Universidade Federal do Ceará
Centro de Ciências/Departamento de Computação
Código da Disciplina: CK0236
Professor: Ismayle de Sousa Santos

Aula 12

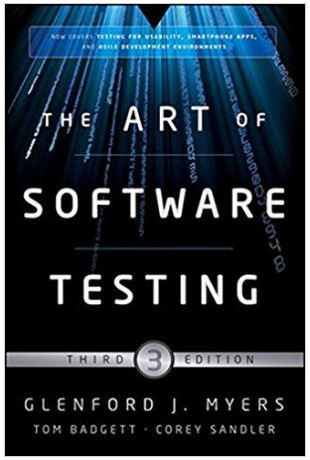
Técnica de Programação II

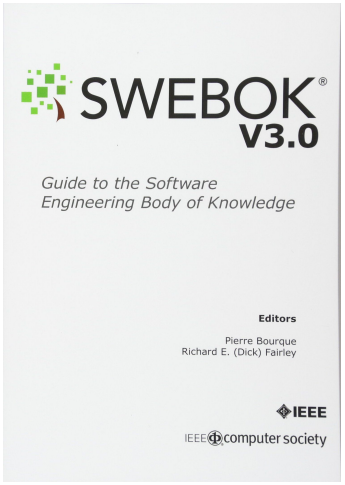
Testes - Princípios, Níveis e Técnicas

qpg4p5x ismaylesantos@great.ufc.br @IsmayleSantos

O que é teste de software?

- É o processo de executar um programa com a intenção de encontrar a presença erros [Myers 2011]
- "Consiste em uma verificação **dinâmica** do comportamento de um programa com um conjunto **finito** de casos de testes adequadamente **selecionados** a partir do domínio de execuções, geralmente infinito, contra o comportamento **esperado**." [SWEBOK 2013]

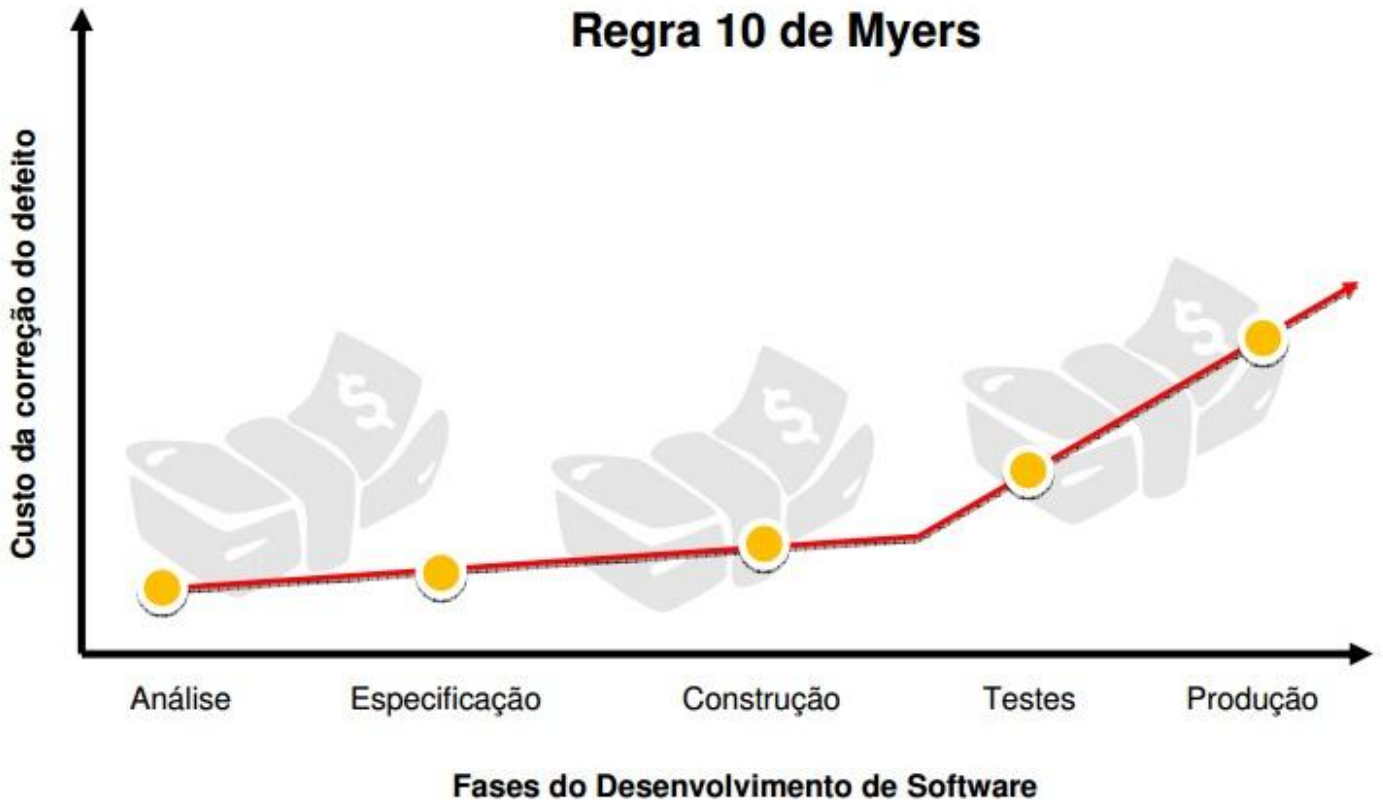




O que é teste de software?

- É uma técnica dinâmica de V&V (Verificação e Validação)
- **Verificação**
  - Assegurar consistência, completude e corretude do produto em cada fase e entre fases consecutivas
  - *Estamos desenvolvendo certo o produto?*
- **Validação**
  - Assegurar que o produto final corresponda aos requisitos do usuário
  - *Estamos desenvolvendo o produto certo?*

Regra 10 de Myers



Fonte: http://www.portalgsti.com.br

Cenário de Teste

- Caminho ou situação a ser testada. Eles focam na **funcionalidade** e não nas entradas dos dados
  - E.g.: Validar o funcionamento do Login do sistema
- A partir dele é definido um conjunto de casos de testes
- Podem ser criados a partir de
  - Casos de Uso
  - Histórias de Usuários
  - Critérios de Aceitação

Caso de Teste

- Consiste de um conjunto de entradas (incluindo ações, quando aplicável), pré-condições de execução e resultados esperados projetados para conduzir a execução de um item de teste para verificar o objetivo do teste [IEEE 29119-1 2013]
- Podem ser derivados a partir de cenários de testes

Caso de Teste

ID	CT001	
Descrição	Fazer login no GREat Tour	
Itens a testar	Verificar se a tentativa de fazer login com um usuário e senha corretos é realizada com sucesso.	
Entradas	**Campo**	**Valor**
	Username	guest
	Password	1234
Saída	Tela inicial da aplicação com o mapa da Recepção do laboratório.	
Procedimento	PT001	

Os Sete Princípios dos Testes

1º Testes indicam a presença de defeitos

E nunca indicam a ausência de defeitos

Os Sete Princípios dos Testes

2º Testes exaustivos são impossíveis

Validar todos as combinações de entradas e pré-condições não é viável

Os Sete Princípios dos Testes

3º Antecipação dos Testes

A atividade de testes deve começar o mais breve possível

Os Sete Princípios dos Testes

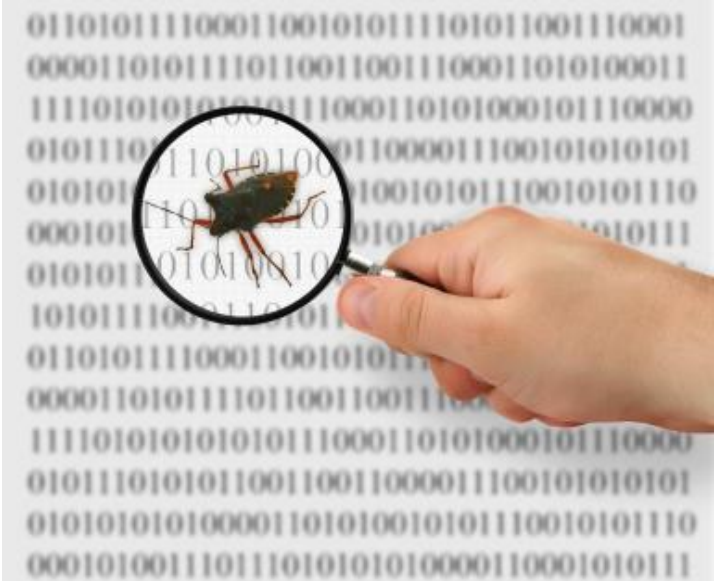
4º Agrupamento de Defeitos

A maior parte dos defeitos são encontrados em uma pequena quantidade de módulos

Os Sete Princípios dos Testes

5º O Paradoxo do Pesticida

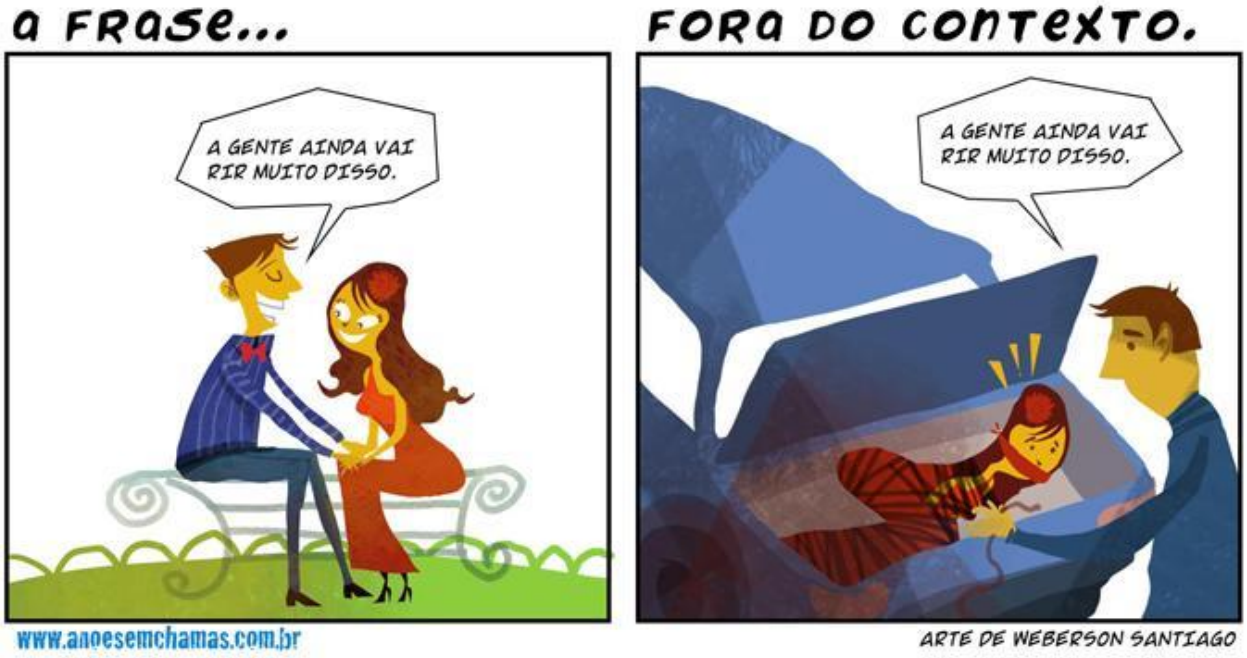
Um conjunto de testes, se executado várias vezes, pode não mais detectar novas falhas.Para contornar esse problema, os casos de teste devem ser frequentemente revisados e atualizados



Os Sete Princípios dos Testes

6º Testes dependem do Contexto

Os testes devem ser elaborados de acordo com o tipo de software

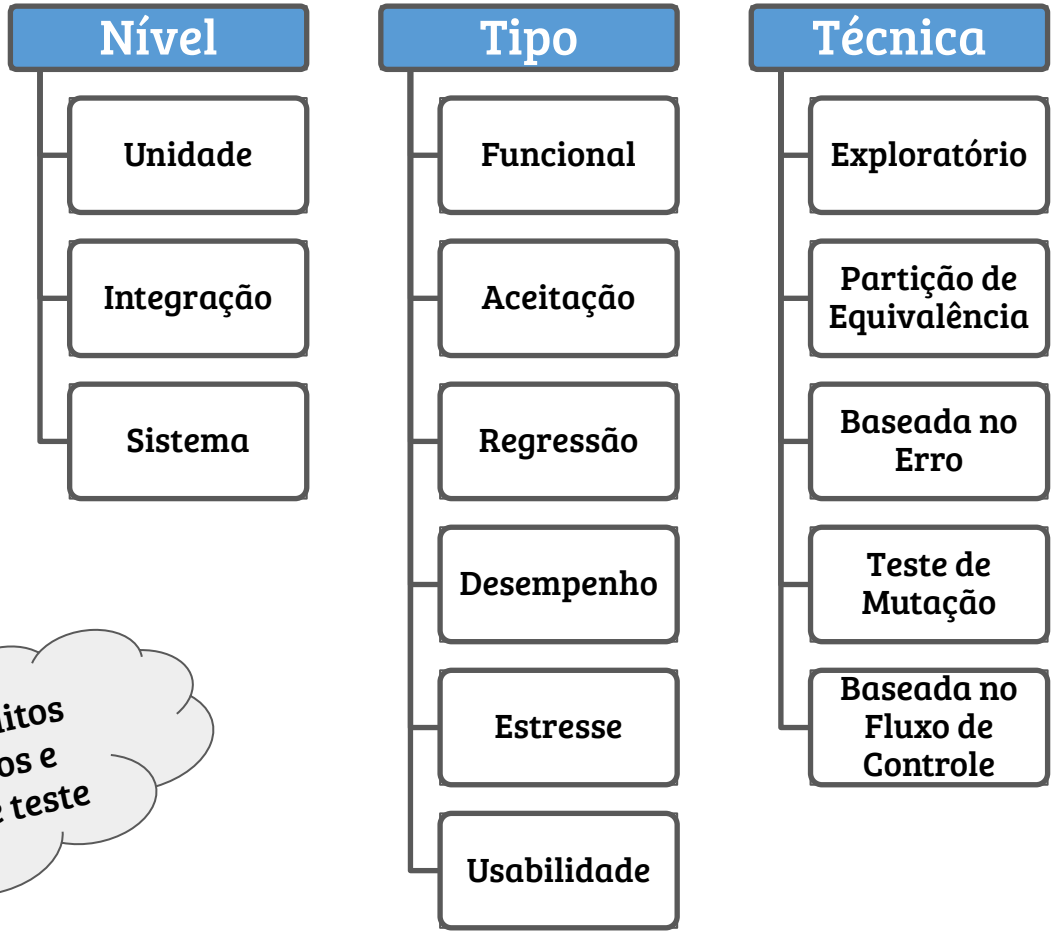


Os Sete Princípios dos Testes

7º A ausência de erros é uma ilusão

De nada adianta o sistema estar funcionando corretamente se ele não atender a necessidade real do usuário

Classificação dos testes

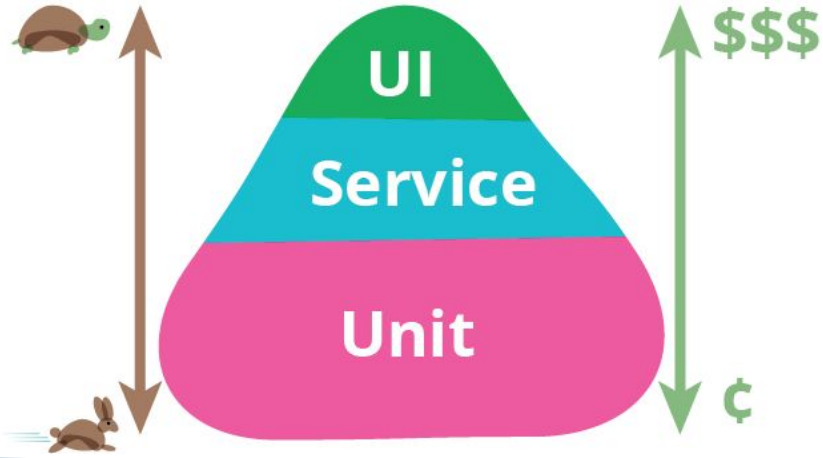


Níveis de Teste

Unidade, Integração e Sistema

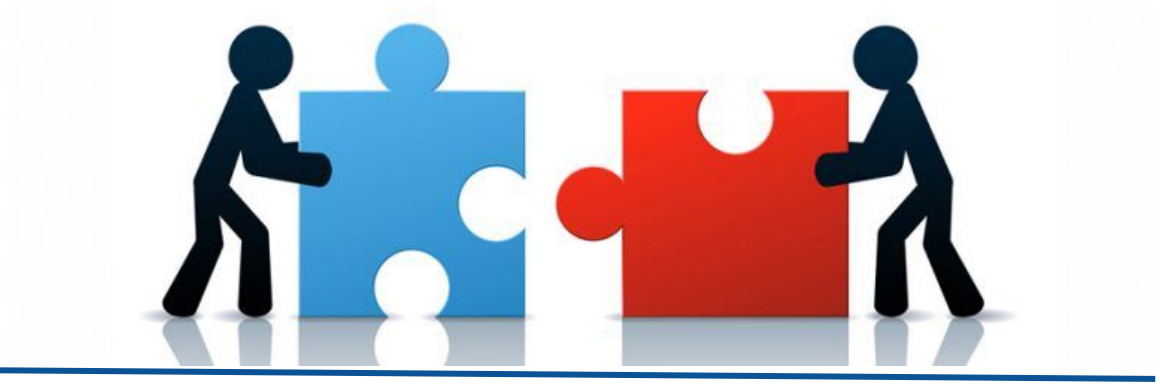
Testes Unitários

- Verifica o funcionamento isolado de pedaços do software (ex.:módulos, objetos classes, etc.) que são testáveis separadamente [SWEBOK 2013]



Teste de Integração

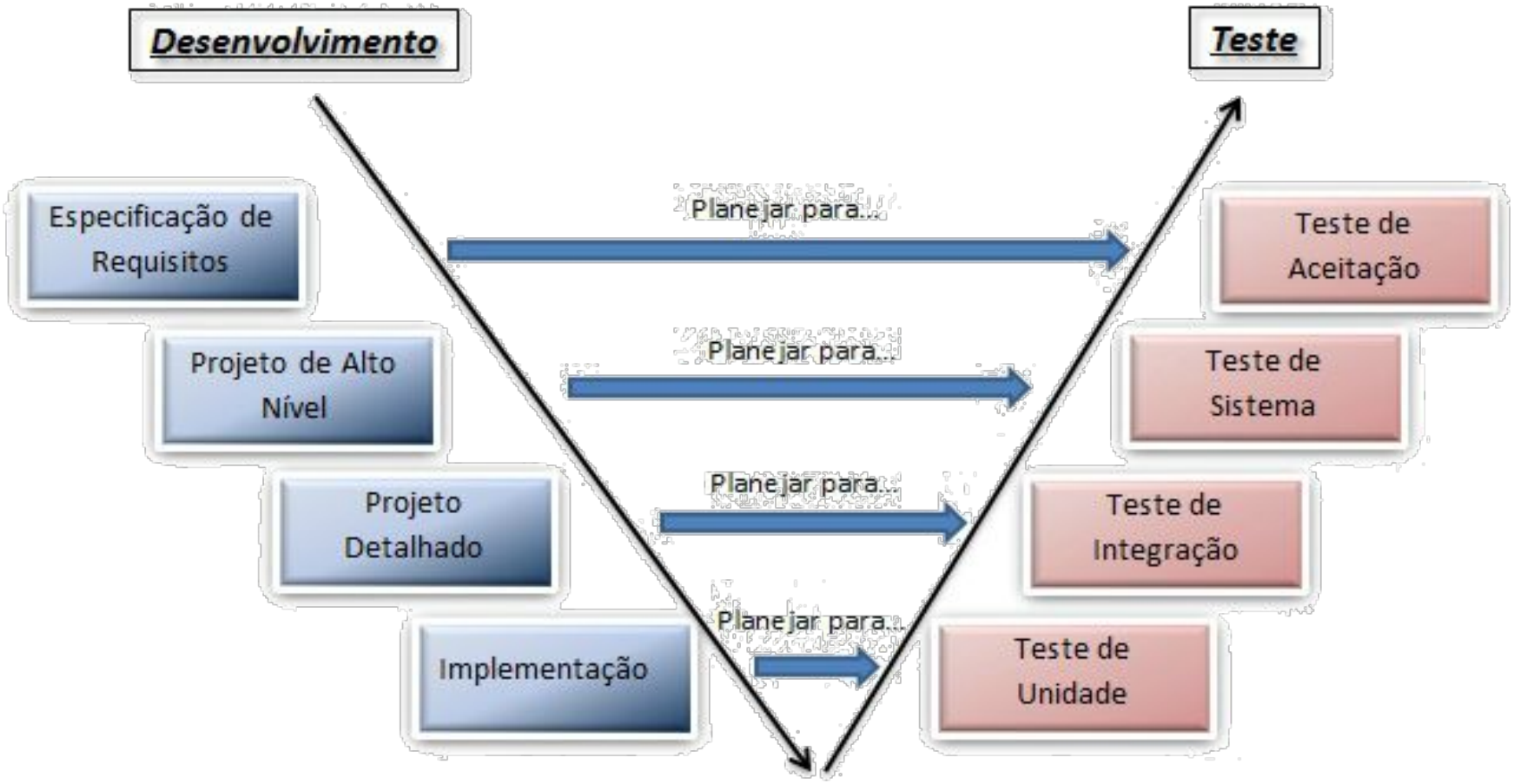
- É o processo de verificar a interação entre componentes de software [SWEBOK 2013]
- Procura por defeitos associados às interfaces entre os módulos



Teste de Sistema

- Verifica o comportamento do sistema como um todo [SWEBOK 2013]

Modelo V


Desenvolvimento
Teste
Especificação de Requisitos
Projeto de Alto Nível
Projeto Detalhado
Implementação
Planejar para...
Planejar para...
Planejar para...
Planejar para...
Teste de Aceitação
Teste de Sistema
Teste de Integração
Teste de Unidade

Algumas Tipos de Teste

Funcional, Aceitação, Regressão, Desempenho, Estresse e Usabilidade

Teste Funcional (Teste de conformidade)

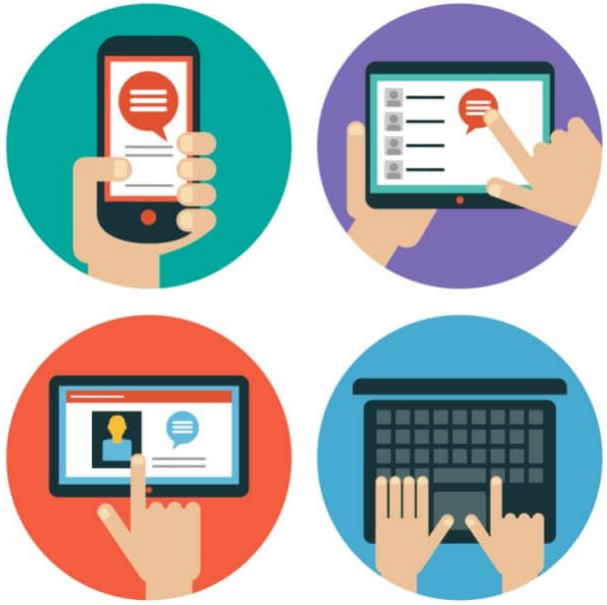
- Verifica que as especificações funcionais foram corretamente implementadas [SWEBOK 2013]

Teste de Aceitação

- Verifica o comportamento do Sistema contra os requisites do cliente [SWEBOK 2013]

Teste de Usabilidade

- Esse tipo de teste avalia o quão fácil é para os usuários finais aprenderem e usarem o software
- Em geral, envolve a documentação que auxilia os usuários e as funcionalidades que suportam as tarefas dos usuários



Teste de Desempenho

- Teste de Desempenho (performance testing)
  - Verifica que o software está de acordo com requisitos de desempenho especificados (e.g capacidade e tempo de resposta) [SWEBOK 2013]
  - Exemplo de pergunta a ser respondida
    - A aplicação suporta 1.000 transações por minuto com 1.000 usuários simultâneos?

Teste de Estresse

- Exercita o software no máximo de carga para o qual ele foi projetado, bem como além dele [SWEBOK 2013]
  - Alguns cenários de crash da aplicação são testados, com o objetivo de determinar a capacidade de recuperação e estabilidade do sistema
  - Exemplo de pergunta a ser respondida
    - Quantas transações por minuto solicitadas por 5.000, 6.000, 7.000 usuários simultâneos, serão suportadas pela aplicação sob condições não especificadas do software e até mesmo do próprio hardware?

Teste de Regressão

- Re-teste seletivo de um sistema ou componente para verificar que modificações não causaram efeitos não esperados e que o sistema ou componente continua de acordo com os requisitos. [SWEBOK 2013]
  - Pode-se re-testar tudo, mas em geral, é muito custoso
  - Dicas para seleção dos teste para re-execução
    - Usar rastreabilidade entre requisitos e testes
    - Identificar rotinas críticas (mais utilizadas/que não podem apresentar problemas)
    - Reexecutar testes que falharam
    - Priorizar os testes de acordo com algum critério
      - e.g. tempo de execução, probabilidade de encontrar falhas

Algumas Técnicas de Teste

Teste Exploratório, Partição de Equivalência, Baseada no Erro, Teste de Mutação e Baseada no Fluxo de Controle

Classificação das técnicas de Teste

- De acordo com a ISO 29119-4, as técnicas de teste podem ser classificadas em
  - Técnicas baseadas na especificação
    - Casos de testes criados a partir da documentação do comportamento esperado
    - Partição de Equivalência
  - Técnicas baseadas na estrutura
    - Casos de testes criados a partir da estrutura do código-fonte
    - E.g.: Baseada no fluxo de controle
  - Técnicas baseadas na experiência
    - Casos de testes criados a partir da experiência do testador e conhecimento do domínio
    - E.g.: Testes exploratórios

Teste Exploratório

- Neste caso, temos simultaneamente aprendizado, projeto de teste e execução de teste. Os testes não são definidos previamente, mas na verdade são projetados, executados e modificados dinamicamente. A efetividade depende da experiência do testador [SWEBOK 2013]
- Heurística IOSC
  - **(I)nput**: Exploração do comportamento do software sob a perspectiva das entradas de dados
  - **(O)utput**: Exploração do comportamento do software sob a perspectiva das saídas de dados
  - **(S)torage**: Exploração do comportamento do software sob a perspectiva dos dados armazenados
  - **(C)omputation**: Exploração do comportamento do software sob a perspectiva da computação/processamento dos dados

Teste Exploratório

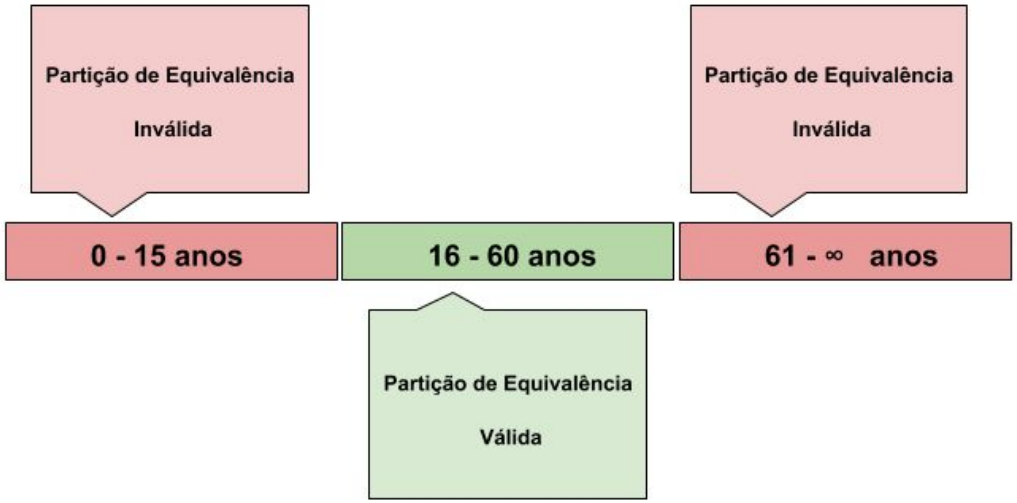
- É possível predizer o funcionamento supostamente correto observando
  - Consistência com a proposta: o comportamento deve ser consistente com sua proposta
  - Consistência com o resto da aplicação: o comportamento deve ser consistente com o comportamento de outras áreas do aplicativo
  - Consistência histórica: o comportamento deve ser consistente ao longo do tempo
  - Consistência com aplicativos semelhantes: o comportamento deve ser consistente com o comportamento de aplicativos semelhantes

Teste Exploratório

- Outras formas de extrair informações para os oráculos
  - Por meio de perguntas ao cliente/desenvolvedor
  - Checklist com características gerais da aplicação (e.g. mensagens de erro e sucesso)

Partição de Equivalência (PE)

- Neste caso, o domínio da entrada é dividido em uma coleção de classes equivalentes, e testes são feitos para cada classe [SWEBOK 2013]
  - Em geral, é usada junto com a Análise de Valor Limite (AVL), que sugere exercitar os valores limítrofes
  - As classes podem ser organizadas entre válidas e inválidas



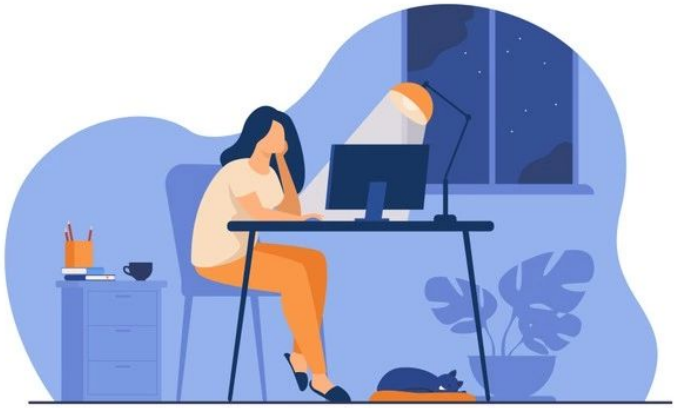
Fonte: https://medium.com/revista-tspi/t%C3%A9cnica-de-teste-particionamento-de-equival%C3%AAncia-d32a7d689d82

Exemplo de Partição de Equivalência (PE)

Entrada	Valores Permitidos	Classes de Equivalência	Casos de teste com PE	Casos de testes com PE e AVL
Idade	Entre 0 e 120 anos	Idade < 0	Idade = -5	Idade = -1
		0 <= Idade <= 120	Idade = 20	Idade = 0 Idade = 1 Idade = 119 Idade = 120
		Idade > 120	Idade = 130	Idade = 121

Baseado em Erro (Error Guessing)

- Testes são projetados para descobrir as possíveis faltas em um dado programa. Uma boa fonte de informação é o histórico de faltas descoberto em projetos anteriores, bem como a experiência dos testadores [SWEBOK 2013]



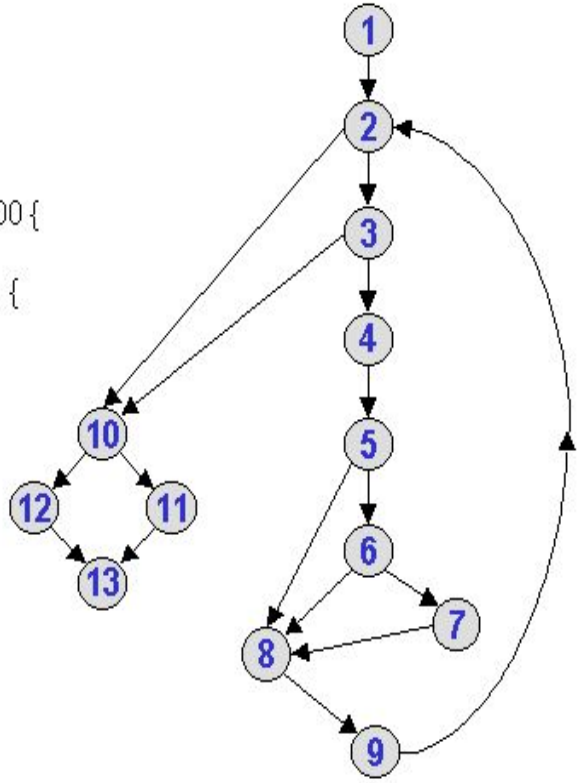
Teste de Mutação (Mutation Testing)

- Um mutante é uma versão ligeiramente modificada do programa sob teste, diferenciando dele por uma pequena mudança. Testes são então projetados para identificar os mutantes [SWEBOK 2013]

Baseado no Fluxo de Controle

- Neste caso, os testes são projetados para cobrir todas as instruções

```
1   i = 1;
    total_entrada = total_valido = 0;
    soma= 0;
2   while valor[i] != -999 && total_entrada < 100 {
3       total_entrada++;
4       if valor[i] ? minimo && valor[i] ? maximo {
5           total.valido++;
6           soma = soma + valor[i];
7       }
8       i++;
    }
9   if total_valido > 0
10      media = soma/total_valido;
    else
11      media = -999;
12  }
```



Fonte: https://www.dimap.ufrn.br/~jair/ES/c8.html

Obrigado!

Por hoje é só pessoal...

Dúvidas?

qpg4p5x

ismaylesantos@great.ufc.br

@IsmayleSantos