características de desempenho que não estejam de acordo com a especificação.	Feito após o sistema estar completo. Pode ser feito por uma equipe independente. Pode-se considerar requisitos funcionais e não funcionais (segurança, performance, etc).

Teste durante o ciclo de vida do software

- Modelos de Desenvolvimento de Software
  - Modelo V
  - Modelos iterativos de desenvolvimento
  - Teste dentro de um modelo de ciclo de vida
- Níveis de Teste
  - Teste de Componente (ou Unidade)
  - Teste de Integração
  - Teste de Sistema
  - Teste de Aceite
- **Tipos de Teste: o alvo do teste**
  - **Teste de Função (Teste funcional)**
  - **Teste de características do produto de software (testes não funcionais)**
  - **Teste de estrutura/arquitetura do software (teste estrutural)**
  - **Teste relacionado a mudanças (teste de confirmação e regressão)**
- Teste de Manutenção

Tipos de Teste: o alvo do teste

- Um grupo de atividades de teste pode ser direcionado para verificar o sistema (ou uma parte do sistema) **com base em um motivo ou alvo específico**.
- Cada **tipo de teste** tem foco em um objetivo particular, que pode ser:
  - o teste de uma funcionalidade, a ser realizada pelo *software*;
  - uma característica da qualidade não funcional, tal como a confiabilidade ou usabilidade, a estrutura ou arquitetura do *software* ou sistema;
  - mudanças relacionadas (p.e.: confirmar que os defeitos foram solucionados - teste de confirmação - e procurar por mudanças inesperadas - teste de regressão.
- Modelos do *software* podem ser elaborados e/ou usados no teste estrutural ou funcional.
- Por exemplo:
  - **Para o teste funcional**: um diagrama de fluxo de processo, um diagrama de transição de estados ou uma especificação do programa;
  - **Para teste estrutural**: um diagrama de controle de fluxo ou modelo de estrutura do menu.

Teste de Função ou Caixa-preta

- Também chamado de Teste Funcional.
- As funções que um sistema, subsistema ou componente devem realizar podem ser descritas nos seguintes produtos de trabalho:
  - especificação de requisitos;
  - casos de uso;
  - especificação funcional;
  - ou podem não estar documentados.
- As funções representam “o que” o sistema faz. Testes funcionais são baseados em funções (descritas nos documentos ou compreendidas pelos testadores), e devem ser realizados **em todos os níveis de teste**.
- Técnicas baseadas em especificação podem ser utilizadas para derivar as condições de teste e casos de testes a partir da funcionalidade do *software* ou sistema.
- Teste funcional considera o comportamento externo do software (teste caixa-preta).
- Um tipo de teste funcional, o teste de segurança, investiga as funções (ex.: um “*firewall*”) relacionados à detecção de ameaça de vírus ou de ações mal intencionadas.

Testes não funcionais

- Podem ser realizados em todos os **níveis de teste**.
- Testes de **características** do produtos de *software*. É o teste de "***como***" o sistema trabalha.
- Testes não funcionais incluem, mas não se limitam a:
  - teste de performance;
  - teste de carga (avaliar o comportamento de um componente ou sistema com carga crescente, p.e., número de usuários paralelo e/ou o número de transações, para determinar qual a carga pode ser manipulada por um componente ou sistema);
  - teste de estresse (avaliar se um sistema ou componente está no limite ou além do limite da sua carga de trabalho prevista ou especificada, ou com menor disponibilidade de recursos, como acesso à memória ou servidores.);
  - teste de usabilidade (determina a extensão até a qual o produto de software é entendido, fácil de aprender, fácil de operar e atraente para os usuários sob condições especificas);
  - teste de interoperabilidade (capacidade do produto de software de interagir com um ou mais componentes especificados ou sistemas);
  - teste de manutenibilidade (testa as alterações feitas em um sistema operacional ou o impacto de um ambiente alterado em um sistema operacional.);
  - teste de confiabilidade (capacidade do produto de software em executar suas funções exigidas sob condições estabelecidas durante um determinado período de tempo, ou para um determinado número de operações);
  - teste de portabilidade (facilidade com que o produto de software pode ser transferido de um ambiente de *hardware* ou *software* para outro).
- O teste é executado para medir as características que podem ser **quantificadas** em uma escala variável (p.e.: tempo de resposta em um teste de performance).
- Estes testes podem ser referenciados a um modelo de qualidade como definido na norma ISO 9126.

Testes Estruturais ou Caixa-branca

- Também conhecidos por Teste de Estrutura ou de Arquitetura do *software*
- Pode ser feito em todos os níveis de testes.
- **Recomenda-se utilizar as técnicas estruturais após as técnicas baseadas em especificação, já que ela auxilia a medição da eficiência do teste através da avaliação da cobertura de um tipo de estrutura**.
- ***Cobertura*** é a extensão que uma estrutura foi exercitada por um conjunto de testes, expresso como uma **porcentagem** de itens cobertos.
- Se a cobertura não atinge 100%, então mais testes devem ser construídos a fim de testar aqueles itens que não foram contemplados para, desta forma, aumentar a cobertura.
- Em todos os níveis de teste, mas especialmente no teste de componente/unidade e teste de integração de componentes, ferramentas podem ser usadas para medir a cobertura do código dos elementos assim como as declarações ou decisões.
- **Teste estrutural deve ser baseado na arquitetura do sistema**.
- Teste de estrutura também pode ser aplicado no sistema, integração de sistema ou nível de teste de aceite (por exemplo, para modelos de negócios ou estrutura de menu).

Teste relacionado a mudanças

- São os Testes de Confirmação e de Regressão;
- **Teste de Confirmação**: quando um defeito é detectado e resolvido, o *software* pode ser retestado para confirmar que o defeito original foi realmente removido.
- **Teste de Regressão**: é o teste repetido de um programa que já foi testado, após sua modificação, para descobrir a existência de algum defeito introduzido ou não coberto originalmente como resultado da mudança.
  - Estes defeitos podem estar no *software* ou em um componente, relacionado ou não ao *software*. É realizado quando o software, ou seu ambiente é modificado.
  - A quantidade de teste de regressão é baseada no risco de não se encontrar defeitos no *software* que estava funcionando previamente.
- Os testes devem ser repetíveis se forem utilizados nos teste de confirmação e para suportar o teste de regressão.
- **Teste de regressão pode ser realizado em todos os níveis de teste, e se aplicam aos testes funcionais, não funcionais e estruturais.**
- Testes de regressão são executados muitas vezes e geralmente desenvolve-se vagarosamente, o que faz com que seja um **forte candidato à automação**.

Lembre-se que: Depurar (resolver defeitos) é uma atividade do desenvolvimento, e não uma atividade do teste.

Teste baseado em defeitos

- Há ainda uma classe de técnica de teste que se baseia em defeitos, isto é, **utiliza o conhecimento sobre a presença de defeitos típicos do processo de implementação para testar o *software***.
- Para saber mais sobre sobre esta técnica de teste, visite:

  https://www.coursera.org/lecture/intro-teste-de-software/teste-de-mutacao-contexto-geral-vBFVa

Atividades

1) Explique o teste de mutação, explique o seu objetivo
2) Como avaliar um caso de teste

Quadro resumo de Técnicas de Teste

Técnicas	Tipo de informação utilizada para derivar os requisitos de teste	Passos principais	Exemplos de critérios	Nível/Fase
Teste Funcional	Especificação do *software*	a) Identificar as funções que o *software* deve realizar. b) Criar casos de teste capazes de verificar se tais funções estão sendo executadas corretamente	- Particionamento em Classes de Equivalência. - Análise do valor limite. - Grafo Causa-Efeito.	Em todas as fases
Teste Estrutural	Conhecimento da estrutura interna do programa (**implementação**, **código fonte**)	A maioria dos critérios desta técnica utiliza uma representação de programa conhecida como grafo de programa.	Baseados em complexidade (McCabe); Baseados em fluxo de controle. Baseados em fluxo de dados.	Unidade e Integração
Técnica baseada em defeitos	Defeitos típicos do processo de implementação de software	Criar implementações alternativas e forçar o testador a projetar casos de teste que revelem os defeitos nelas introduzidos	Teste de Mutação	Unidade e Integração

Exercício: identificando as técnicas adequadas

O uso de uma técnica não exclui o uso de outra. Ou seja, você pode usar várias técnicas para gerar os casos de teste. A escolha de qual técnica utilizar dependerá de uma série de fatores, por exemplo o tipo do sistema, clientes, requisitos contratuais, nível e tipo de riscos, objetivos do teste, documentação disponível, conhecimento dos testadores, tempo, dinheiro, ciclo de desenvolvimento adotado (cascata, iterativo/incremental, etc.), modelo de caso de uso e uma experiência prévia dos tipos de defeitos encontrados, dentre outras questões.

Você e sua equipe receberam a tarefa de implantar o ScriptLattes, que é um script GNU-GPL desenvolvido para a extração e compilação automática de: produções bibliográficas, produções técnicas, produções artísticas, orientações, projetos de pesquisa, prêmios e títulos, grafo de colaborações e mapa de geolocalização de um conjunto de pesquisadores cadastrados na plataforma Lattes.

Vocês possuem informações sobre a especificação do ScriptLattes (manual, funcionalidades, atores, etc.), mas não possuem documentação necessária sobre o código em si e não possuem conhecimento da estrutura interna, a qual foi criada por terceiros.

Nesse cenário, qual seria a opção de Técnica de Teste mais apropriada para derivar os casos de teste?

a) Teste Estrutural

b) Teste de Mutação

c) Teste Funcional

Teste durante o ciclo de vida do software

- Modelos de Desenvolvimento de Software
  - Modelo V
  - Modelos iterativos de desenvolvimento
  - Teste dentro de um modelo de ciclo de vida
- Níveis de Teste
  - Teste de Componente (ou Unidade)
  - Teste de Integração
  - Teste de Sistema
  - Teste de Aceite
- Tipos de Teste: o alvo do teste
  - Teste de Função (Teste funcional)
  - Teste de características do produto de software (testes não funcionais)
  - Teste de estrutura/arquitetura do software (teste estrutural)
  - Teste relacionado a mudanças (teste de confirmação e regressão)
- **Teste de Manutenção**

Teste de Manutenção

- Uma vez desenvolvido, um sistema pode ficar ativo por anos ou até mesmo décadas.
- Durante este tempo o sistema e seu ambiente podem ser corrigidos, modificados ou completados.
- Teste de manutenção é realizado no mesmo sistema operacional e é iniciado por modificações, migrações ou retirada de *software* ou sistema.
- Alguns exemplos de modificações incluem:
  - melhorias planejadas (ex.: baseadas em "*releases*");
  - mudanças corretivas e emergenciais;
  - mudanças de ambiente (atualização em SO ou BD);
  - correções ("*patches*") para expor e encontrar vulnerabilidades do sistema operacional.
- **Teste de manutenção por migraçã**o (ex.: de uma plataforma a outra) pode incluir testes operacionais do novo ambiente tanto quanto a mudança de *software*.
- **Teste de manutenção para retirada de um sistema** pode incluir o teste de migração de dados, ou arquivamento se longos períodos de retenção de dados forem necessários.

Teste de Manutenção

- Além de testar o que foi alterado, o teste de manutenção inclui teste de regressão massivo para as partes do sistema que não foram testadas.
- O escopo do teste de manutenção está relacionado ao risco da mudança, ao tamanho do sistema existente e ao tamanho da mudança.
- Dependendo da mudança, o teste de manutenção pode ser feito em todos ou alguns níveis, e em todos ou alguns tipos de testes.
- A determinação de como um sistema pode ser afetado por mudanças é chamado de **análise de impacto**, e pode ser usado para ajudar a decidir quantos testes de regressão serão realizados.
- Teste de manutenção pode se tornar uma tarefa complicada se as especificações estiverem desatualizadas ou incompletas.

Referências

ISTQB. **Certified Tester Foundation Level Syllabus:** Capítulo 1. Ago. 2011. Acesso em: 12/09/2018. Disponível em: http://bstqb.org.br/uploads/syllabus/syllabus_ctfl_2011br.pdf

LAGARES, Vivian; ELIZA, Renata. Gestão de Defeitos no Teste de Software. DEVMEDIA, 2015. Acesso em: 12/09/2018. Disponível em: http://www.devmedia.com.br/gestao-de-defeitos-no-teste-de-software/21940

Justo, Daniela Sbizera Profª Dra. Introdução a teste de software. IFSC. Gaspar.

Anne Caroline O. Rocha – Tester Certified – BSTQB – NTI|UFPB