SSC721 – Teste e Inspeção de Software

Teste de Software

Simone Senger de Souza
srocio@icmc.usp.br

Instituto de Ciências Matemáticas e de Computação — ICMC/USP

Roteiro

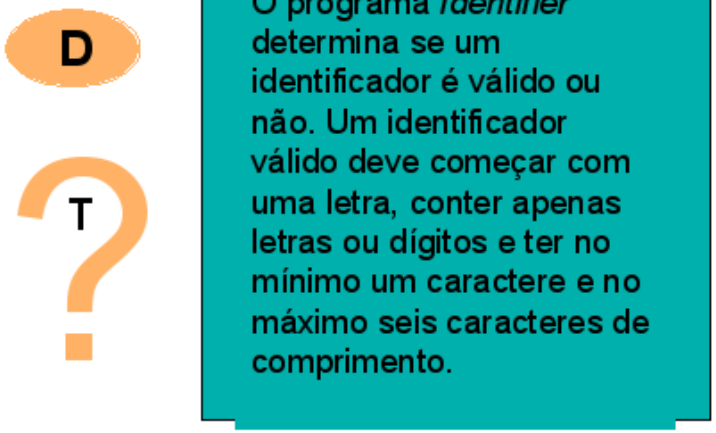
- Teste de Software
- Terminologia de Erros
- Caso de Teste
- Técnicas de Teste
- Etapas de Teste
- Fases de Teste
- Conclusão

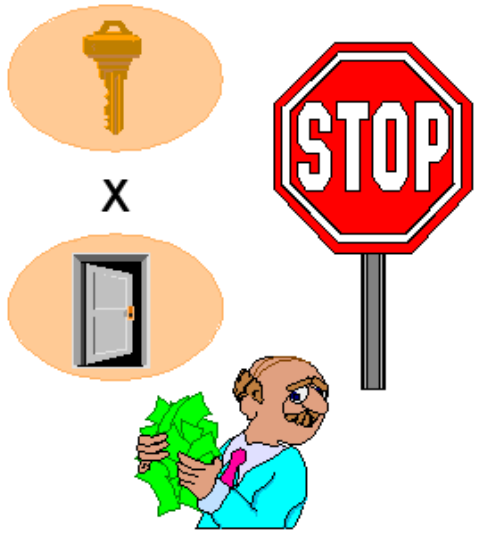
Teste de Software

- Análise dinâmica do produto de software.
  - Processo de executar o software de modo controlado, observando seu comportamento em relação aos requisitos especificados.
- Processo de executar um programa com a intenção de encontrar erros.
  - O teste bem sucedido é aquele que consegue determinar casos de teste que resultem na falha do programa sendo analisado.

Teste de Software

- Como testar esse produto?
  - Que casos de teste utilizar?
  - Como garantir que o produto não contém erros? Critérios...
  - Como decidir se o produto foi suficientemente testado?

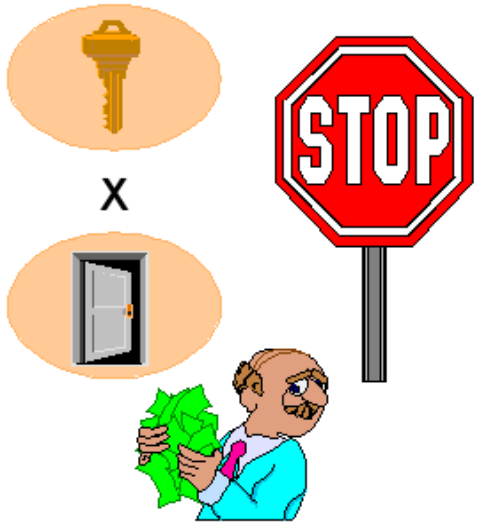




Teste de Software

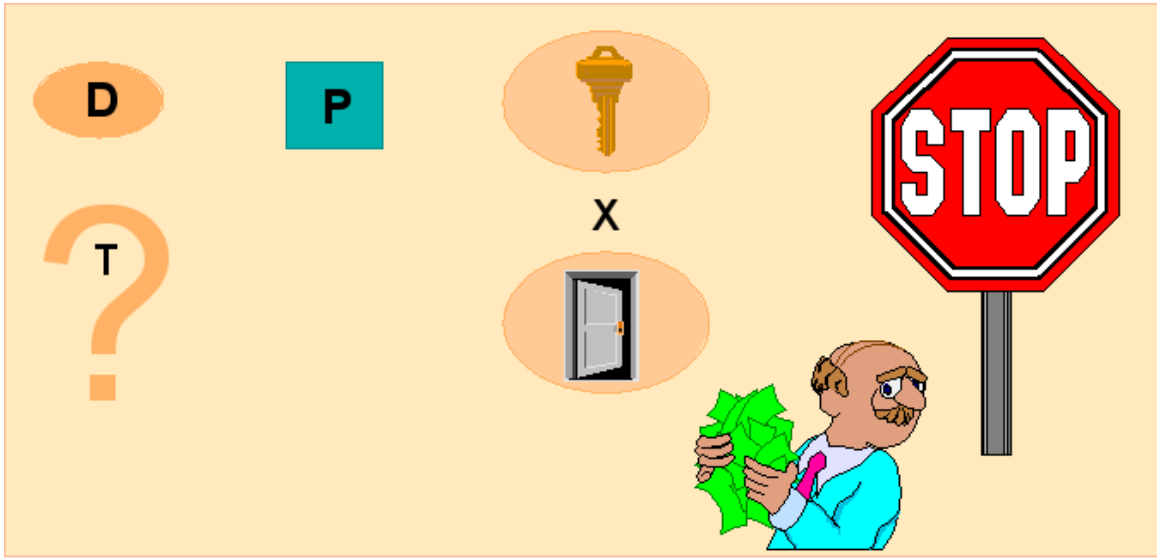
- Como testar esse produto?
  - Que casos de teste utilizar?
  - Como garantir que o produto não contém erros? Critérios...
  - Como decidir se o produto foi suficientemente testado?


```
main () {
   int   valor, num, fat;
   fat= 1;
   scanf("%d",&valor);
   num = valor;
   if (num >= 0) {
     while (num > 1) {
       fat = fat * num;
       num--; }
     printf("%d\n",fat); }
   else
     printf("Erro!\n"); }
```



Objetivos do Teste

- Revelar a presença de erros.



- Inexistência de erro:
  - Software é de alta qualidade?
  - Conjunto de casos de teste T é de baixa qualidade?

Objetivos do Teste

- Revelar a presença de erros.

Através da atividade de teste pode-se detectar a **presença de erros** em um programa, embora não se possa concluir a ausência deles.

- Quando o processo de teste não detecta erros...
  - O teste pode não ter sido suficientemente completo para revelar os erros existentes.
  - Não se pode afirmar que o programa esteja isento de erros e, portanto, correto.

Objetivos do Teste

- Verificar se o software executa conforme especificado na documentação do projeto.
- Fornecer uma avaliação da qualidade do software.

Apesar de não ser possível, por meio de testes, provar que um programa está correto, estes contribuem para **aumentar a confiança** de que o software desempenha as funções especificadas.

Processo de Teste

- Diferentes organizações e indivíduos têm visões diferentes do propósito dos testes.
- Processo de Software *versus* Processo de Teste.
- Níveis de Maturidade de Teste:
  - Nível 0 - Não há diferença entre teste e depuração.

Processo de Teste

- Diferentes organizações e indivíduos têm visões diferentes do propósito dos testes.
- Processo de Software *versus* Processo de Teste.
- Níveis de Maturidade de Teste:
  - Nível 0 - Não há diferença entre teste e depuração.
  - Nível 1 - O propósito do teste é mostrar que o software funciona.

Processo de Teste

- Diferentes organizações e indivíduos têm visões diferentes do propósito dos testes.
- Processo de Software *versus* Processo de Teste.
- Níveis de Maturidade de Teste:
  - Nível 0 - Não há diferença entre teste e depuração.
  - 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.

Processo de Teste

- Diferentes organizações e indivíduos têm visões diferentes do propósito dos testes.
- Processo de Software *versus* Processo de Teste.
- Níveis de Maturidade de Teste:
  - Nível 0 - Não há diferença entre teste e depuração.
  - 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.

Processo de Teste

- Diferentes organizações e indivíduos têm visões diferentes do propósito dos testes.
- Processo de Software *versus* Processo de Teste.
- Níveis de Maturidade de Teste:
  - Nível 0 - Não há diferença entre teste e depuração.
  - 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.

Desafios do Teste

- Alguém já testou algum produto de software?

Desafios do Teste

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

Desafios do Teste

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

Desafios do Teste

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

Desafios do Teste

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

Desafios do Teste

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

Desafios do Teste

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

Desafios do Teste

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

Desafios do Teste

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

Desafios do Teste

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

Desafios do Teste

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

Limitações do Teste

- Observe o exemplo a seguir:

```
1  int blech(int j) {
2      j = j - 1;  // deveria ser j = j + 1
3      j = j / 30000;
4      return j;
5  }
```

Código-fonte: `blech.java`

- Considerando o tipo inteiro com 16 bits (2 bytes), o menor valor possível seria -32.768 e o maior seria 32.767, resultando em 65.536 valores diferentes possíveis.
- Haverá tempo suficiente para se criar 65.536 casos de teste? E se os programas forem maiores? Quantos casos de teste serão necessários?

Limitações do Teste

```
1  int blech(int j) {
2      j = j - 1;  // deveria ser j = j + 1
3      j = j / 30000;
4      return j;
5  }
```

Código-fonte: `blech.java`

- Quais valores escolher?

Entrada (j)	Saída Esperada	Saída Obtida
1	0	0
42	0	0
40000	1	1
-64000	-2	-2

- Quais valores de entrada revelam o erro?

Limitações do Teste

- Nenhum dos casos de testes anteriores revelou o erro.

Limitações do Teste

- Nenhum dos casos de testes anteriores revelou o erro.
- Somente quatro valores do intervalo de entrada válido revelam o erro:

Entrada (j)	Saída Esperada	Saída Obtida
-30000	0	-1
-29999	0	-1
30000	1	0
29999	1	0

Limitações do Teste

- Nenhum dos casos de testes anteriores revelou o erro.
- Somente quatro valores do intervalo de entrada válido revelam o erro:

Entrada (j)	Saída Esperada	Saída Obtida
-30000	0	-1
-29999	0	-1
30000	1	0
29999	1	0

- Qual a chance de tais valores serem selecionados???

Limitações do Teste

- Corretitude de Programas
- Equivalência de Programas
- Executabilidade
- Correção Coincidente

Corretitude de Programas

- Não existe um algoritmo de teste de propósito geral para provar a corretitude de um programa.
  - Diz-se que um programa P com domínio de entrada D é correto com respeito a uma especificação S se, para qualquer item de dado $d \in D, S(d) = P^*(d)$.

  Ou seja...

Se o comportamento do programa está de acordo com o comportamento esperado para todos os itens de dado pertencentes ao seu domínio de entrada.

- Necessidade de um oráculo de teste.

Oráculo de Teste

Testador ou algum outro mecanismo que possa determinar, para qualquer item de dado d pertencente ao domínio de entrada D, se $S(d) = P^*(d)$ dentro de limites de tempo e esforços razoáveis.

Ou seja...

- Um oráculo é responsável por decidir se os valores de saída resultantes da execução do programa são corretos.
  - Qualquer programa, processo ou dados que forneça ao projetista a saída esperada para um caso de teste.

Oráculo de Teste

- Tipos de oráculos:
  - Kiddie Oracles
    - Simplesmente execute o programa e observe sua saída. Se ela parecer correta, deve estar correta.
  - Conjunto de Teste de Regressão
    - Execute o programa e compare a saída obtida com a saída produzida por uma versão mais antiga do programa.
  - Validação de Dados
    - Execute o programa e compare a saída obtida com uma saída padrão determinada por uma tabela, fórmula ou outra definição aceitável de saída válida.

Equivalência de Programas

- Dados dois programas, se eles são equivalentes.
- Dados dois caminhos (seqüências de comandos) de um programa ou de programas diferentes, se eles computam a mesma função.

- Dados dois programas P_1 e P_2 com domínio de entrada D, diz-se que P_1 e P_2 são equivalentes se, para qualquer item de dado $d \in D, P_1^*(d) = P_2^*(d)$.

Ou seja...

Se o comportamento de ambos os programas é idêntico para todos os itens de dados pertencentes ao domínio de entrada.

Executabilidade

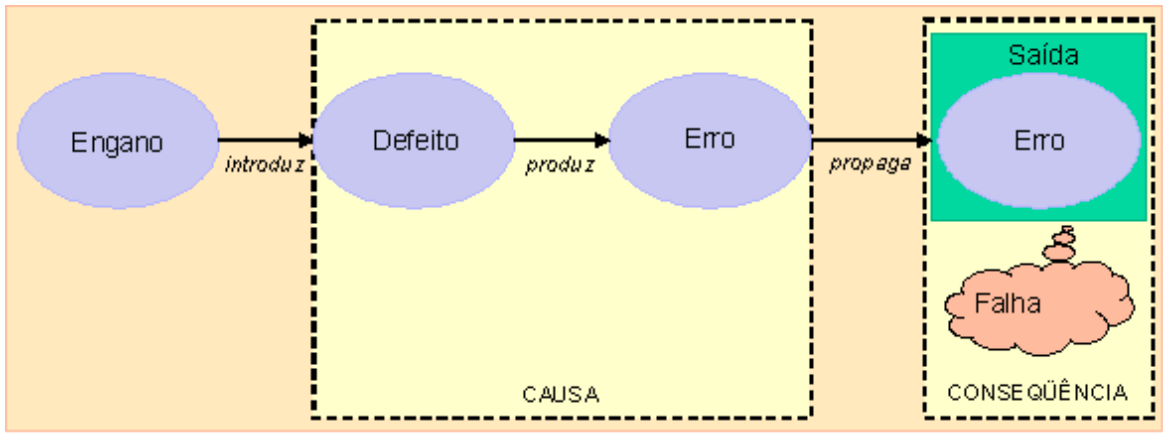
- Dado um programa P com domínio de entrada D, diz-se que um caminho é executável se existe ao menos um item de dado $d \in D$ que leve à execução desse caminho.
- Do contrário, diz-se que o caminho é não-executável.

Correção Coincidente

- O programa pode apresentar, coincidentemente, um resultado correto para um particular item de dado $d \in D$, satisfazendo a um requisito de teste e não revelando a presença do erro.
  - Entretanto, se escolhido um outro dado de entrada, o resultado obtido seria incorreto e a presença do erro seria identificada.

Terminologia de Erros

- Engano x Defeito x Erro x Falha (Padrão IEEE 610.12-1990)
  - Um engano introduz um defeito no software.
  - O defeito, quando ativado, pode produzir um erro.
  - O erro, se propagado até a saída do software, constitui uma falha.



Engano x Defeito x Erro x Falha

- Considere o comando $(z = y + x)$ substituído por engano pelo comando $(z = y - x)$.
- Se o defeito introduzido for ativado com $(x = 0)$:
  - Nenhum valor incorreto para a variável z é produzido.
  - O defeito é ativado mas não produz um erro e, conseqüentemente, não ocorre uma falha.
- Para qualquer outro valor de x $(x \neq 0)$:
  - A ativação do defeito produz um erro na variável z.
  - Tal erro, se propagado até a saída, caracteriza uma falha.

Engano (*Mistake*)

- Ação humana que introduz um defeito no software.
  - Ação incorreta tomada pelo programador.

Defeito (*Fault*)

- Passo, processo ou definição de dados incorreta, incompleta, ausente ou extra que, ao ser executada, pode produzir um erro no programa.
  - Instrução ou comando incorreto.

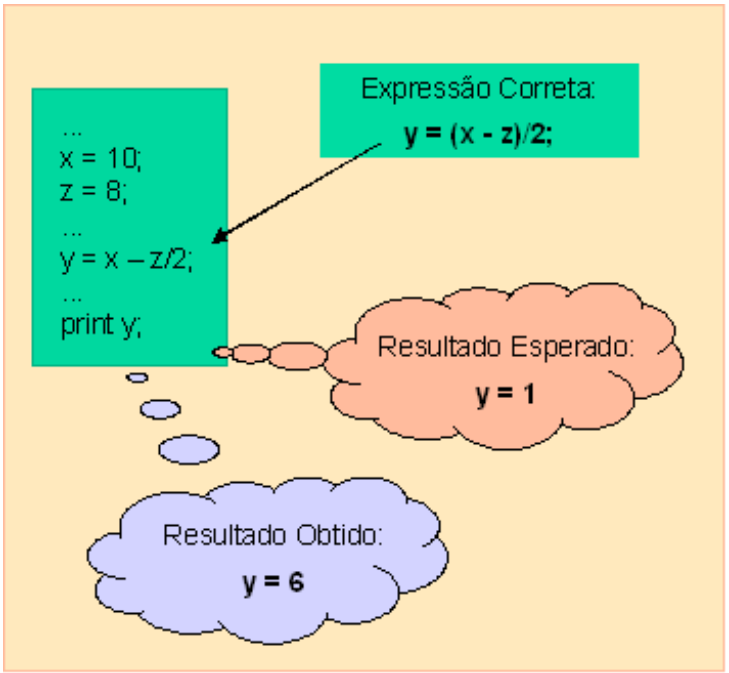
Erro (*Error*)

- Diferença entre o valor obtido e o valor esperado, ou seja, qualquer estado na execução do programa, intermediário ou final, que seja inconsistente.
  - Erro computacional.
  - Erro de domínio.

Erro Computacional

- Um erro computacional provoca uma computação incorreta mas a seqüência de comandos executada é igual à seqüência esperada.

A atribuição de um valor incorreto a uma variável do programa por uma expressão aritmética correta corresponde a um erro computacional.

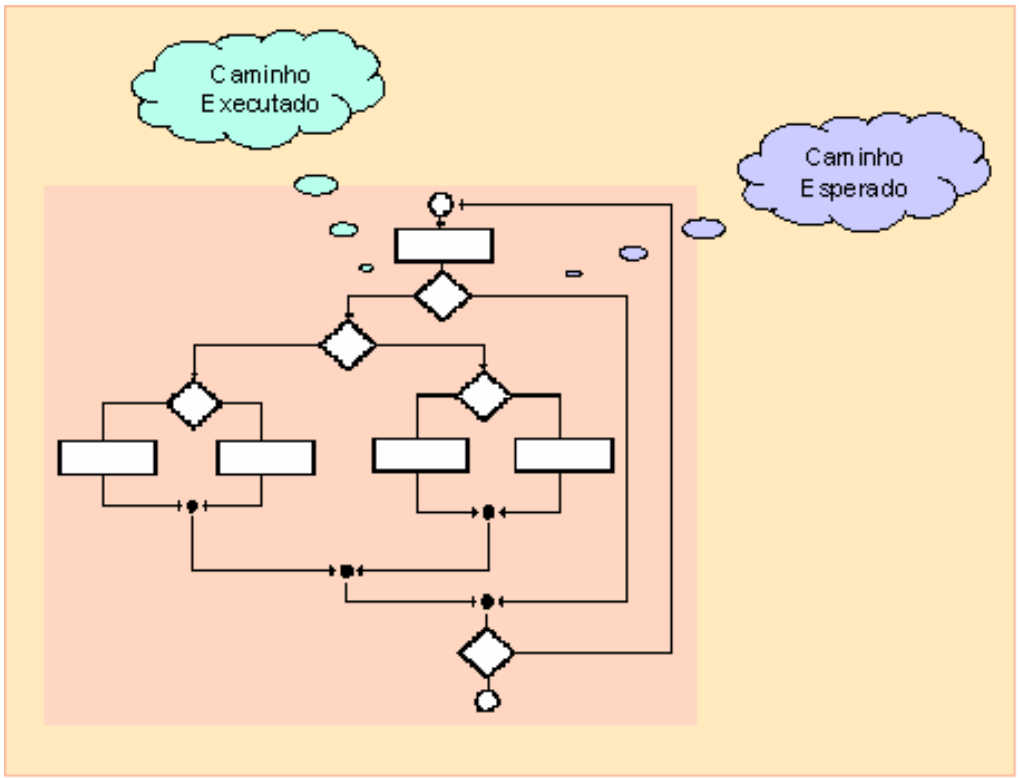


Erro de Domínio

- Um erro de domínio faz com que a seqüência de comandos executada seja diferente da seqüência esperada, ou seja, uma seqüência errada é selecionada.
- Causas
  - Erros de domínio podem ser gerados por um erro computacional ou uma condição incorreta de um comando de decisão.

Erro de Domínio

A seleção de saídas incorretas do comando de decisão devido a operadores relacionais incorretos corresponde a um erro de domínio.



Falha(*Failure*)

- Evento notável em que o sistema viola suas especificações, ou seja, é a produção de uma saída incorreta com relação à especificação.

Caso de Teste

- O segredo do sucesso do teste está no projeto dos casos de teste.

"The design of tests for software and other engineering products can be as challenging as the initial design of the product itself. Yet ... software engineers often treat testing as an afterthought, developing test cases that 'feel right' but have little assurance of being complete. Recalling the objectives of testing, we must design tests that have the highest likelihood of finding the most errors with a minimum amount of time and effort."

Caso de Teste

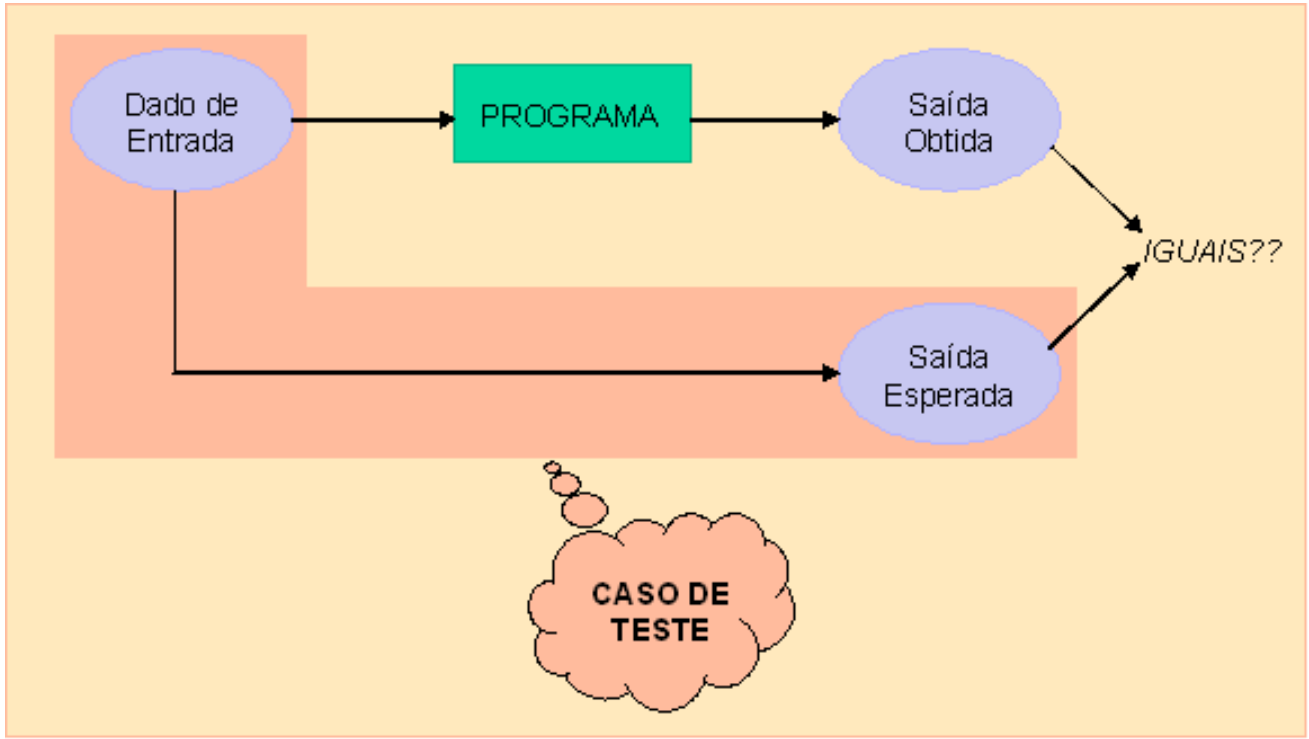
- Um bom caso de teste tem alta probabilidade de revelar um erro ainda não descoberto.
- Casos de teste também podem revelar especificações incompletas ou ambíguas.

Partes de um Caso de Teste

Par ordenado (d; S(d)) no qual d é um elemento pertencente ao domínio de entrada D e S(d) corresponde à sua respectiva saída esperada.

- Dado de entrada (*inputs*).
  - Dado necessário para uma execução do programa.
- Saída esperada (*outputs*).
  - Resultado de uma execução do programa.
  - Pressupõe-se a existência de um oráculo de teste.

Caso de Teste



Ordem de Execução

- Existem dois estilos de projeto de casos de teste relacionados com a ordem de execução:
  - Casos de teste em cascata.
  - Casos de teste independentes.

Casos de Teste em Cascata

- Os casos de teste devem ser executados um após o outro, em uma ordem específica.
- O estado do sistema deixado pelo primeiro caso de teste é reaproveitado pelo segundo e assim sucessivamente.
- Por exemplo, considere o teste de uma base de dados:
  - Criar um registro.
  - Ler um registro.
  - Atualizar um registro.
  - Ler um registro.
  - Apagar um registro.
  - Ler o registro apagado.

Casos de Teste em Cascata

- Vantagem: Casos de testes tendem a ser pequenos e simples. Fáceis de serem projetados, criados e mantidos.
- Desvantagem: Se um caso de teste falhar, os casos de teste subseqüentes também podem falhar.

Casos de Teste Independentes

- Cada caso de teste é inteiramente auto-contido.
- Vantagem: Casos de teste podem ser executados em qualquer ordem.
- Desvantagem: Casos de teste tendem a ser grandes e complexos, mais difíceis de serem projetados, criados e mantidos.

Técnicas de Teste

- Diferentes tipos de teste podem ser utilizados para verificar se um programa comporta-se como especificado.
- Basicamente, os testes podem ser classificados em teste caixa-preta (*black-box testing*) ou teste caixa-branca (*white-box testing*).
- Esses tipos de teste correspondem às chamadas técnicas de teste.

Técnicas de Teste

- As técnicas de teste são definidas conforme o tipo de informação utilizada para realizar o teste.
- Contemplam diferentes perspectivas do software: aspecto complementar!!!!
  - Técnica Caixa-Preta
    - Os testes são baseados exclusivamente na especificação de requisitos do programa.
    - Nenhum conhecimento de como o programa está implementado é requerido.
  - Técnica Caixa-Branca
    - Os testes são baseados na estrutura interna do programa, ou seja, na implementação do mesmo.

Critérios de Teste

- Maneira sistemática e planejada para conduzir os testes.
- Fornece indicações a respeito de quais casos de teste utilizar de modo a aumentar as chances de revelar erros no programa.
- Quando erros não forem revelados...
  - Estabelecer um nível elevado de confiança na correção do programa.

Critérios de Teste

- Propriedades Mínimas:
  1. Garantir, do ponto de vista de fluxo de controle, a cobertura de todos os desvios condicionais.
  2. Requerer, do ponto de vista de fluxo de dados, ao menos um uso de todo resultado computacional.
  3. Requerer um conjunto de casos de teste finito.

Critério de Geração

Procedimento para **escolher** casos de teste para o teste de P.

- T é C-adequado por construção.

Critério de Adequação

Predicado para **avaliar** um conjunto de casos de teste T no teste do programa P.

- Um critério de adequação C é utilizado para verificar se o conjunto de casos de teste T satisfaz os requisitos de teste estabelecidos pelo critério.

Análise de Cobertura

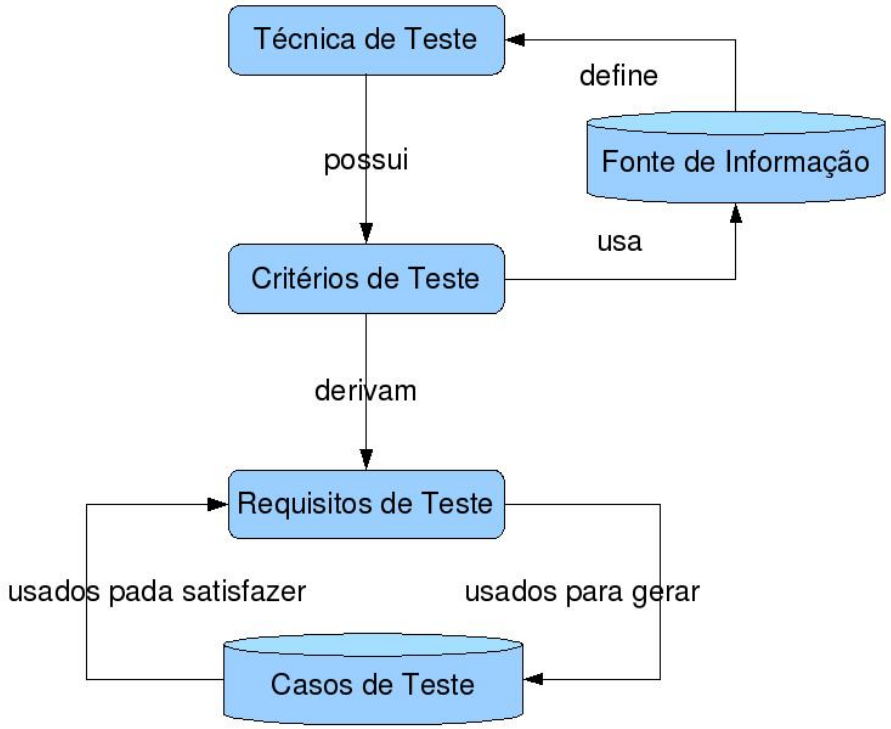
- Consiste em determinar o percentual de elementos estabelecidos por um critério de teste que foram executados pelo conjunto de casos de teste.
- Com essas informações...
  - O conjunto de casos de teste pode ser melhorado para que os elementos ainda não abordados sejam testados.
    - Adição de novos casos de teste ao conjunto.

Requisitos de Teste

- Cada critério estabelece um conjunto de requisitos de teste específico.
- Casos de teste são selecionados de modo a satisfazer os requisitos estabelecidos pelo critério em questão.
- Dados um programa P, um conjunto de casos de teste T e um critério C, diz-se que:
  - T é C-adequado para o teste de P se T preencher os requisitos de teste estabelecidos pelo critério C.

Técnicas, Critérios e Requisitos de Teste

- Termos utilizados na área de teste e seus relacionamentos.



Etapas de Teste

- A atividade de teste envolve basicamente quatro etapas:
  - Planejamento
  - Projeto de casos de teste
  - Execução do programa
  - Análise de resultados

Planejamento

- São estimados os recursos necessários e definidas as estratégias, métodos e técnicas de teste a serem utilizadas.
- Plano de Teste
  - Apresenta o planejamento para a execução do teste, incluindo:
    - Abrangência
    - Abordagem
    - Recursos e requisitos do ambiente
    - Cronogramas
  - Identifica os itens e as funcionalidades a serem testados.
  - Identifica as tarefas de teste a serem realizadas e os riscos associados com a atividade de teste.

Projeto de Casos de Teste

- São elaborados os casos de teste com os quais o programa deve ser testado.
  - É uma das etapas fundamentais da atividade de teste.
  - Pode ser tão difícil quanto o projeto do próprio produto a ser testado.
  - É um dos melhores mecanismos para prevenção de defeitos.
  - É tão eficaz em identificar erros quanto a execução dos casos de teste projetados.
  - Casos de teste elaborados nessa etapa possuem forte dependência com relação ao critério de teste utilizado.

Execução do Programa

- O programa é executado com os casos de teste elaborados.
  - Fazer registros cronológicos de detalhes relevantes relacionados com a execução dos testes.
  - Documentar qualquer evento que ocorra durante a atividade de teste e que requeira análise posterior.

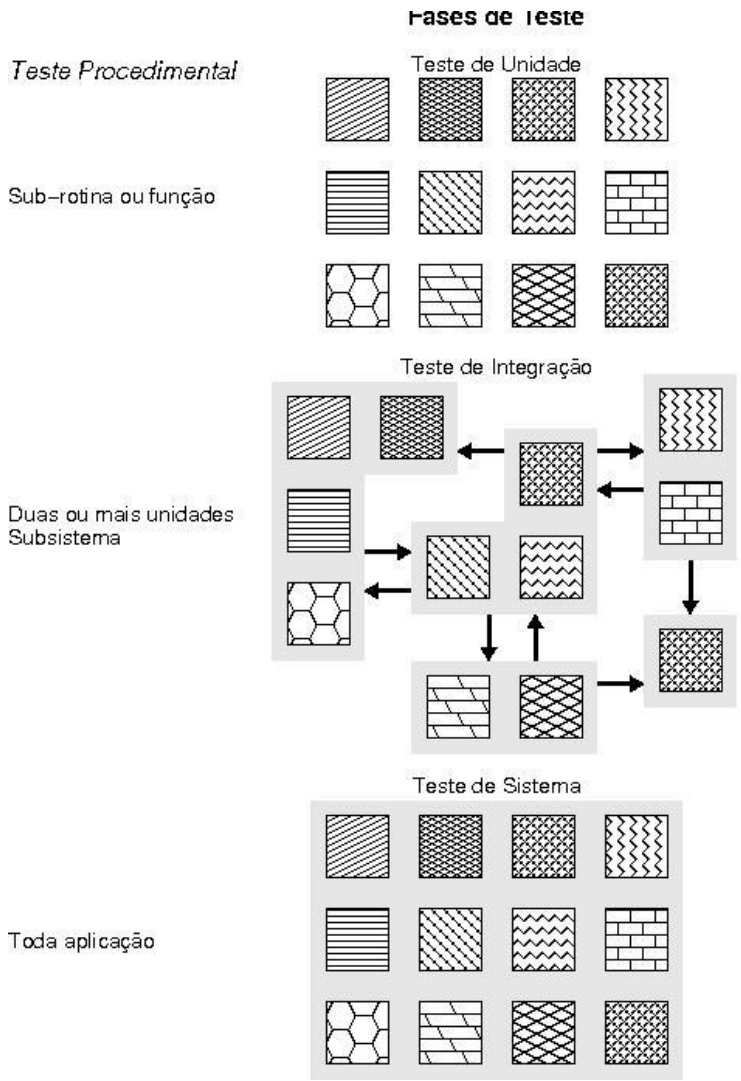
Análise dos Resultados

- O comportamento do programa é avaliado em relação aos casos de teste a fim de determinar se o mesmo está correto ou não.
  - Fornecer avaliações com base nos resultados observados.

Fases de Teste

- Assim como outras atividades de Engenharia de Software, a atividade de teste também é dividida em fases.
  - Cada fase aborda diferentes tipos de erros e aspectos do software.
- Conceito de dividir para conquistar.
- Objetivo: minimizar a complexidade na condução dos testes.
- Idéia: Iniciar os testes a partir da menor unidade executável até atingir o programa como um todo.
  - Teste de Unidade
  - Teste de Integração
  - Teste de Sistema

Fases de Teste



Teste de Unidade

- Concentra esforços nas unidades de projeto do software a fim de verificar se estas funcionam adequadamente.
- Cada unidade do programa é explorada separadamente, procurando-se identificar erros de lógica e de implementação nas mesmas.

Teste de Unidade

- Unidade (Módulo)
  - Componente de software que não pode ser dividido.
  - Menor parte funcional de um programa que pode ser executada.
- Diferentes linguagens possuem unidades diferentes.
  - Pascal e C possuem procedimentos ou funções.
  - Java e C++ possuem métodos (ou classes???).
  - Basic e COBOL pode ser o programa todo.

Teste de Unidade

- Como testar uma unidade que depende de outra para ser executada?

Teste de Unidade

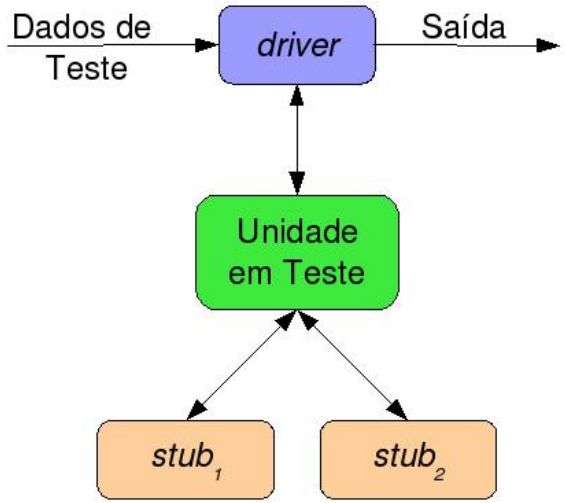
- Como testar uma unidade que depende de outra para ser executada?
- Como testar uma unidade que precisa receber dados de outra unidade para ser executada?

Drivers e Stubs

- Para auxiliar no teste de unidade, em geral, são necessários *drivers* e *stubs*.
- O *driver* é responsável por fornecer para uma dada unidade os dados necessários para que ela possa ser executada e, posteriormente, apresentar os resultados ao testador.
- O *stub* serve para simular o comportamento de uma unidade que ainda não foi desenvolvida, mas da qual a unidade em teste depende.

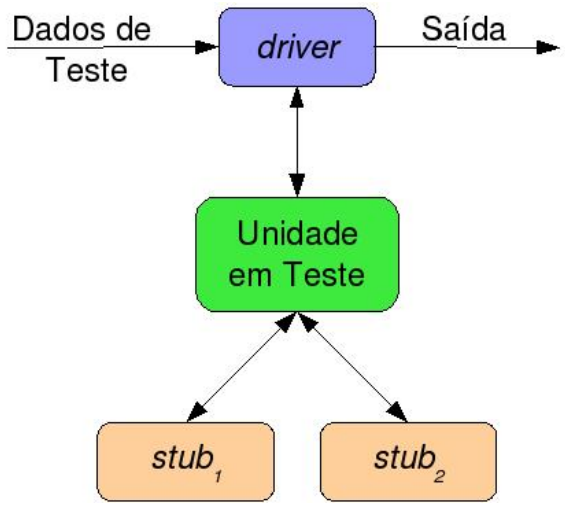
Drivers e Stubs

- Considere F uma dada unidade a ser testada.
- *Driver*
  - Responsável por ativar e coordenar o teste de F, recebendo os dados de teste fornecidos pelo testador, passando tais dados na forma de parâmetros para F, coletando os resultados relevantes produzidos por F, e apresentando-os para o testador.



Drivers e Stubs

- Considere F uma dada unidade a ser testada.
- *Stub*
  - Unidade que substitui, durante o teste, uma unidade utilizada (chamada) por F.
  - Em geral, simula o comportamento da unidade chamada por F com o mínimo de computação ou manipulação de dados.

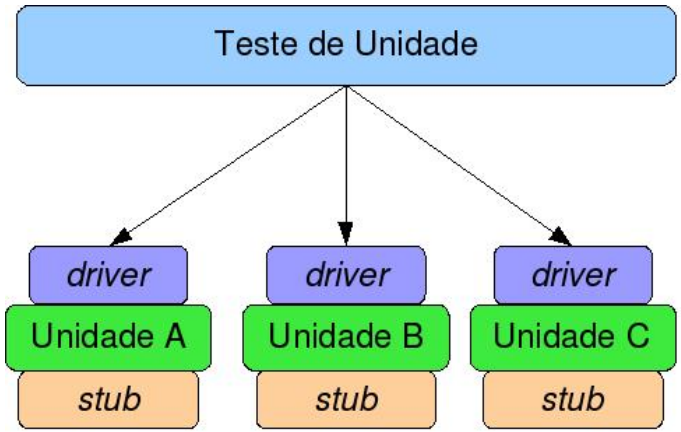


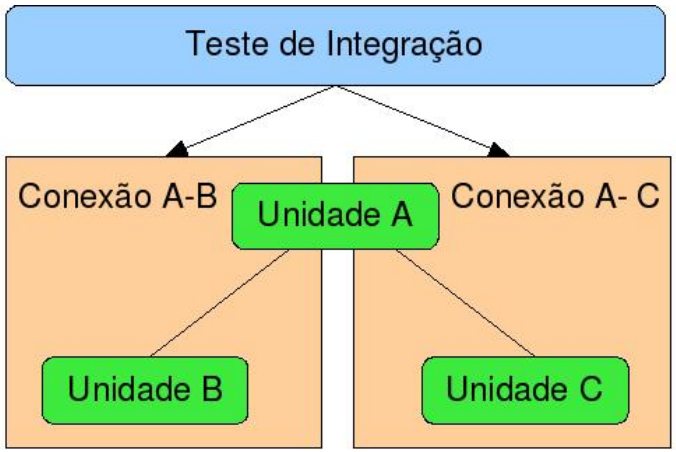
Teste de Integração

- Atividade sistemática aplicada durante a integração da estrutura do programa.
- Visa a identificar erros associados às interfaces entre os módulos do software.
- O objetivo é, a partir dos módulos testados no nível de unidade, construir a estrutura de programa que foi determinada pelo projeto.
  - Verificar se as unidades testadas individualmente comunicam-se conforme desejado.

Teste de Integração

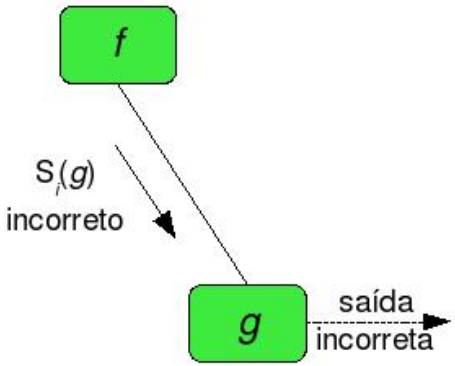
- Por que testar a integração entre unidades se as mesmas, em isolado, funcionam corretamente?
  - Dados podem se perder na interface das unidades.
  - Variáveis globais podem sofrer alterações indesejadas.

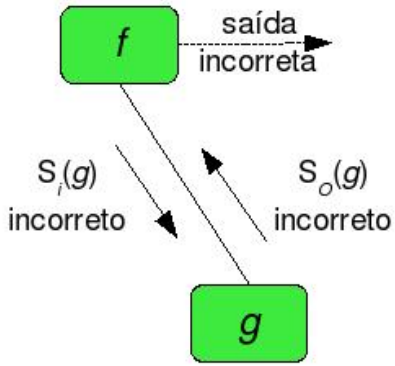


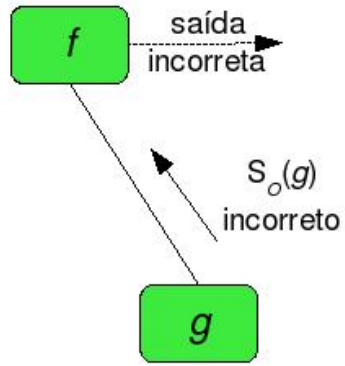


Teste de Integração

- Tipos de erros de integração:







Teste de Sistema

- Visa a identificar erros de funções e características de desempenho que não estejam de acordo com a especificação.
- O objetivo é assegurar que o software e os demais elementos que compõem o sistema (por exemplo, hardware, banco de dados, sistema operacional) combinam-se adequadamente e que a função/desempenho global desejada é obtida.

Concluindo...

- A atividade de teste é um processo executado em paralelo com as demais atividades do ciclo de vida de desenvolvimento do software.
- A principal etapa da atividade de teste é o projeto de casos de teste.
  - A idéia é selecionar casos de teste com maior probabilidade de revelar os erros existentes.
- Diferentes técnicas e critérios de teste existem para auxiliar na atividade de teste.
- Além disso, o teste deve ser conduzido em fases para reduzir a complexidade.

Exercício de Fixação

- Projete casos de teste para o seguinte programa:

O programa **string** solicita do usuário um inteiro positivo no intervalo entre 1 e 20 e, então, solicita uma cadeia de caracteres desse comprimento. Após isso, o programa solicita um caracter e retorna a posição na cadeia em que o caracter é encontrado pela primeira vez ou uma mensagem indicando que o caracter não está presente na cadeia. O usuário tem a opção de procurar por vários caracteres.