Conceitos Básicos de Teste de Software

PPGCC12-Teste de Software

Reginaldo Ré

Aula 02

Introdução

Introdução (1/3)

- Alguém já testou algum programa ou software?
- Quais foram os maiores desafios?

Introdução (2/3)

- 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 não documentados ou que mudam rapidamente
  - Não há treinamento no **processo de teste**
  - Falta de **ferramenta** de apoio
  - Gerentes que **desconhecem teste** ou que não se preocupam com qualidade

Introdução (3/3)

- Por que existem defeitos nos sistemas?
- Quais são as causas das falhas?
- Como melhorar o processo de desenvolvimento e evitar falhas?

Exemplo (1/3)

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

- É razoável ter 65.536 casos de teste? (com 16 bits a faixa vai de -32.768 a 32.767)
- Quais dados de teste vocês usariam?

Exemplo (2/3)

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

- Os dados a seguir não revelam o defeito:

Entrada	Saída Esperada	Saída Obtida
1	0	0
42	0	0
32.000	1	1
-32.000	-1	-1

Exemplo (3/3)

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

- Somente quatro valores do intervalo de entrada válido revelam o erro:

Entrada	Saída Esperada	Saída Obtida
-30.000	0	-1
-29.999	0	-1
30.000	1	0
29.999	1	0

Teste de Software

Teste de Software

- Qual é o objetivo?
- O que é?
- Como realizar o teste?
  - **Etapas de teste**
- Como testar e medir?
  - **Técnicas e critérios de teste**
- O que testar?
  - **Fases de teste**

Objetivo dos Testes

Revelar a presença de erros

- Quando isso não acontece:
  - Software de alta qualidade? Defeitos não existem!
  - Teste de baixa qualidade?
- A qualidade do teste está fortemente ligada à qualidade do **conjunto de casos de teste**
  - Como selecionar bons casos de teste?

Objetivo dos Testes

Nas organizações

- Diferentes organizações possuem diferentes visões do propósito dos testes
- Níveis de maturidade
  - 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 uma disciplina mental institucionalizada, resultando em software de baixo risco, sem que haja muito esforço de teste

O que é teste de software?

- Análise **dinâmica** do software
  - Execução controlada
  - Observação do comportamento *versus* requisitos especificados
    - Verificar se o software executa **conforme especificado**
- Tem a intenção de encontrar **erros**
  - Quando bem sucedido determina casos de teste que evidenciem falhas
- Fornece uma avaliação da qualidade do software
  - Não prova/mostra a ausência de defeitos, mas **aumenta a confiabilidade** do software

Etapas de teste

Como realizar o teste?

1. Planejamento
   - Desenvolvimento de um **Plano de Testes**
2. Projeto de casos de teste
   - Seleção e aplicação de **Critérios de teste**
3. Execução dos testes
   - Registros cronológicos dos resultados e detalhes da execução
4. Análise dos resultados

Técnicas e critérios de teste

Como testar e medir?

- Sistematizam a atividade de teste
- Ajudam a decidir quando parar de testar
- Oferecem uma medida da atividade
  - Cobertura ?
- Mesmo quando não encontram defeito, estabelecem um nível de qualidade do teste

Fases de Teste

O que testar?

- Cada fase aborda diferentes tipos de erros e aspectos
- Usa o conceito de dividir para conquistar
  - Como forma de minimizar a complexidade
- Iniciar os testes a partir da menor unidade executável até atingir todo o programa
  - Teste de Unidade
  - Teste de Integração
  - Teste de Sistema

Limitações do Teste

- Corretitude de programas
- Equivalência de programas
- Executabilidade
- Correção coincidente

Limitações do Teste

Corretitude de programas

- Não existe algoritmo de teste de propósito geral que prove **corretitude** de um programa
  - 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 dados pertencntes ao seu domínio de entrada*

- Necessidade de um **Oráculo de teste**

Limitações do Teste

Corretitude de programas: oráculo de teste

> *É um testador ou algum mecanismo que posso determinar, para qualquer item de dado d pertencente ao domínio de entrada D, se $S(d) = P^*(d)$ dentro de limites de tempo e esforço razoáveis*

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

Limitações do Teste

Corretitude de programas: tipos de oráculo de teste

- Kiddie oracles
  - Execute o programa, se ela parecer correta, deve estar correta
- Conjunto de teste de regressão
  - Execute o programa e compare a saída obtida com a saída de uma versão mais antiga do programa
- Validação de dados
  - Execute o programa saída obtida com uma saída padrão determinada por uma tabela, fórmula, ou outra definição aceitável de saída válida

Limitações do Teste

Equivalência de Programas

- Dados dois programas, determinar se eles são **equivalentes**
- Dados dois caminhos (sequência 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 ítens de dados pertencentes ao domínio de entrada

Limitações do Teste

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-executavel**

Limitações do Teste

Correção coincidente

- O programa pode apresentar, **coincidentemente**, um cresultado 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

Terminologia de Erros

- Engano x Defeito x Erro x Falha
- Padrão IEEE 610.12-1990
  - Um **engano** introduz um defeito
  - O **defeito**, quando ativado, *pode* produzir um **erro**
  - O **erro**, *se propagado até a saída*, evidencia uma **falha**
    - Erro Computacional
    - Erro de D omínio

Erro Computacional (1/2)

- Provoca uma **computação incorreta** mas a sequência de comandos executada é **igual** à sequencia esperada

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

Erro Computacional (2/2)

```c
int blech(int x, int y){
  int z = x - y; // deveria ser x + y
  return (z);
}
```

- Exemplo:
  - Se $x = 1$ e $y = 0$
    - a saída esperada é 1
    - a saída obtida é 1
    - e o **erro computacional** não é revelado
  - Se $x = 1$ e $y = 1$
    - a saída esperada é 2
    - a saída obtida é 0
    - e o **erro computacional** é revelado
    - em verdade, o erro é revelado sempre que $y \neq 0$

Erro de Domínio

- Faz com que a sequência de comandos executada seja **diferente** da sequencia esperada
  - Um **sequência errada** é executada
- Causas:
  - Um erro computacional
  - Uma condição incorreta de um comando de decisão

Seleção de saídas incorretas de um comando de decisão corresponde a um erro de domínio

Casos de Teste

Casos de Teste

- O sucesso do teste está ligado aos **casos de teste**
  - Um bom caso de teste tem **alta probabiliadde** de revelar um erro ainda não descoberto
  - Também, casos de teste podem revelar especificações **incompletas** e **ambíguas**

Partes de um Caso de Teste

- Um caso de teste é:

> *Um 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*

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

Ordem de Execução (1/4)

- Existem dois estilos de projeto de casos de teste, considerando a sua **ordem de execução**
  - Casos de teste em cascata
  - Casos de teste independentes

Ordem de Execução (2/4)

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

Ordem de Execução (3/4)

Casos de Teste em Cascata

- Vantagens
  - Casos de testes tendem a ser pequenos e simples
  - Fáceis de serem projetados, criados e mantido.
- Desvantagens
  - Se um caso de teste falhar, os casos de teste subseqüentes também podem falhar.

Ordem de Execução (4/4)

Casos de Teste Independentes

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

Técnicas de Teste

Técnicas de Teste (1/2)

- 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*)
  - teste **caixa-branca** (*white-box testing*)
- Esses tipos de teste correspondem ao que chamamos de **técnicas de teste**

Técnicas de Teste (2/2)

- As técnicas de teste são definidas conforme o **tipo de informação** utilizado para realizar o teste
- Contemplam diferentes perspectivas do software
  - Portanto, as técnicas são **complementares**
  - 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

Criterios de Teste (1/4)

- Maneira sistemática e planejada para conduzir os testes
- Fornece indicações a respeito de **quais casos de teste usar**
  - De modo a aumentar as chances de revelar erros
- Quando erros não forem revelados...
  - Estabelece um nível elevado de **confiança** na correção do programa

Criterios de Teste (2/4)

- 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**

Criterios de Teste (3/4)

Criterios de Geração

*Procedimento para **escolher** casos de teste para o teste do programa P*

- O conjunto T, formado pelos casos de teste escolhido, é, portanto, $C - adequado$ **por construção**

Criterios de Teste (4/4)

Criterios 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 cojunto de casos de teste T satisfaz os requisitos de teste estabelecidos pelo critério

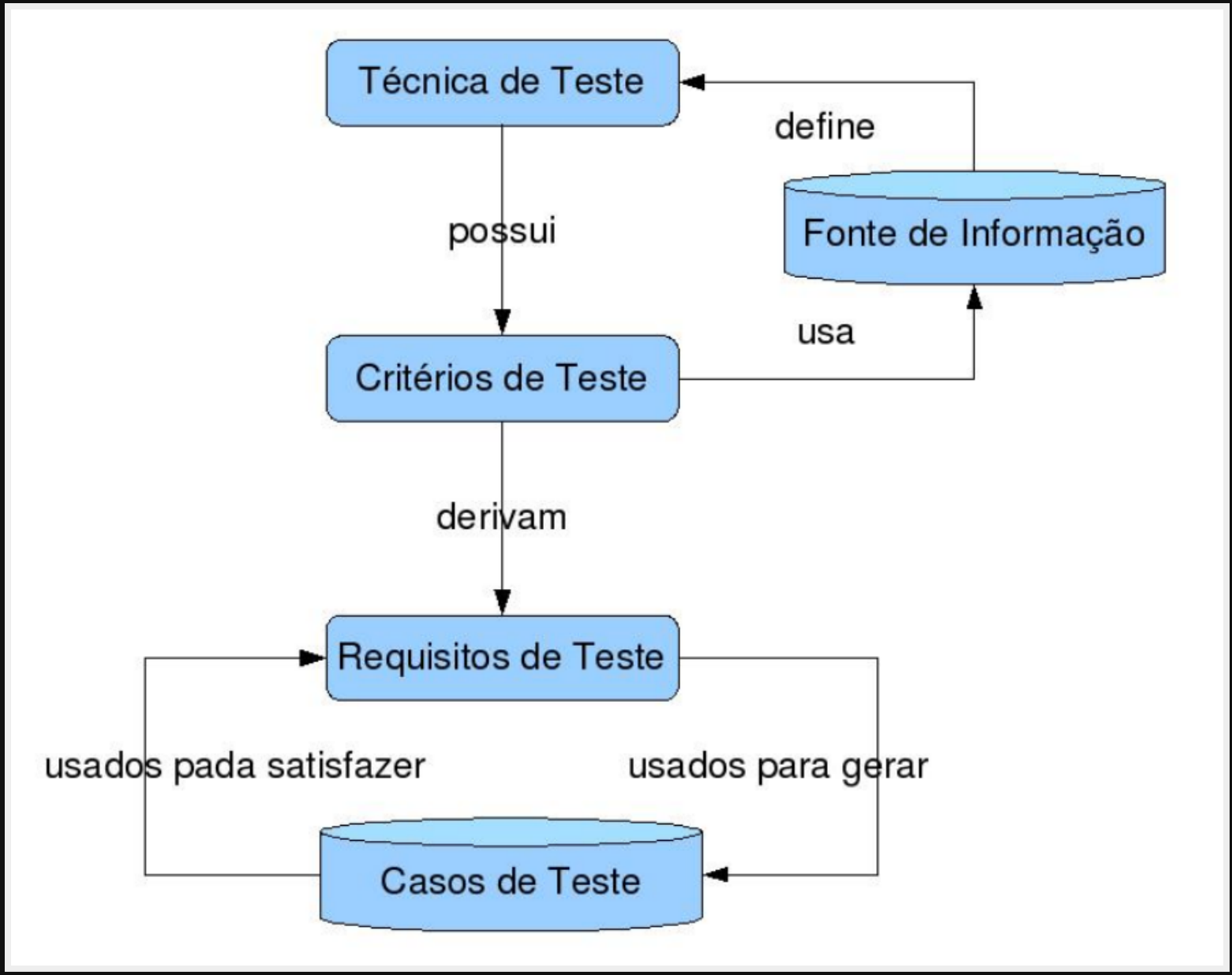
Análise de Cobertura

- 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 caos de teste ao conjunto

Requisitos de Teste

- Cada critério estabelece um cojunto 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



Considerações Finais

Pontos importantes

- A atividade de teste é um processo executado em **paralelo** com as demais atividades de desenvolvimento
- O ponto central da atividade de teste é o **projeto de casos de teste**
  - Selecionar **casos de teste** com a maior probabilidade de revelar **erros**
- Diferentes **técnicas** e **critérios** de teste são complementares
- Teste devem ser executados em fases, para reduzir a complexidade