Teste de Software

Gilmar Ferreira Arantes
- gilmar.arantes@goias.gov.br

Gesis - STI

Goiânia, 17 de Janeiro de 2020

INSTITUTO DE INFORMÁTICA UFG

Sumário I

INSTITUTO DE INFORMÁTICA UFG

Quem Sou Eu...

1. Nome: Gilmar Ferreira Arantes
2. Gradução - Análise de Sistemas - Universo - 1999-2003;
3. MBA Gestão de TI - Alfa - 2004-2006;
4. Especialista em Análise e Projeto de Software - UFG - 2006-2007;
5. Mestre em Ciência da Computação - UFG - 2009-2012;
6. Gestor de Tecnologia da Informação - SEDI - 01/02/2007;
7. Professor Assistente - INF/UFG - 28/11/2013.
8. Emails: gilmar.arantes@goias.gov.br, gilmar.arantes@ufg.br e gilmar.arantes@gmail.com

INSTITUTO DE INFORMÁTICA UFG

E Vocês?

1. Nome.
2. Atividades na Seduc.
3. Experiência com Testes.

Filosofando...

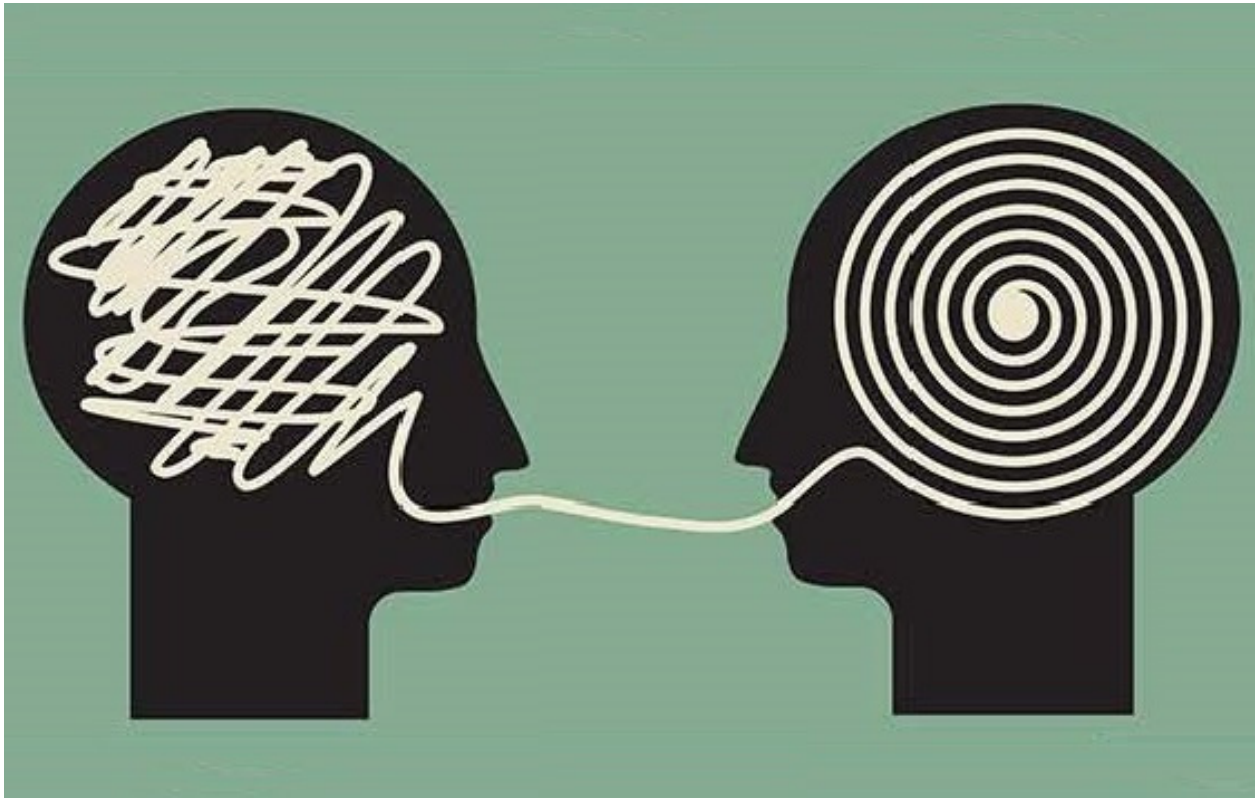


Figure: 1

- Filosofando. [Google, 2019]

Indivíduo x Sociedade

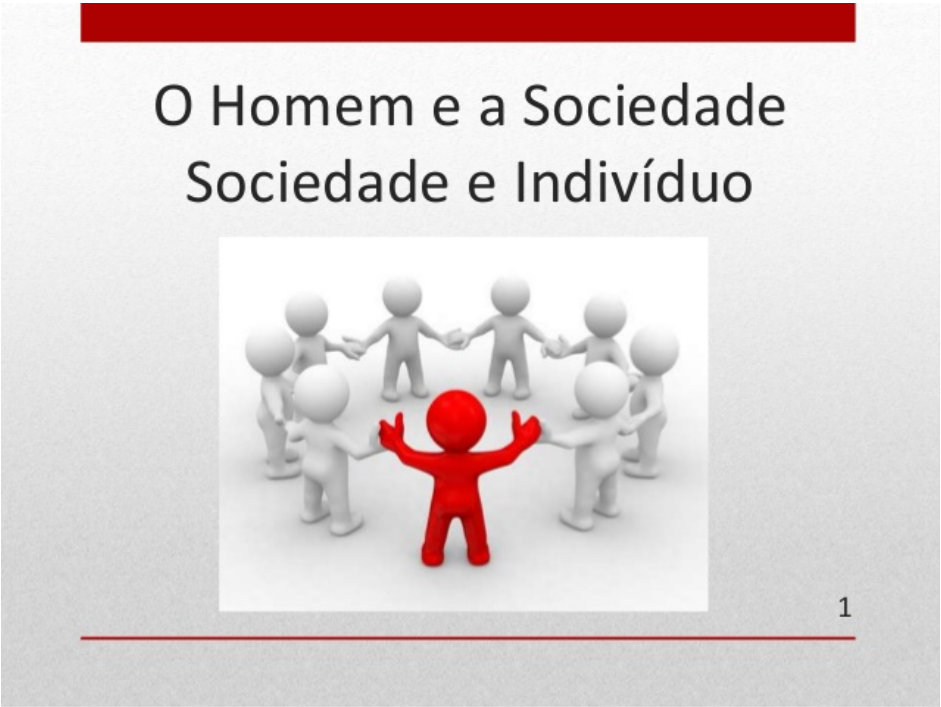


Figure: 2

- Indivíduo x Sociedade. [Google, 2019]

INSTITUTO DE INFORMÁTICA UFG

Indivíduo x Sociedade

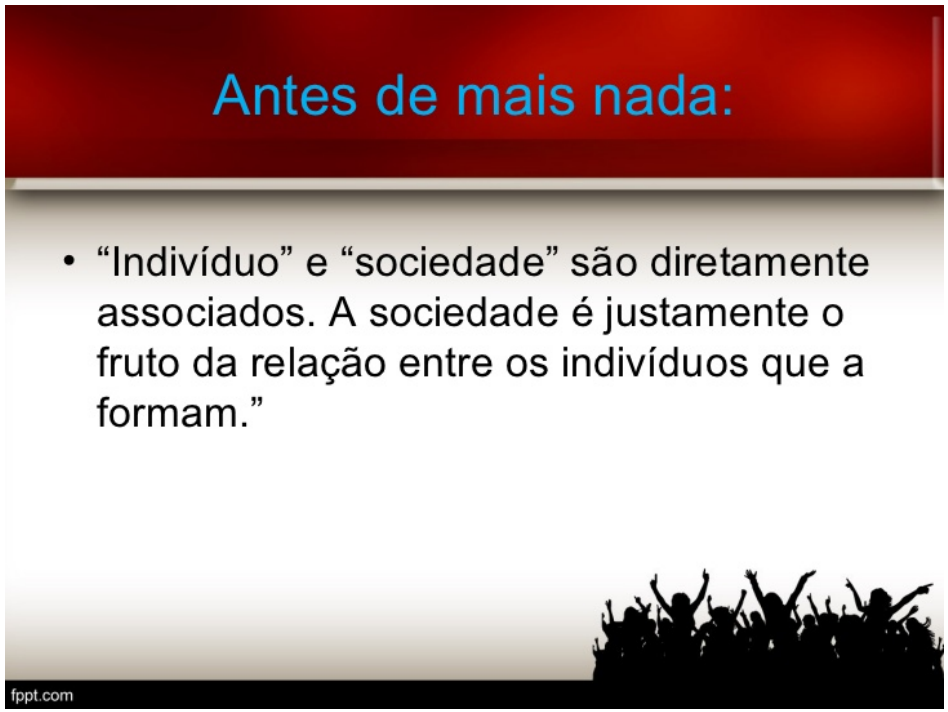


Figure: 3

- Indivíduo x Sociedade. [Google, 2019]

INSTITUTO DE INFORMÁTICA UFG

- No Desenvolvimento de Software...

Figure: 4

- Equipe de Desenvolvimento de Software. [Google, 2019]

INSTITUTO DE INFORMÁTICA UFG

Qualidade de Software

- Nossos Processos tem qualidade?
- Nossos Produtos tem qualidade?

Figure: 5

- Qualidade de Software. [Google, 2019]

INSTITUTO DE INFORMÁTICA UFG

Qualidade de Software

- Segundo a norma brasileira NBR ISO 8402, qualidade é a totalidade das características de uma entidade que lhe confere a capacidade de satisfazer as necessidades explícitas e implícitas.

Qualidade de Software

- Segundo a norma brasileira NBR ISO 8402, qualidade é a totalidade das características de uma entidade que lhe confere a capacidade de satisfazer as necessidades explícitas e implícitas.
- Conquistar um produto de qualidade requer planejamento de atividades que darão suporte na obtenção da mesma.

INSTITUTO DE INFORMÁTICA UFG

Qualidade de Software

- Segundo a norma brasileira NBR ISO 8402, qualidade é a totalidade das características de uma entidade que lhe confere a capacidade de satisfazer as necessidades explícitas e implícitas.
- Conquistar um produto de qualidade requer planejamento de atividades que darão suporte na obtenção da mesma.
- Essa prática requer gerenciamento, documentação, padrões de convenções e métricas, auditorias, relatórios e ferramentas.

INSTITUTO DE INFORMÁTICA UFG

Qualidade de Software

- Segundo a norma brasileira NBR ISO 8402, qualidade é a totalidade das características de uma entidade que lhe confere a capacidade de satisfazer as necessidades explícitas e implícitas.
- Conquistar um produto de qualidade requer planejamento de atividades que darão suporte na obtenção da mesma.
- Essa prática requer gerenciamento, documentação, padrões de convenções e métricas, auditorias, relatórios e ferramentas.
- Obviamente a qualidade se adapta ao estilo do produto e do processo.

INSTITUTO DE INFORMÁTICA UFG

Verificação e Validação

Figure: 6

- Verificação x Validação. [Google, 2019]

Importância do Teste de Software

- Por que o teste é importante?
- Porque objetiva melhorar a qualidade do produto entregue ao nosso cliente, ou seja, objetiva garantir que o produto entregue correspondente às necessidades explicitadas pelo cliente.
- Nem mais, nem menos.

INSTITUTO DE INFORMÁTICA UFG

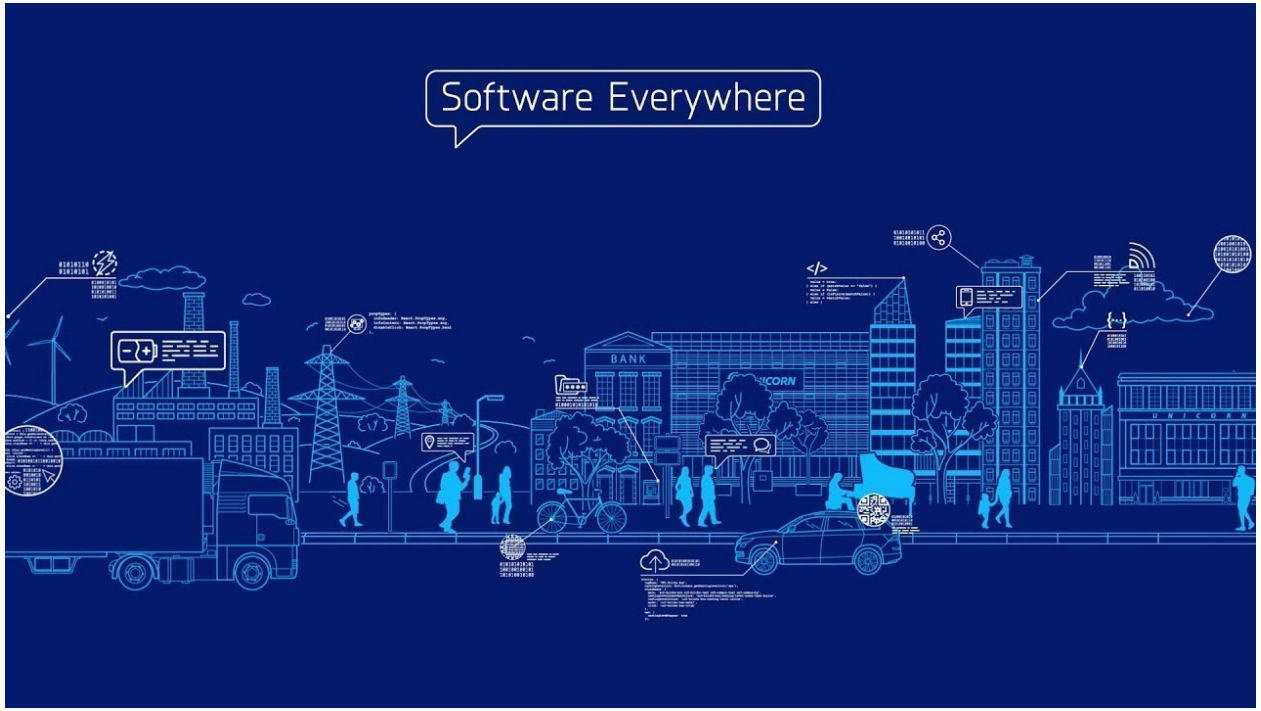


Figure: 7

- Onipresença do Software. [Google, 2019]

INSTITUTO DE INFORMÁTICA UFG

Teste x Debug

- Testar e Debugar é a mesma coisa?
- a Resposta é...
- Não.
- Teste objetiva revelar a presença de defeitos.
- Debug é o processo de encontrar o corrigir os defeitos.

Teste x Homologação

- Testar e homologar é a mesma coisa?
- a Resposta é...
- Não.
- os objetivos são diferentes.
- Teste objetiva revelar a presença de defeitos.
- Homologar objetiva aferir que o funcionamento esteja adequado às necessidades explícitadas.

INSTITUTO DE INFORMÁTICA UFG

Custo do Teste

- Por que é difícil a implantação de um processo de teste?
- Custo.

Custo dos Defeitos

- O site `https://raygun.com/blog/costly-software-errors-history/`, apresenta um relatório atualizado (2019) do ranking dos maiores custos associados a defeitos de software, que serão descritos a seguir:

- Prejuízo de US$ 125 milhões.

Figure: 8

- NASA's Mars Climate Orbiter. [HARLEY, 2019]

INSTITUTO DE INFORMÁTICA UFG

- O Ariane 5 custou cerca de US $ 8 bilhões para ser desenvolvido e carregava uma carga útil de US $ 500 milhões em satélites quando explodiu.

Figure: 9

- Ariane 5 Flight 501. [HARLEY, 2019]

INSTITUTO DE INFORMÁTICA UFG

Therac 25

- O Therac-25 é o nome de uma máquina de radiografia fabricada pela Atomic Energy of Canada (AECL) em 1985.
- Esse dispositivo “assassino” foi responsável pela morte de três pacientes entre 1985 e 1987.
- A causa da morte dessas pessoas foi o envenenamento por radiação.
- O problema estava na quantidade de radiação emitida em seu funcionamento.
- Enquanto um paciente deveria receber cerca de 200 rads, a Therac-25 bombardeava as pessoas com o valor absurdo de 15 mil e 20 mil rads.

INSTITUTO DE INFORMÁTICA UFG

Figure: 10

- Therac 25. [Google, 2019]

O que é teste?

Figure: 11

- ???

O que é teste?

Segundo a norma [IEEE 610.12.1990]

"Teste é o processo de operar um sistema ou componente sob condições específicas, observando e registrando os resultados, avaliando alguns aspectos do sistema ou componente."

INSTITUTO DE INFORMÁTICA UFG

Níveis de Maturidade

- Segundo [Beizer, 1990], são cinco:

Níveis de Maturidade

- Segundo [Beizer, 1990]
- Nível 0 - Não há diferença entre teste e depuração (debugging).

Níveis de Maturidade

- Segundo [Beizer, 1990]
- Nível 0 - Não há diferença entre teste e depuração (debugging).
- Nível 1 - O propósito do teste é mostrar que o software funciona.

Níveis de Maturidade

- Segundo [Beizer, 1990]
- Nível 0 - Não há diferença entre teste e depuração (debugging).
- Nível 1 - O propósito do teste é mostrar que o software funciona.
- Nível 2 - O propósito do teste é mostrar que o software não funciona.

Níveis de Maturidade

- Segundo [Beizer, 1990]
- Nível 0 - Não há diferença entre teste e depuração (debugging).
- Nível 1 - O propósito do teste é mostrar que o software funciona.
- Nível 2 - O propósito do teste é mostrar que o software não funciona.
- Nível 3 - O propósito do teste não é provar nada, mas reduzir o risco de não funcionamento a um valor aceitável.

INSTITUTO DE INFORMÁTICA UFG

Níveis de Maturidade

- Segundo [Beizer, 1990]
- Nível 0 - Não há diferença entre teste e depuração (debugging).
- Nível 1 - O propósito do teste é mostrar que o software funciona.
- Nível 2 - O propósito do teste é mostrar que o software não funciona.
- Nível 3 - O propósito do teste não é provar nada, mas reduzir o risco de não funcionamento a um valor aceitável.
- Nível 4 - Teste não é uma ação, mas sim uma disciplina mental (institucionalizada na empresa) que resulta em software de baixo risco sem que seja empregado muito esforço de teste.

INSTITUTO DE INFORMÁTICA UFG

Taxonomia

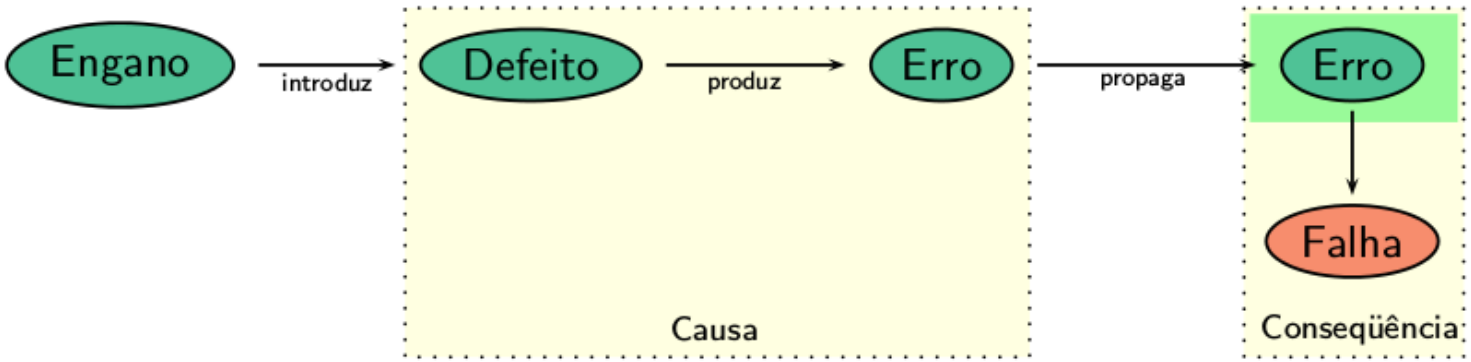


Figure: 12

- [Google, 2019]

Taxonomia

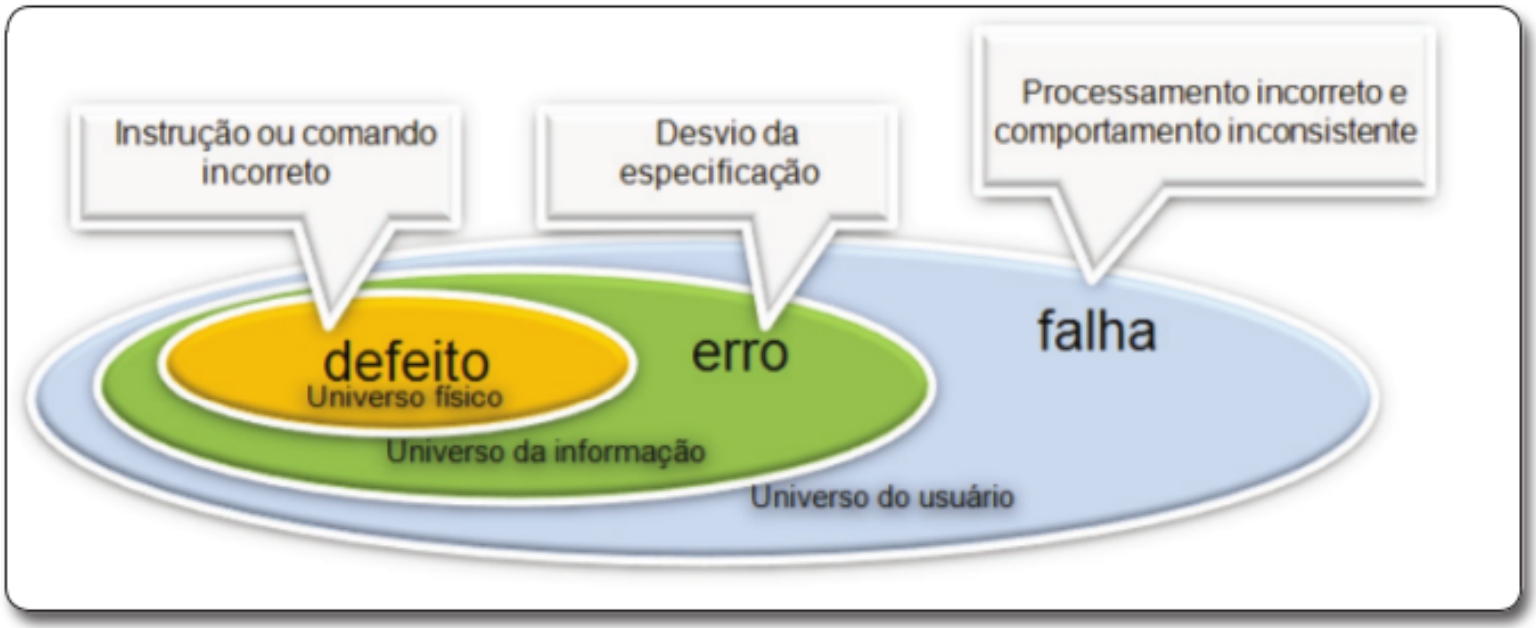


Figure: 13

- [NETO, 2009]

Dado de Teste

- Qualquer valor definido como entrada ou saída para um caso de teste qualquer.

Caso de Teste

- Especificação de um teste. Pode ser textual, script em liguagem de programção, etc.
- Deve conter:
  1. Identificador;
  2. Valor de Entrada;
  3. Pré-condição;
  4. Roteiro ou procedimento de execução;
  5. Resultado esperado.

Critério de Teste

- serve para selecionar e avaliar casos de teste de forma a aumentar as possibilidades de provocar falhas ou, quando isso não ocorre, estabelecer um nível elevado de confiança na correção do produto.

Técnica de Teste

- Forma de se definir e conduzir os testes. São definidas a partir da fonte da informação obtida para se elaborar o teste.

Por que Testar?

Figure: 14

- ???

Por que Testar?

PORQUE NÃO EXISTE SOFTWARE LIVRE DE DEFEITO.

Figure: 15

- [Google, 2019]

INSTITUTO DE INFORMÁTICA UFG

Qual a Origem dos Defeitos?

Figure: 16

- ???

Origem dos Defeitos

- **Erros de Construção**: (falha para satisfazer a especificação através de erros na implementação);
- **Erros de Especificação**: (falha para escrever uma especificação que corretamente represente o projeto);
- **Erros de Projeto**: (falha para satisfazer o entendimento de um Requisito);
- **Erros de Requisitos**: (falha para satisfazer um requisito real).

INSTITUTO DE INFORMÁTICA UFG

Por que Testar?

OS DEFEITOS ESTÃO POR TODA PARTE.

Figure: 17

- [Google, 2019]

INSTITUTO DE INFORMÁTICA UFG

Eficácia do Teste

Independentemente do tipo ou da origem dos defeitos, nossos testes devem ser capazes de revelar a presenạ dos mesmos.

Figure: 18

- [Google, 2019]

Paradoxo do Pesticida [Beizer, 1990]

Figure: 19

- [Google, 2019]

Objetivo do Teste

Figure: 20

- ???

Objetivo do Teste

Revelar a presença de defeitos nos nossos softwares.

Figure: 21

- [Google, 2019]

INSTITUTO DE INFORMÁTICA UFG

Quando Testar?

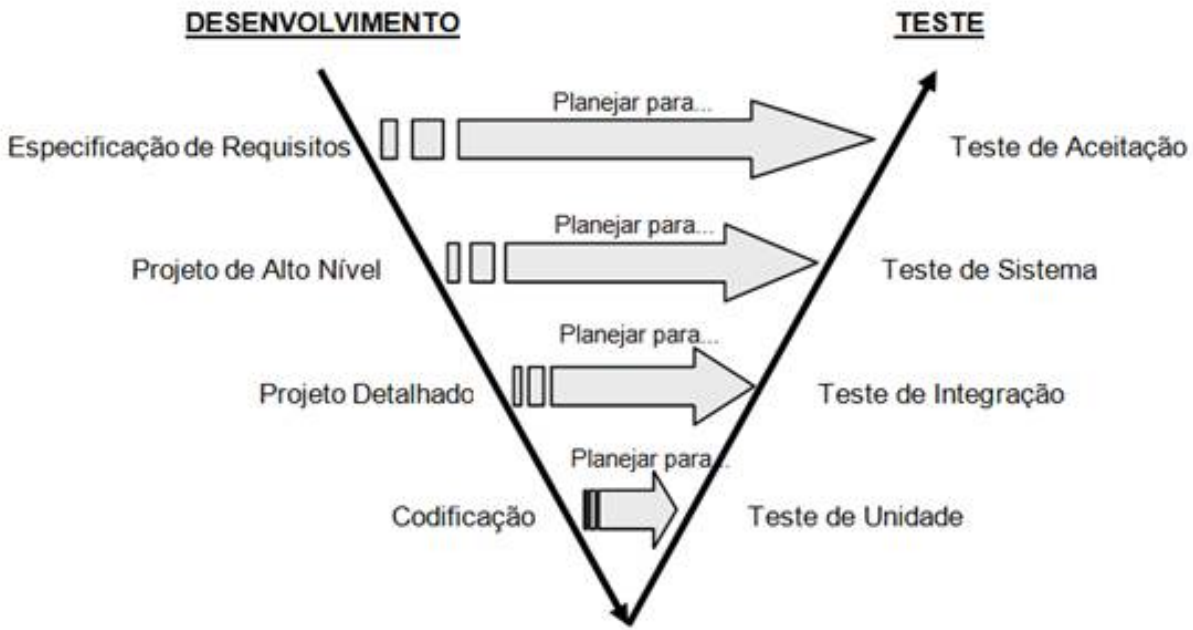


Figure: 22

- [Google, 2019]

INSTITUTO DE INFORMÁTICA UFG

Efetividade dos Níveis de Teste

- Os testes de unidades executados podem reduzir entre 30% e 50% dos defeitos dos softwares.
- O processo de revisão de código permite diminuir entre 20% e 30% os defeitos.
- O processo de testes de sistemas pode remover de 30% a 50% os defeitos remanescentes.

INSTITUTO DE INFORMÁTICA UFG

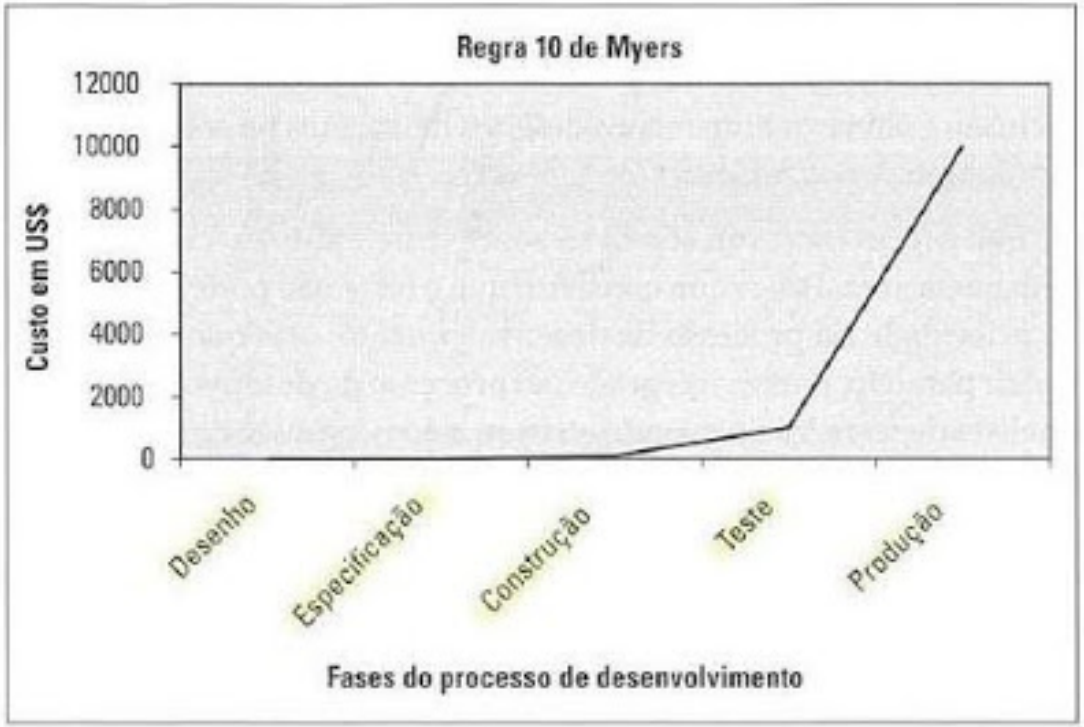


Figure: 23

- Regra 10 de Myers. [Google, 2019]

Níveis de Teste

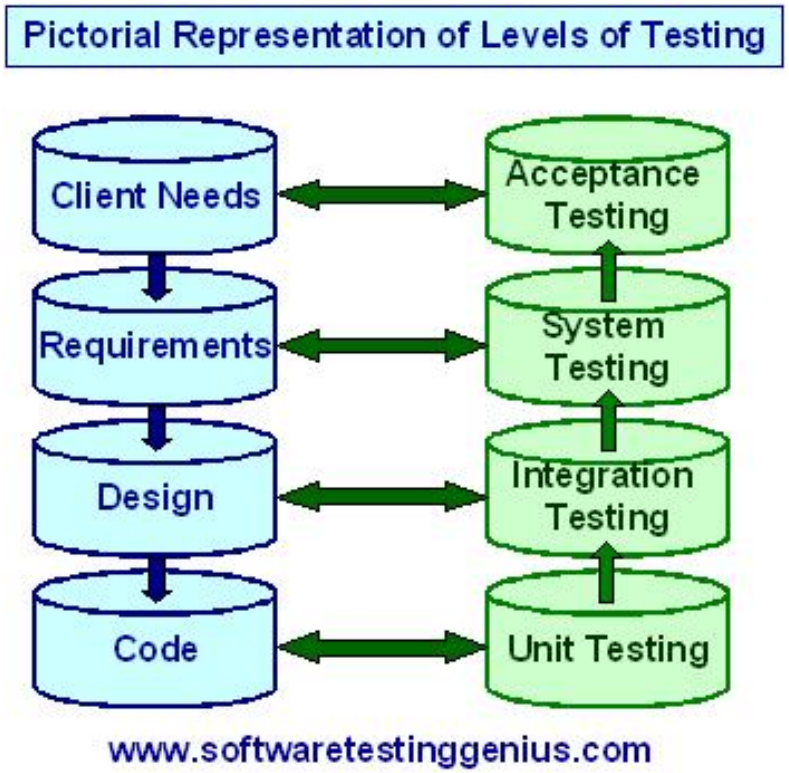


Figure: 24

- [Google, 2019]

Tipos de Teste

- Diferentes tipos de testes podem ser utilizados para verificar se um programa se comporta como o especificado.
- Basicamente, os testes podem ser classificados em:
  - Teste Funcional (**teste caixa-preta**);
  - Teste Estrutural (**teste caixa-branca**) e
  - Teste Baseado em Defeitos (**Testes de Mutantes**).

INSTITUTO DE INFORMÁTICA UFG

Tipos de Teste

- A técnica de teste é definida pelo tipo de informação utilizada para realizar o teste.

Técnica de Teste Funcional

- **Técnica caixa-preta** - os testes são baseados exclusivamente na especificação de requisitos do programa. Nenhum conhecimento de como o programa está implementado é requerido.

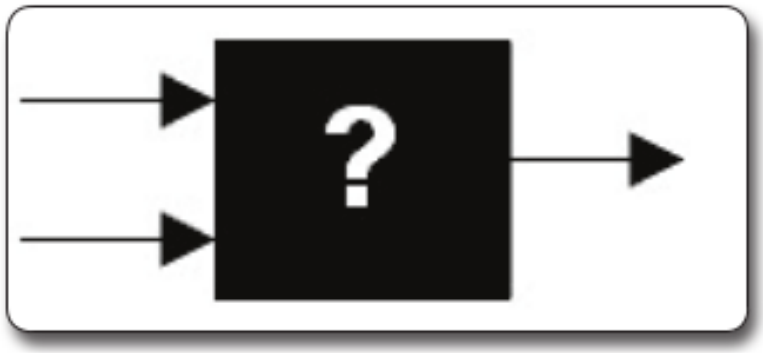


Figure: 25

- Técnica de Teste Funcional. [NETO, 2009]

INSTITUTO DE INFORMÁTICA UFG

Técnica de Teste Estrutural

- **Técnica caixa-branca** - os testes são baseados na estrutura interna do programa, ou seja, na implementação do mesmo.

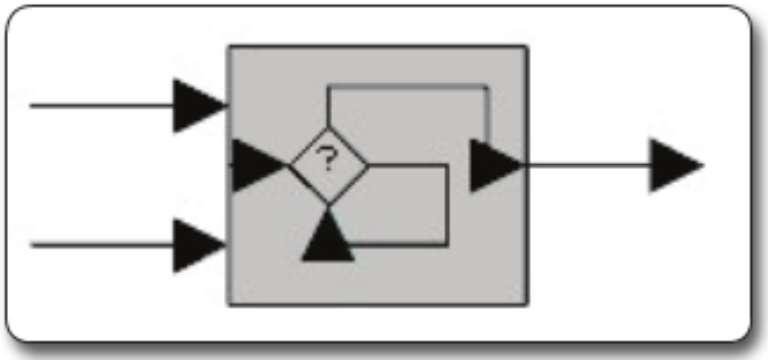


Figure: 26

- Técnica de Teste Estrutural. [NETO, 2009]

INSTITUTO DE INFORMÁTICA UFG

Técnica de Teste Baseada em Defeitos

- **Técnica baseada em defeitos** - os testes são baseados em informações históricas sobre defeitos cometidos frequuentemente durante o processo de desenvolvimento de software.

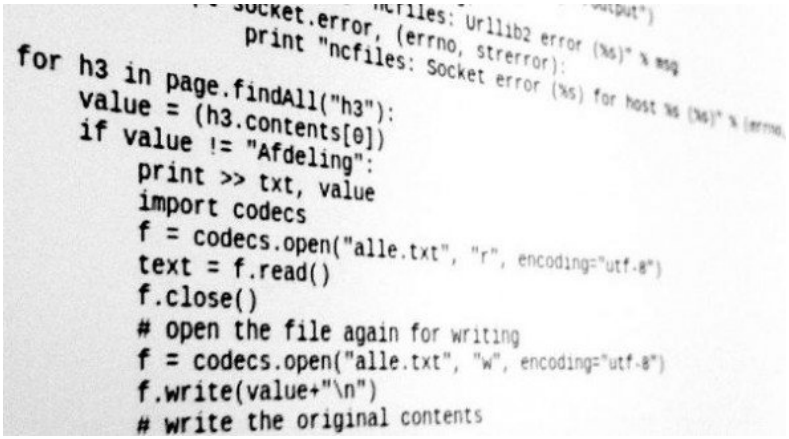


Figure: 27

- Teste de Mutantes. [NETO, 2009]

Variações de Tipos de Teste

Teste de Performance

Modelagem do uso esperado do sistema, pela simulação do acesso de vários usuários concorrentemente.

Figure: 28

- Teste de Performance. [Google, 2019]

Variações de Tipos de Teste

Teste de Carga

Testar o comportamento do sistema submetendo-o ao processamento de um grande volume de dados.

Figure: 29

INSTITUTO DE INFORMÁTICA UFG

Variações de Tipos de Teste

Teste de Estresse

Determinar a estabilidade de um dado sistema para além da sua capacidade normal de operação.

Figure: 30

- Teste de Estresse. [Google, 2019]

INSTITUTO DE INFORMÁTICA UFG

Variações de Tipos de Teste

Teste de Confiabilidade

Determinar por quanto tempo o sistema é capaz de sustentar uma performance ótima.

Figure: 31

INSTITUTO DE INFORMÁTICA UFG

Variações de Tipos de Teste

Teste de Regressão

1. Os testes de regressão geralmente são executados após a correção de algum defeito ou após a adição de uma nova funcionalidade.

INSTITUTO DE INFORMÁTICA UFG

Variações de Tipos de Teste

Teste de Regressão

1. Os testes de regressão geralmente são executados após a correção de algum defeito ou após a adição de uma nova funcionalidade.
2. Seu objetivo é garantir que nenhum defeito foi acrescentado ao sistema após sua modificação.

INSTITUTO DE INFORMÁTICA UFG

Variações de Tipos de Teste

Teste de Regressão

1. Os testes de regressão geralmente são executados após a correção de algum defeito ou após a adição de uma nova funcionalidade.
2. Seu objetivo é garantir que nenhum defeito foi acrescentado ao sistema após sua modificação.
3. De nada adianta testar um sistema, verificar que ele não possui defeitos para aquele conjunto de casos de teste e após modificações no sistema, aqueles casos de teste não serem novamente executados, pois as novas mudanças podem trazer defeitos para o sistema.

INSTITUTO DE INFORMÁTICA UFG

Variações de Tipos de Teste

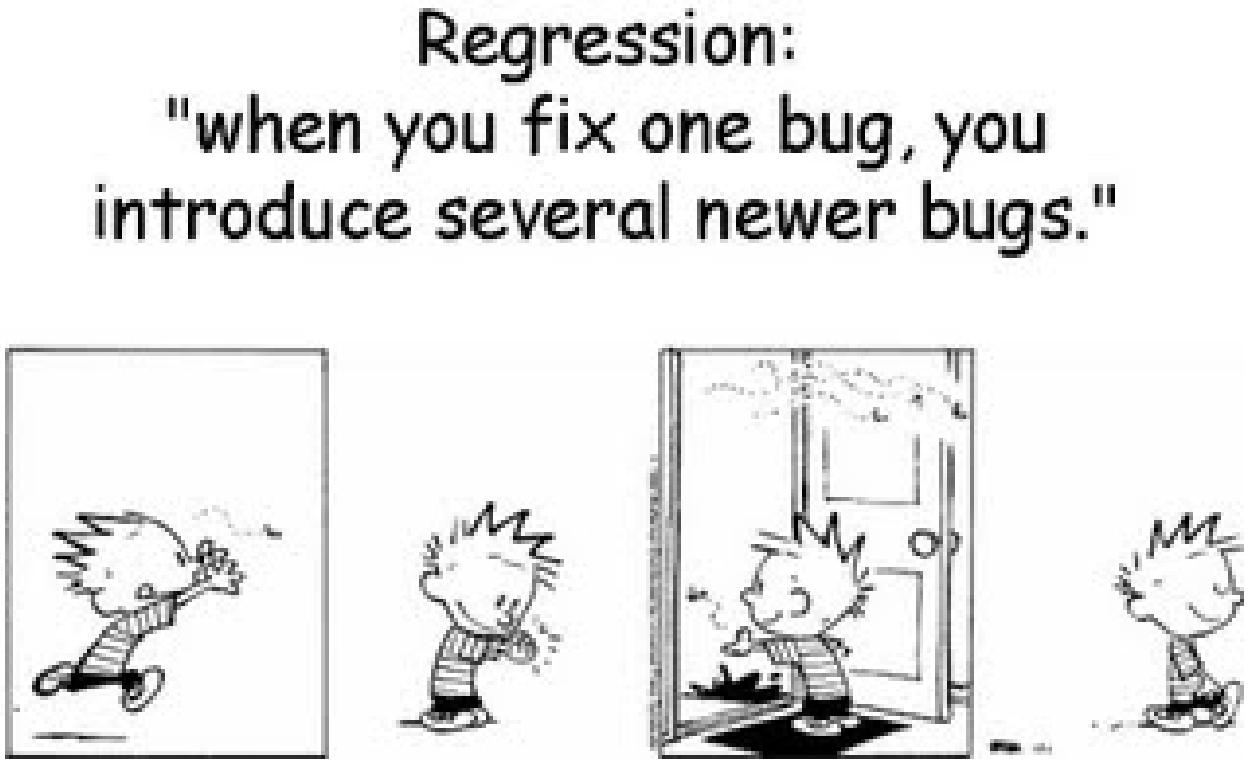


Figure: 32

- Teste de Regressão. [Google, 2019]

Desafios

- Alguém já testou algum produto de software?

Desafios

- Alguém já testou algum produto de software?
- Quais foram os maiores desafios?

Desafios

- Alguém já testou algum produto de software?
- Quais foram os maiores desafios?
- Alguns problemas comuns são:

Desafios

- Alguém já testou algum produto de software?
- Quais foram os maiores desafios?
- Alguns problemas comuns são:
  1. Não há tempo para o teste exaustivo.

Desafios

- Alguém já testou algum produto de software?
- Quais foram os maiores desafios?
- Alguns problemas comuns são:
  1. Não há tempo para o teste exaustivo.
  2. Muitas combinações de entrada para serem exercitadas.

INSTITUTO DE INFORMÁTICA UFG

Desafios

- Alguém já testou algum produto de software?
- Quais foram os maiores desafios?
- Alguns problemas comuns são:
  1. Não há tempo para o teste exaustivo.
  2. Muitas combinações de entrada para serem exercitadas.
  3. Dificuldade em determinar os resultados esperados para cada caso de teste.

INSTITUTO DE INFORMÁTICA UFG

Desafios

- Alguém já testou algum produto de software?
- Quais foram os maiores desafios?
- Alguns problemas comuns são:
  1. Não há tempo para o teste exaustivo.
  2. Muitas combinações de entrada para serem exercitadas.
  3. Dificuldade em determinar os resultados esperados para cada caso de teste.
  4. Requisitos do software inexistentes ou que mudam rapidamente.

INSTITUTO DE INFORMÁTICA UFG

Desafios

- Alguém já testou algum produto de software?
- Quais foram os maiores desafios?
- Alguns problemas comuns são:
  1. Não há tempo para o teste exaustivo.
  2. Muitas combinações de entrada para serem exercitadas.
  3. Dificuldade em determinar os resultados esperados para cada caso de teste.
  4. Requisitos do software inexistentes ou que mudam rapidamente.
  5. Não há tempo suficiente para o teste.

INSTITUTO DE INFORMÁTICA UFG

Desafios

- Alguém já testou algum produto de software?
- Quais foram os maiores desafios?
- Alguns problemas comuns são:
  1. Não há tempo para o teste exaustivo.
  2. Muitas combinações de entrada para serem exercitadas.
  3. Dificuldade em determinar os resultados esperados para cada caso de teste.
  4. Requisitos do software inexistentes ou que mudam rapidamente.
  5. Não há tempo suficiente para o teste.
  6. Não há treinamento no processo de teste.

INSTITUTO DE INFORMÁTICA UFG

Desafios

- Alguém já testou algum produto de software?
- Quais foram os maiores desafios?
- Alguns problemas comuns são:
  1. Não há tempo para o teste exaustivo.
  2. Muitas combinações de entrada para serem exercitadas.
  3. Dificuldade em determinar os resultados esperados para cada caso de teste.
  4. Requisitos do software inexistentes ou que mudam rapidamente.
  5. Não há tempo suficiente para o teste.
  6. Não há treinamento no processo de teste.
  7. Não há ferramenta de apoio.

INSTITUTO DE INFORMÁTICA UFG

Desafios

- Alguém já testou algum produto de software?
- Quais foram os maiores desafios?
- Alguns problemas comuns são:
  1. Não há tempo para o teste exaustivo.
  2. Muitas combinações de entrada para serem exercitadas.
  3. Dificuldade em determinar os resultados esperados para cada caso de teste.
  4. Requisitos do software inexistentes ou que mudam rapidamente.
  5. Não há tempo suficiente para o teste.
  6. Não há treinamento no processo de teste.
  7. Não há ferramenta de apoio.
  8. Gerentes que desconhecem teste ou que não se preocupam com qualidade.

INSTITUTO DE INFORMÁTICA UFG

Pratica 1

- Elaborar um conjunto de casos de testes suficientes para um programa com a seguinte especificação:
- O programa deve determinar se um identificador é válido ou não. Um identificador válido deve começar com uma letra e conter apenas letras ou dígitos. Além disso, deve ter no mínimo um caractere e no máximo seis caracteres de comprimento.

INSTITUTO DE INFORMÁTICA UFG

Pratica 2

- Avaliação Escolar.

Processo de Teste Caixa Preta

- Os passos básicos para se aplicar um critério de teste caixa preta são os seguintes:

Processo de Teste Caixa Preta

- Os passos básicos para se aplicar um critério de teste caixa preta são os seguintes:
  1. A especificação de requisitos é analisada.

Processo de Teste Caixa Preta

- Os passos básicos para se aplicar um critério de teste caixa preta são os seguintes:
  1. A especificação de requisitos é analisada.
  2. Entradas válidas são escolhidas (com base na especificação) para determinar se o produto em teste se comporta corretamente.

INSTITUTO DE INFORMÁTICA UFG

Processo de Teste Caixa Preta

- Os passos básicos para se aplicar um critério de teste caixa preta são os seguintes:
  1. A especificação de requisitos é analisada.
  2. Entradas válidas são escolhidas (com base na especificação) para determinar se o produto em teste se comporta corretamente.
  3. Entradas inválidas também são escolhidas para verificar se são detectadas e manipuladas adequadamente.

INSTITUTO DE INFORMÁTICA UFG

Processo de Teste Caixa Preta

- Os passos básicos para se aplicar um critério de teste caixa preta são os seguintes:
  1. A especificação de requisitos é analisada.
  2. Entradas válidas são escolhidas (com base na especificação) para determinar se o produto em teste se comporta corretamente.
  3. Entradas inválidas também são escolhidas para verificar se são detectadas e manipuladas adequadamente.
  4. As saídas esperadas para as entradas escolhidas são determinadas.

INSTITUTO DE INFORMÁTICA UFG

Processo de Teste Caixa Preta

- Os passos básicos para se aplicar um critério de teste caixa preta são os seguintes:
  1. A especificação de requisitos é analisada.
  2. Entradas válidas são escolhidas (com base na especificação) para determinar se o produto em teste se comporta corretamente.
  3. Entradas inválidas também são escolhidas para verificar se são detectadas e manipuladas adequadamente.
  4. As saídas esperadas para as entradas escolhidas são determinadas.
  5. Os casos de testes são construídos.

INSTITUTO DE INFORMÁTICA UFG

Processo de Teste Caixa Preta

- Os passos básicos para se aplicar um critério de teste caixa preta são os seguintes:
  1. A especificação de requisitos é analisada.
  2. Entradas válidas são escolhidas (com base na especificação) para determinar se o produto em teste se comporta corretamente.
  3. Entradas inválidas também são escolhidas para verificar se são detectadas e manipuladas adequadamente.
  4. As saídas esperadas para as entradas escolhidas são determinadas.
  5. Os casos de testes são construídos.
  6. O conjunto de teste é executado.

INSTITUTO DE INFORMÁTICA UFG

Processo de Teste Caixa Preta

- Os passos básicos para se aplicar um critério de teste caixa preta são os seguintes:
  1. A especificação de requisitos é analisada.
  2. Entradas válidas são escolhidas (com base na especificação) para determinar se o produto em teste se comporta corretamente.
  3. Entradas inválidas também são escolhidas para verificar se são detectadas e manipuladas adequadamente.
  4. As saídas esperadas para as entradas escolhidas são determinadas.
  5. Os casos de testes são construídos.
  6. O conjunto de teste é executado.
  7. As saídas obtidas são comparadas com as saídas esperadas.

INSTITUTO DE INFORMÁTICA UFG

Processo de Teste Caixa Preta

- Os passos básicos para se aplicar um critério de teste caixa preta são os seguintes:
  1. A especificação de requisitos é analisada.
  2. Entradas válidas são escolhidas (com base na especificação) para determinar se o produto em teste se comporta corretamente.
  3. Entradas inválidas também são escolhidas para verificar se são detectadas e manipuladas adequadamente.
  4. As saídas esperadas para as entradas escolhidas são determinadas.
  5. Os casos de testes são construídos.
  6. O conjunto de teste é executado.
  7. As saídas obtidas são comparadas com as saídas esperadas.
  8. Um relatório é gerado para avaliar o resultado dos testes.

INSTITUTO DE INFORMÁTICA UFG

Critérios de Teste Funcional

1. Particionamento de Equivalência (*Equivalence Partition*).

Critérios de Teste Funcional

1. Particionamento de Equivalência (*Equivalence Partition*).
2. Análise do Valor Limite (*Boundary Value Analysis*).

INSTITUTO DE INFORMÁTICA UFG

Critérios de Teste Funcional

1. Particionamento de Equivalência (*Equivalence Partition*).
2. Análise do Valor Limite (*Boundary Value Analysis*).
3. Tabela de Decisão (*Decision Table*).

INSTITUTO DE INFORMÁTICA UFG

Critérios de Teste Funcional

1. Particionamento de Equivalência (*Equivalence Partition*).
2. Análise do Valor Limite (*Boundary Value Analysis*).
3. Tabela de Decisão (*Decision Table*).
4. Grafo de Causa e Efeito (*Cause-Effect Graph*).

INSTITUTO DE INFORMÁTICA UFG

Critérios de Teste Funcional

1. Particionamento de Equivalência (*Equivalence Partition*).
2. Análise do Valor Limite (*Boundary Value Analysis*).
3. Tabela de Decisão (*Decision Table*).
4. Grafo de Causa e Efeito (*Cause-Effect Graph*).
5. Teste de Todos os Pares (*Pairwise Testing*).

INSTITUTO DE INFORMÁTICA UFG

Critérios de Teste Funcional

1. Particionamento de Equivalência (*Equivalence Partition*).
2. Análise do Valor Limite (*Boundary Value Analysis*).
3. Tabela de Decisão (*Decision Table*).
4. Grafo de Causa e Efeito (*Cause-Effect Graph*).
5. Teste de Todos os Pares (*Pairwise Testing*).
6. Teste de Transição de Estado (*State-Transition Testing*).

INSTITUTO DE INFORMÁTICA UFG

Critérios de Teste Funcional

1. Particionamento de Equivalência (*Equivalence Partition*).
2. Análise do Valor Limite (*Boundary Value Analysis*).
3. Tabela de Decisão (*Decision Table*).
4. Grafo de Causa e Efeito (*Cause-Effect Graph*).
5. Teste de Todos os Pares (*Pairwise Testing*).
6. Teste de Transição de Estado (*State-Transition Testing*).
7. Teste de Análise de Domínio (*Domain Analysis Testing*).

INSTITUTO DE INFORMÁTICA UFG

Critérios de Teste Funcional

1. Particionamento de Equivalência (*Equivalence Partition*).
2. Análise do Valor Limite (*Boundary Value Analysis*).
3. Tabela de Decisão (*Decision Table*).
4. Grafo de Causa e Efeito (*Cause-Effect Graph*).
5. Teste de Todos os Pares (*Pairwise Testing*).
6. Teste de Transição de Estado (*State-Transition Testing*).
7. Teste de Análise de Domínio (*Domain Analysis Testing*).
8. Teste Funcional Sistemático (*Systematic Functional Testing*).

INSTITUTO DE INFORMÁTICA UFG

Critérios de Teste Funcional

1. Particionamento de Equivalência (*Equivalence Partition*).
2. Análise do Valor Limite (*Boundary Value Analysis*).
3. Tabela de Decisão (*Decision Table*).
4. Grafo de Causa e Efeito (*Cause-Effect Graph*).
5. Teste de Todos os Pares (*Pairwise Testing*).
6. Teste de Transição de Estado (*State-Transition Testing*).
7. Teste de Análise de Domínio (*Domain Analysis Testing*).
8. Teste Funcional Sistemático (*Systematic Functional Testing*).
9. Teste de Caso de Uso (*Use Case Testing*).

INSTITUTO DE INFORMÁTICA UFG

Particionamento Por Equivalência

1. Consiste na divisão do domínio de entrada e saída em intervalos.

Particionamento Por Equivalência

1. Consiste na divisão do domínio de entrada e saída em intervalos.
2. Qualquer valor dentro do intervalo tem a mesma importância, ou seja, qualquer valor escolhido é adequado.

INSTITUTO DE INFORMÁTICA UFG

Particionamento Por Equivalência

1. Consiste na divisão do domínio de entrada e saída em intervalos.
2. Qualquer valor dentro do intervalo tem a mesma importância, ou seja, qualquer valor escolhido é adequado.
3. O mesmo se aplica para os demais intervalos de dados.

INSTITUTO DE INFORMÁTICA UFG

Particionamento Por Equivalência

1. Consiste na divisão do domínio de entrada e saída em intervalos.
2. Qualquer valor dentro do intervalo tem a mesma importância, ou seja, qualquer valor escolhido é adequado.
3. O mesmo se aplica para os demais intervalos de dados.
4. Tais intervalos determinam o que é chamado de **classe de equivalência**.

INSTITUTO DE INFORMÁTICA UFG

Particionamento Por Equivalência

1. Consiste na divisão do domínio de entrada e saída em intervalos.
2. Qualquer valor dentro do intervalo tem a mesma importância, ou seja, qualquer valor escolhido é adequado.
3. O mesmo se aplica para os demais intervalos de dados.
4. Tais intervalos determinam o que é chamado de **classe de equivalência**.
5. Qualquer valor no intervalo de uma classe é considerado equivalente em termos de teste. Assim sendo:

INSTITUTO DE INFORMÁTICA UFG

Particionamento Por Equivalência

1. Consiste na divisão do domínio de entrada e saída em intervalos.
2. Qualquer valor dentro do intervalo tem a mesma importância, ou seja, qualquer valor escolhido é adequado.
3. O mesmo se aplica para os demais intervalos de dados.
4. Tais intervalos determinam o que é chamado de **classe de equivalência**.
5. Qualquer valor no intervalo de uma classe é considerado equivalente em termos de teste. Assim sendo:
   - Se um caso de teste de uma classe de equivalência revela um defeito, qualquer caso de teste da mesma classe também revelaria e vice-versa.

INSTITUTO DE INFORMÁTICA UFG

Particionamento por Equivalência (*Equivalence Partition*)

- Conjunto de Valores

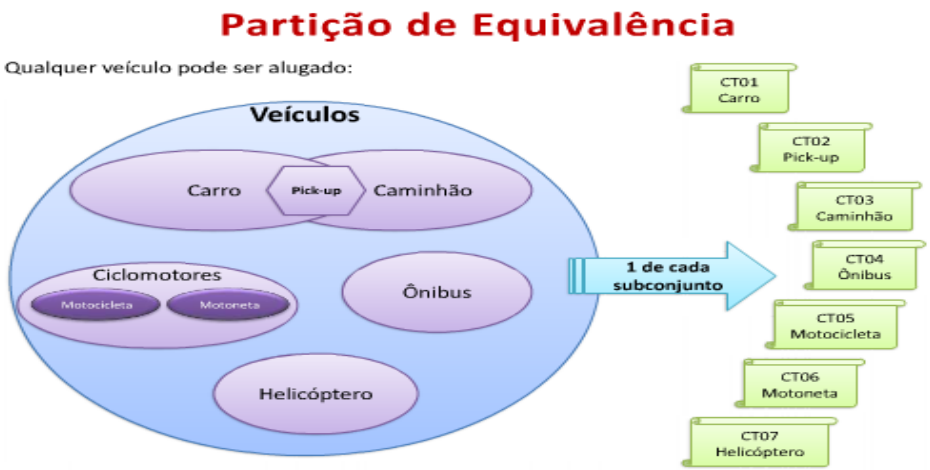


Figure: 33

- PCE. [Google, 2019]

- Intervalo de Valores

Figure: 34

- PCE. [Google, 2019]

INSTITUTO DE INFORMÁTICA UFG

Particionamento por Equivalência - Diretrizes

- Indenpendentemente do tipo do atributo:
  - Atributo - Obrigatório (S/N):
    - Definir uma classe válida (atributo informado);
    - Definir uma classe inválida (atributo não informado).

Particionamento por Equivalência - Diretrizes

- Atributos do tipo String:
  - Acentuação gráfica / Caracteres especiais (sim ou não)
    - Definir uma classe para o “sim”;
    - Definir uma classe para o “não”
  - Tamanho mínimo / Tamanho máximo
    - Definir uma classe inválida para valor inferior ao mínimo;
    - Definir uma classe válida para valor dentro do intervalo entre o mínimo e o máximo.
    - Definir uma classe inválida para valor superior do máximo.

INSTITUTO DE INFORMÁTICA UFG

Particionamento por Equivalência - Diretrizes

- Atributos de tipos numéricos
  - Valores Inteiros:
    - Definir uma classe inválida para valores inferiores ao limite inferior;
    - Definir uma classe válida para valores dentro do intervalo entre os limites;
    - Definir uma classe inválida para valores acima do limite superior.
  - Valores de ponto-flutuante:
    - Definir três classes de equivalência para a quantidade de casas depois da vírgula.

Particionamento por Equivalência - Diretrizes

- Atributos do tipo lógico
  - Definir duas classes de equivalência.
- Atributos do tipo Data:
  - Definir classes de equivalência para:
    - Data Válida e inválida;
    - Formato
    - Data inferior e data superior.

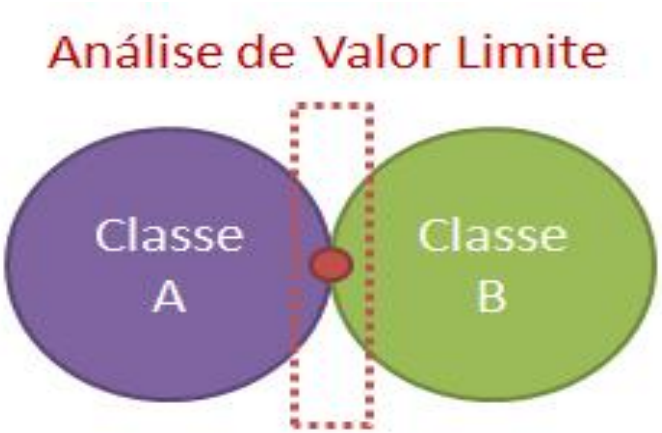
Prática 1 - Classes de Equivalência

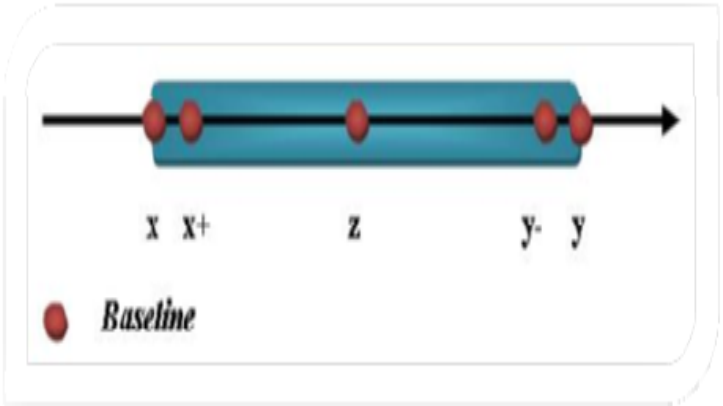
Condições de Entrada	Classes Válidas	Classes Inválidas	
Tamanho t do identificador	$1 \leq t \leq 6$ (1)	$t < 1$ (2)	$t > 6$ (3)
Primeiro caractere c é uma letra	Sim (4)	Não (5)	
Só contém caracteres válidos	Sim (6)	Não (7)	

INSTITUTO DE INFORMÁTICA UFG

Critérios de Teste Caixa Preta

Análise do Valor Limite (*Boundary Value Analysis*).





INSTITUTO DE INFORMÁTICA UFG

Teste Funcional e as Fases de Teste

- Por ser independente da implementação, critérios da técnica caixa preta podem ser utilizados em todas as fases de teste.

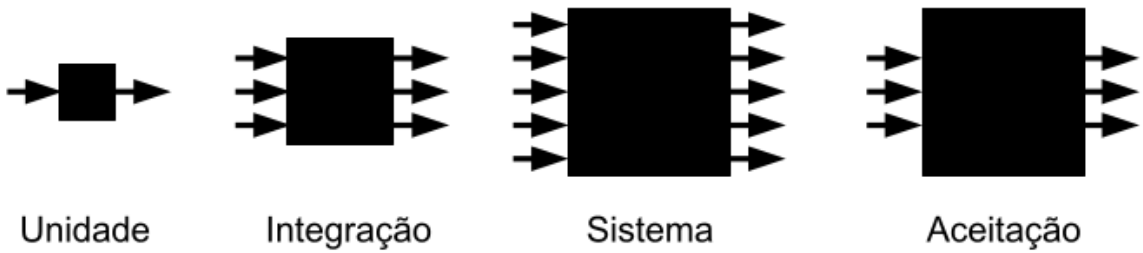
Teste Funcional e as Fases de Teste

- Por ser independente da implementação, critérios da técnica caixa preta podem ser utilizados em todas as fases de teste.
- A medida que se move do teste de unidade para o teste de sistema, entradas e saídas mais complexas são exigidas, mas a abordagem permanece a mesma.

INSTITUTO DE INFORMÁTICA UFG

Teste Funcional e as Fases de Teste

- Por ser independente da implementação, critérios da técnica caixa preta podem ser utilizados em todas as fases de teste.
- A medida que se move do teste de unidade para o teste de sistema, entradas e saídas mais complexas são exigidas, mas a abordagem permanece a mesma.
- Além disso, principalmente no nível de sistema, o teste caixa-preta é de fundamental importância uma vez que a estrutura interna do sistema é muito complexa, inviabilizando o teste caixa-branca.



INSTITUTO DE INFORMÁTICA UFG

Vantagens do Teste Funcional

- Pode ser utilizado em todas as fases de teste.

Vantagens do Teste Funcional

- Pode ser utilizado em todas as fases de teste.
- Independente do paradigma de programação utilizado.

Vantagens do Teste Funcional

- Pode ser utilizado em todas as fases de teste.
- Independente do paradigma de programação utilizado.
- Teste funcional sistemático direciona o testador a escolher subconjuntos de teste que são eficientes e efetivos em identificar defeitos. Melhor que o teste aleatório [**?**].

INSTITUTO DE INFORMÁTICA UFG

Vantagens do Teste Funcional

- Pode ser utilizado em todas as fases de teste.
- Independente do paradigma de programação utilizado.
- Teste funcional sistemático direciona o testador a escolher subconjuntos de teste que são eficientes e efetivos em identificar defeitos. Melhor que o teste aleatório [**?**].
- Eficaz em detectar determinados tipos de defeitos:

INSTITUTO DE INFORMÁTICA UFG

Vantagens do Teste Funcional

- Pode ser utilizado em todas as fases de teste.
- Independente do paradigma de programação utilizado.
- Teste funcional sistemático direciona o testador a escolher subconjuntos de teste que são eficientes e efetivos em identificar defeitos. Melhor que o teste aleatório [**?**].
- Eficaz em detectar determinados tipos de defeitos:
  - Funcionalidade ausente, por exemplo.

INSTITUTO DE INFORMÁTICA UFG

Desvantagens do Teste Funcional

- Dependente de uma boa especificação de requisitos.

Desvantagens do Teste Funcional

- Dependente de uma boa especificação de requisitos.
- Não permite determinar que partes essenciais ou críticas da implementação do software tenham sido executadas.

Conceitos Básicos

- Técnica Caixa Branca oposto da Técnica Caixa Preta.

Conceitos Básicos

- Técnica Caixa Branca oposto da Técnica Caixa Preta.
  - Baseia-se nos caminhos internos, estrutura e implementação do produto em teste.

Conceitos Básicos

- Técnica Caixa Branca oposto da Técnica Caixa Preta.
  - Baseia-se nos caminhos internos, estrutura e implementação do produto em teste.
  - Requer conhecimento do código do produto em teste para ser aplicada.

INSTITUTO DE INFORMÁTICA UFG

Processo do Teste Caixa Branca

- Os passos básicos para se aplicar um critério de teste caixa branca são os seguintes:

Processo do Teste Caixa Branca

- Os passos básicos para se aplicar um critério de teste caixa branca são os seguintes:
  1. A implementação do produto em teste é analisada.

Processo do Teste Caixa Branca

- Os passos básicos para se aplicar um critério de teste caixa branca são os seguintes:
  1. A implementação do produto em teste é analisada.
  2. Caminhos através da implementação são escolhidos.

INSTITUTO DE INFORMÁTICA UFG

Processo do Teste Caixa Branca

- Os passos básicos para se aplicar um critério de teste caixa branca são os seguintes:
  1. A implementação do produto em teste é analisada.
  2. Caminhos através da implementação são escolhidos.
  3. Valores de entradas são selecionados de modo que os caminhos selecionados sejam executados.

INSTITUTO DE INFORMÁTICA UFG

Processo do Teste Caixa Branca

- Os passos básicos para se aplicar um critério de teste caixa branca são os seguintes:
  1. A implementação do produto em teste é analisada.
  2. Caminhos através da implementação são escolhidos.
  3. Valores de entradas são selecionados de modo que os caminhos selecionados sejam executados.
  4. As saídas esperadas para as entradas escolhidas são determinadas.

INSTITUTO DE INFORMÁTICA UFG

Processo do Teste Caixa Branca

- Os passos básicos para se aplicar um critério de teste caixa branca são os seguintes:
  1. A implementação do produto em teste é analisada.
  2. Caminhos através da implementação são escolhidos.
  3. Valores de entradas são selecionados de modo que os caminhos selecionados sejam executados.
  4. As saídas esperadas para as entradas escolhidas são determinadas.
  5. Os casos de testes são construídos.

INSTITUTO DE INFORMÁTICA UFG

Processo do Teste Caixa Branca

- Os passos básicos para se aplicar um critério de teste caixa branca são os seguintes:
  1. A implementação do produto em teste é analisada.
  2. Caminhos através da implementação são escolhidos.
  3. Valores de entradas são selecionados de modo que os caminhos selecionados sejam executados.
  4. As saídas esperadas para as entradas escolhidas são determinadas.
  5. Os casos de testes são construídos.
  6. Os casos de testes são executados.

INSTITUTO DE INFORMÁTICA UFG

Processo do Teste Caixa Branca

- Os passos básicos para se aplicar um critério de teste caixa branca são os seguintes:
  1. A implementação do produto em teste é analisada.
  2. Caminhos através da implementação são escolhidos.
  3. Valores de entradas são selecionados de modo que os caminhos selecionados sejam executados.
  4. As saídas esperadas para as entradas escolhidas são determinadas.
  5. Os casos de testes são construídos.
  6. Os casos de testes são executados.
  7. As saídas obtidas são comparadas com as saídas esperadas.

INSTITUTO DE INFORMÁTICA UFG

Processo do Teste Caixa Branca

- Os passos básicos para se aplicar um critério de teste caixa branca são os seguintes:
  1. A implementação do produto em teste é analisada.
  2. Caminhos através da implementação são escolhidos.
  3. Valores de entradas são selecionados de modo que os caminhos selecionados sejam executados.
  4. As saídas esperadas para as entradas escolhidas são determinadas.
  5. Os casos de testes são construídos.
  6. Os casos de testes são executados.
  7. As saídas obtidas são comparadas com as saídas esperadas.
  8. Um relatório é gerado para avaliar o resultado dos testes.

INSTITUTO DE INFORMÁTICA UFG

Critérios de Teste Estrutural

- Critérios Baseados em Fluxo de Controle:
  - Todos-Nós, Todas-Arestas, Todos-Caminhos.
- Critérios Baseados em Fluxo de Dados:
  - Todas-Definições, Todos-P-Usos, Todos-C-Usos, Todos-Usos, Todos-Du-Caminhos, Todos-Pot-Usos,Todos-Pot-Du-Caminhos, Todos-Pot-Usos/Du.

INSTITUTO DE INFORMÁTICA UFG

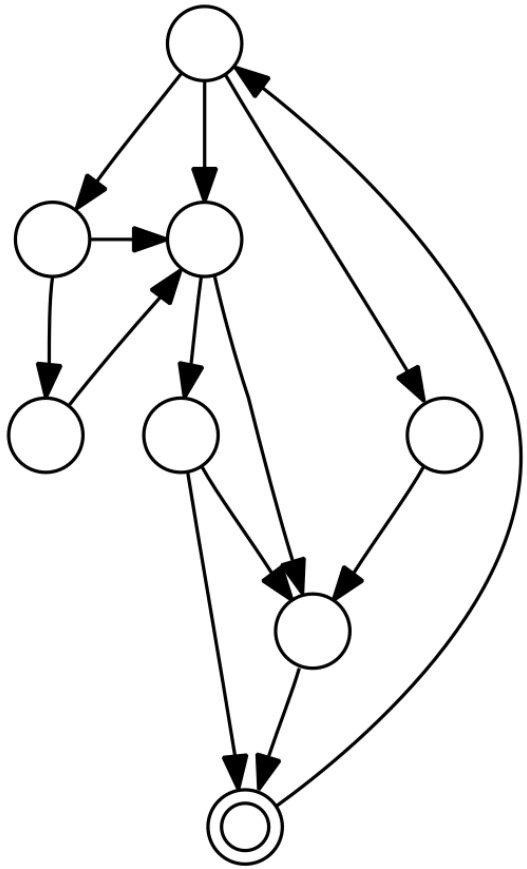
Uso nas Fases de Teste

- Critérios da técnica caixa branca também podem ser utilizados em todas as fases/níveis de teste.
- São mais comuns no teste de unidade e de integração.
- Teste de caminhos:
  - Caminhos dentro de uma unidade.
  - Caminhos entre unidades.
  - Caminhos entre sub-sistemas.
  - Caminhos entre o sistema todo.

INSTITUTO DE INFORMÁTICA UFG

Uso nas Fases de Teste

- Diferentes caminhos dentro de um produto em teste.



Desvantagens do Teste Caixa Branca

- O número de caminhos a serem executados pode ser infinito (semelhante ao teste exaustivo).

Desvantagens do Teste Caixa Branca

- O número de caminhos a serem executados pode ser infinito (semelhante ao teste exaustivo).
- O caso de teste selecionado pode não revelar o defeito sensível a dado. Por exemplo:

Desvantagens do Teste Caixa Branca

- O número de caminhos a serem executados pode ser infinito (semelhante ao teste exaustivo).
- O caso de teste selecionado pode não revelar o defeito sensível a dado. Por exemplo:
  - y = x * 2; // deveria ser y = x ** 2

INSTITUTO DE INFORMÁTICA UFG

Desvantagens do Teste Caixa Branca

- O número de caminhos a serem executados pode ser infinito (semelhante ao teste exaustivo).
- O caso de teste selecionado pode não revelar o defeito sensível a dado. Por exemplo:
  - y = x * 2; // deveria ser y = x ** 2
  - funciona corretamente para x = 0, y=0 e x = 2, y = 4.

INSTITUTO DE INFORMÁTICA UFG

Desvantagens do Teste Caixa Branca

- O número de caminhos a serem executados pode ser infinito (semelhante ao teste exaustivo).
- O caso de teste selecionado pode não revelar o defeito sensível a dado. Por exemplo:
  - y = x * 2; // deveria ser y = x ** 2
  - funciona corretamente para x = 0, y=0 e x = 2, y = 4.
- Assume fluxo de controle correto (ou próximo do correto). Casos de testes são baseados em caminhos existentes: caminhos inexistentes não podem ser descobertos.

INSTITUTO DE INFORMÁTICA UFG

Desvantagens do Teste Caixa Branca

- O número de caminhos a serem executados pode ser infinito (semelhante ao teste exaustivo).
- O caso de teste selecionado pode não revelar o defeito sensível a dado. Por exemplo:
  - y = x * 2; // deveria ser y = x ** 2
  - funciona corretamente para x = 0, y=0 e x = 2, y = 4.
- Assume fluxo de controle correto (ou próximo do correto). Casos de testes são baseados em caminhos existentes: caminhos inexistentes não podem ser descobertos.
- Determinação de caminhos não executáveis pode ser um problema.

INSTITUTO DE INFORMÁTICA UFG

Desvantagens do Teste Caixa Branca

- O número de caminhos a serem executados pode ser infinito (semelhante ao teste exaustivo).
- O caso de teste selecionado pode não revelar o defeito sensível a dado. Por exemplo:
  - y = x * 2; // deveria ser y = x ** 2
  - funciona corretamente para x = 0, y=0 e x = 2, y = 4.
- Assume fluxo de controle correto (ou próximo do correto). Casos de testes são baseados em caminhos existentes: caminhos inexistentes não podem ser descobertos.
- Determinação de caminhos não executáveis pode ser um problema.
- Dificuldade de automatização.

INSTITUTO DE INFORMÁTICA UFG

Desvantagens do Teste Caixa Branca

- O número de caminhos a serem executados pode ser infinito (semelhante ao teste exaustivo).
- O caso de teste selecionado pode não revelar o defeito sensível a dado. Por exemplo:
  - y = x * 2; // deveria ser y = x ** 2
  - funciona corretamente para x = 0, y=0 e x = 2, y = 4.
- Assume fluxo de controle correto (ou próximo do correto). Casos de testes são baseados em caminhos existentes: caminhos inexistentes não podem ser descobertos.
- Determinação de caminhos não executáveis pode ser um problema.
- Dificuldade de automatização.
- Habilidades de programação avançadas exigidas para compreender o código e decidir pela executabilidade ou não de um caminho.

INSTITUTO DE INFORMÁTICA UFG

Vantagens do Teste Caixa Branca

- Eficaz em determinar defeitos de lógica e de programação, especialmente no nível de unidade.
- É possível garantir que partes essenciais ou críticas do programa tenham sido executadas.
  - Requisito mínimo de teste: garantir que o programa foi liberado tendo seus comandos executados ao menos uma vez por pelo menos um caso de teste.
- É poss??vel saber o que ainda não foi testado.

Níveis de Cobertura

- Diferentes níveis de cobertura podem ser definidos em função dos elementos do GFC.
- Cobertura: porcentagem dos requisitos que foram testados versus o total de requisitos gerados.
- Oito diferentes n??veis de cobertura são definidos por [Copeland, 2003].
- Quanto maior o nível, maior o rigor do critério de teste, ou seja, mais caso de teste ele exige para ser satisfeito.
  - Nível 0 ⟵ Nível 1 ⟵ Nível 2 ⟵ Nível 3 ⟵ Nível 4 ⟵ Nível 5 ⟵ Nível 6 ⟵ Nível 7

INSTITUTO DE INFORMÁTICA UFG

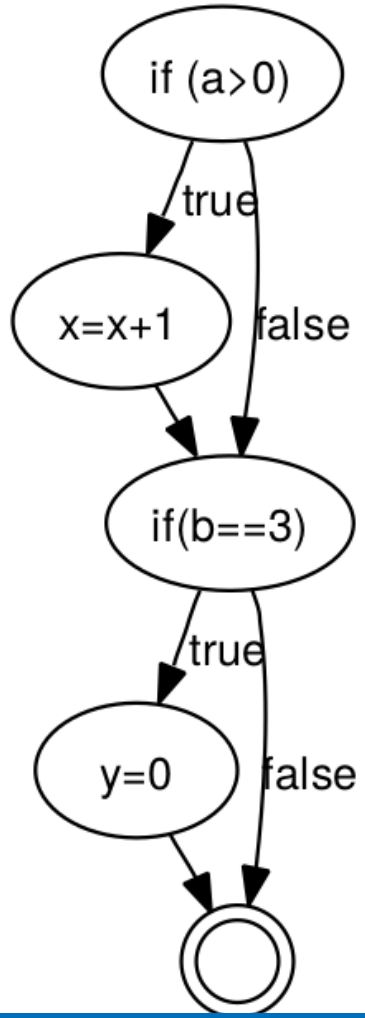
Níveis de Cobertura

- Nível 0: qualquer valor de cobertura inferior a 100% da cobertura de todos os comandos.
- Nível 1: 100% de cobertura de comandos. (Todos-Nós)
- Nível 2: 100% de cobertura de decisões. (Todos-Arcos)
- Nível 3: 100% de cobertura de condições.
- Nível 4: 100% de cobertura decisões/condições.
- Nível 5: 100% de cobertura de condições múltiplas.
- Nível 6: cobertura de loop.
- Nível 7: 100% de cobertura de caminhos. (Todos-Caminhos)

INSTITUTO DE INFORMÁTICA UFG

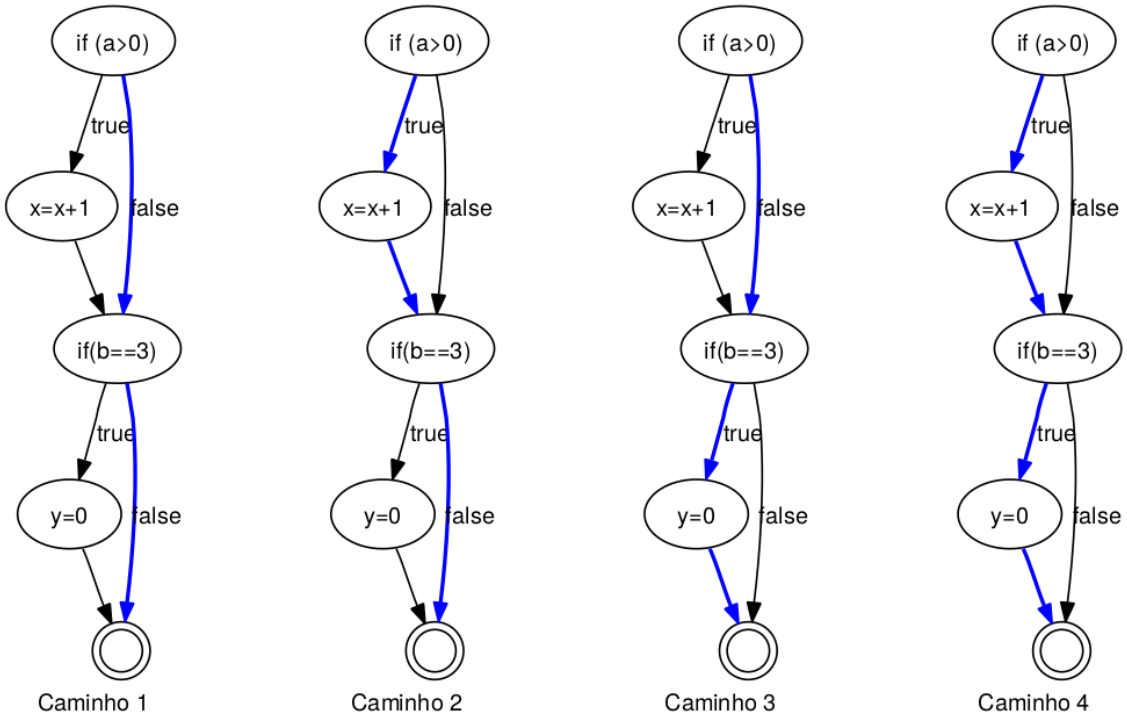
Níveis de Cobertura - Exemplo

```
1   if (a>0){
2       x=x+1;
3   }
4   if (b==3){
5       y=0;
6   }
```



Níveis de Cobertura - Exemplo

Nível 1: 100% de cobertura de comandos (requisito mínimo de teste)



Um caso de teste é suficiente para cobrir todos os comandos mas não todos os caminhos. Por exemplo, use a=6 e b=3 para cobrir o "Caminho 4".

INSTITUTO DE INFORMÁTICA UFG

Referências Bibliográficas

Beizer, Boris.
Software Testing Techniques.
Van Nostrand Reinhold Company, New York, 2nd edition,1990.

B. Boehm and V. R. Basili.
Software defect reduction top 10 list.
Computer, 34(1):135-137, 2001. ISSN 0018-9162. doi: http://dx.doi.org/10.1109/2.962984.

Lee Copeland.
A Practitioner's Guide to Software Test Design
Artech House Publishers, Boston, 2003.

R. D. Craig and S. P. Jaskiel.
Systematic Software Testing.
Artech House Publishers, 2002.

Imagens, Google
Google Imagens.
https://www.google.com.br. Acesso em 19/06/2019.

IEEE.
IEEE standard glossary of software engineering terminology.
Standard 610.12-1990 (R2002),
IEEE Computer Society Press, 2002.

Harley, Nick.
11 of the most costly software errors in history [2019 update]
https://raygun.com/blog/costly-software-errors-history/. Acesso em 19/06/2019.

Neto, Arilo Cláudio Dias Neto.
Introdução ao Teste de Software.
Engenharia de Software Magazine. Ed. 01, 2009

F. Shull, V. Basili, B. Boehm, A. W. Brown, P. Costa, M. Lindvall, D. Port, I. Rus, R.e Tesoriero, and M. Zelkowitz.
What we have learned about fighting defects.
In VIII International Symposium on Software Metrics - METRICS'02, pages 249-258, Washington, DC, USA, June 2002. IEEE Computer Society. ISBN 0-7695-1339-5.

INSTITUTO DE INFORMÁTICA UFG