



Teste Baseado em Erros

PPGCC12-Teste de Software

Reginaldo Ré

Aula 08



Introdução



Introdução

Em aulas anteriores

- As técnicas de teste são definidas conforme o tipo de informação utilizada para realizar o teste.
- Contemplam diferentes perspectivas do software:
 - Aspecto complementar!!!!
 - Técnica Funcional
 - Os testes são baseados exclusivamente na especificação de requisitos do programa.
 - Nenhum conhecimento de como o programa está implementado é requerido.
 - Técnica Estrutural
 - Os testes são baseados exclusivamente na especificação de requisitos do programa.

Introdução

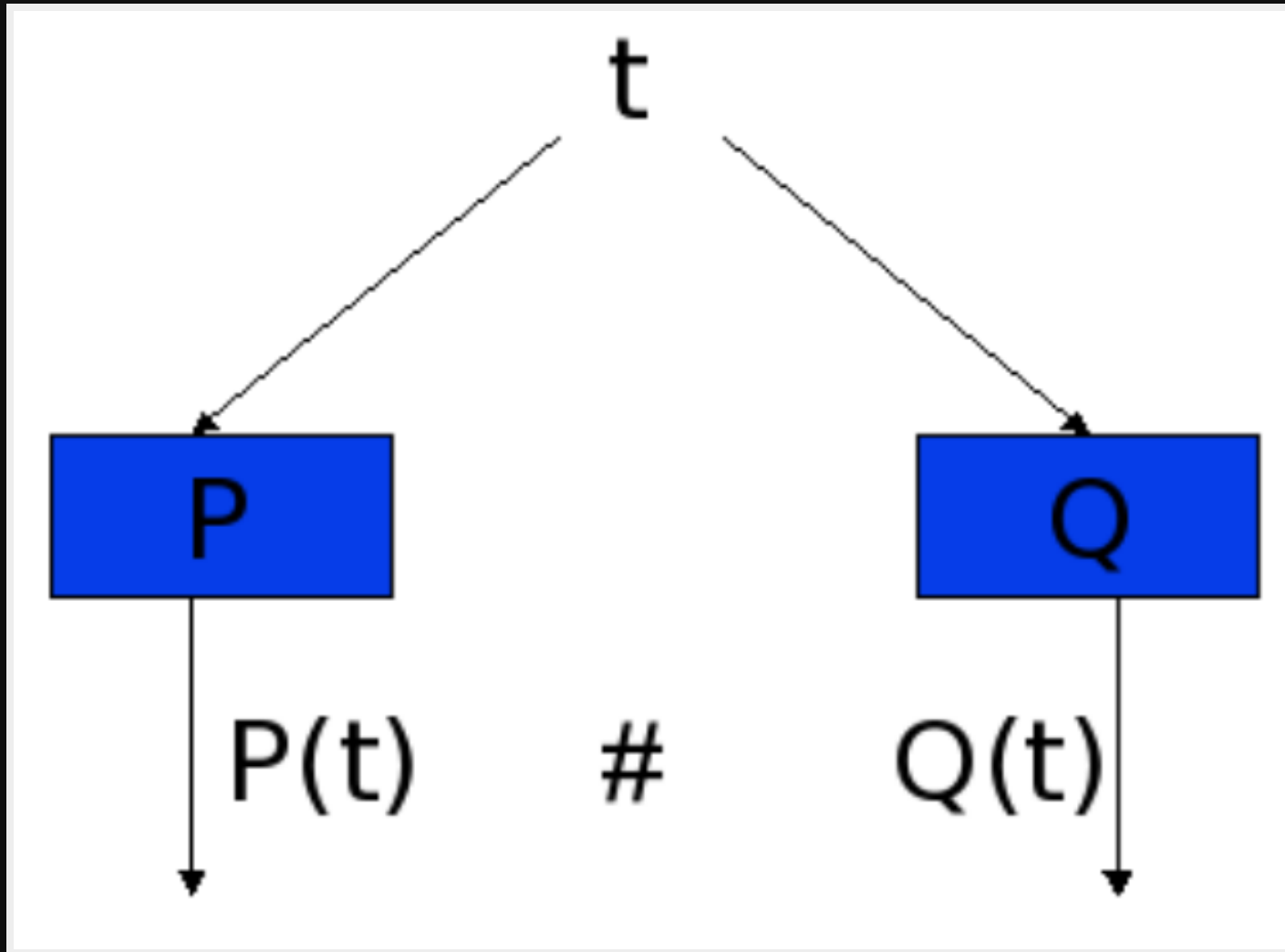
Técnica de teste baseada em erros

- Os requisitos de teste são derivados a partir dos erros mais freqüentes cometidos durante o processo de desenvolvimento do software.- Técnica flexível, podendo ser aplicada em diferentes contextos
- Técnica flexível, podendo ser aplicada em diferentes contextos
 - Modelos e especificações, por exemplo
- Teste de mutação e semeadura de erros
 - Teste de mutação é o critério mais conhecido

Critério Análise de Mutantes

Critério Análise de Mutantes

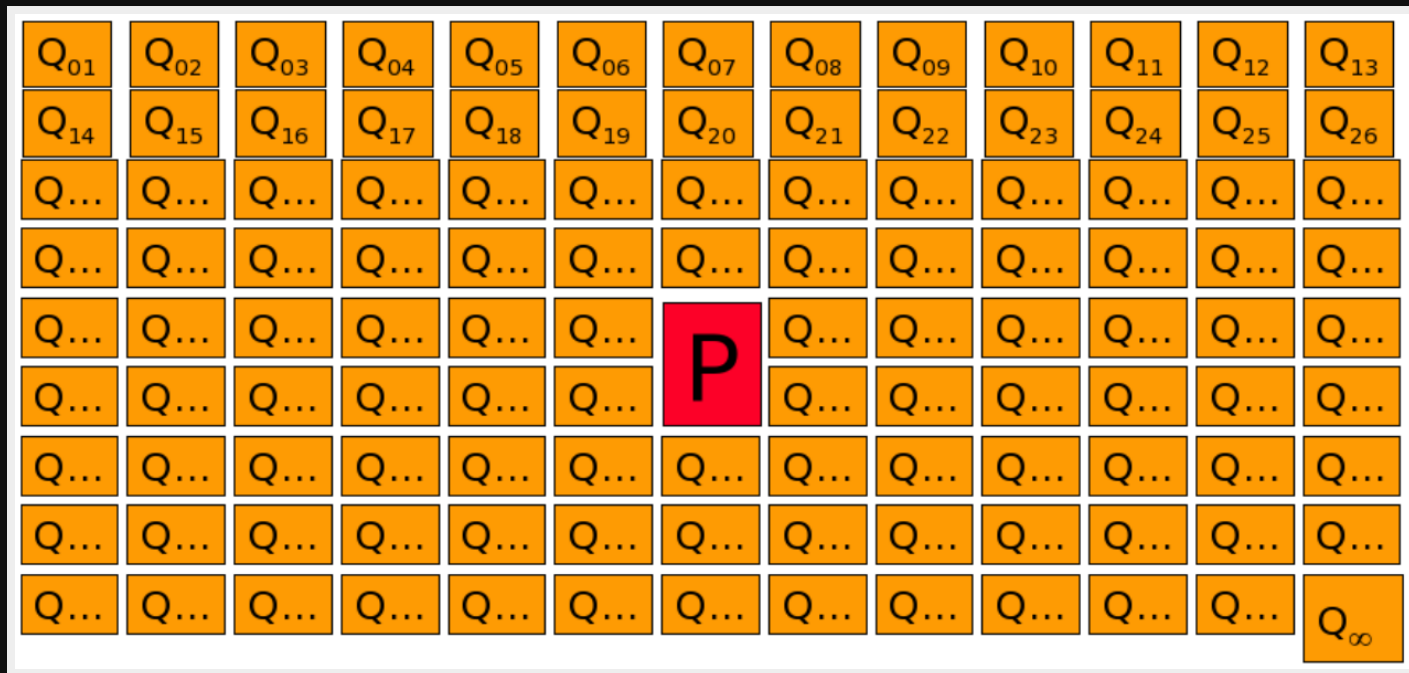
- A idéia é mostrar que P não possui determinados tipos de defeitos



Critério Análise de Mutantes

Correção Absoluta

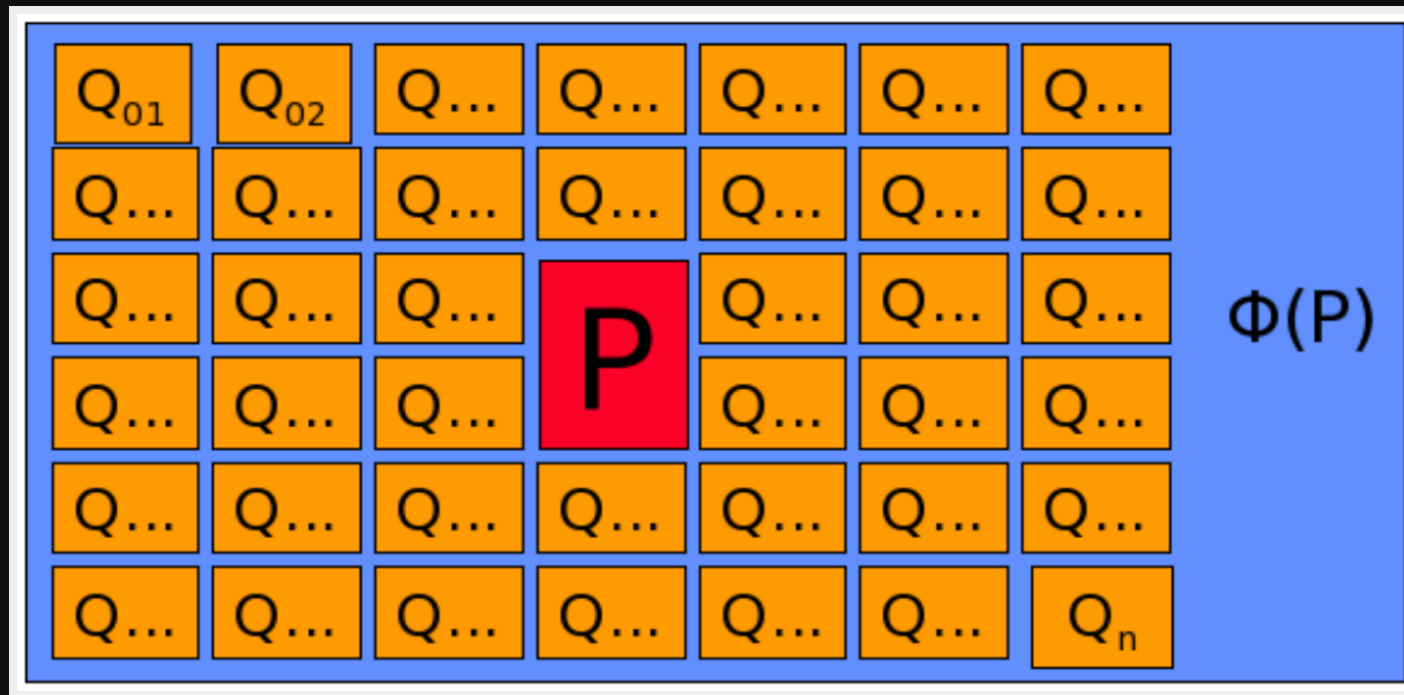
- Considerando todos os programas Q , é possível provar a correção de P



Critério Análise de Mutantes

Correção Relativa

- A vizinhança é infinita
- Impossível executar e comparar cada programa Q_i
- Estabelece-se então uma vizinhança $\phi(P)$ finita
 - Tais programas possuem apenas pequenos desvios sintáticos que representam defeitos simples



Critério Análise de Mutantes

Pontos Importantes

- Introdução de pequenos desvios sintáticos
 - Que representam erros frequentes
 - O testador tenta mostrar que os erros sintáticos resultam e erros semânticos
 - Evidenciar diferenças no comportamento entre P e Q

Critério Análise de Mutantes

Hipótese do Programador Competente

Programadores experientes escrevem programas corretos ou muito próximos do correto

Critério Análise de Mutantes

Efeito de Acoplamento

Casos de teste capazes de revelar erros simples são tão sensíveis que, implicitamente, também são capazes de revelar erros complexos.

Passos para a aplicação do critério Análise de Mutantes

1. Geração dos Mutantes
2. Execução do Programa
3. Execução dos Mutantes
4. Análise dos Mutantes Vivos

1. Geração dos Mutantes

Passos para a aplicação do critério Análise de Mutantes

- Para modelar os desvios sintáticos mais comuns usa-se **operadores de mutação**
 - Dependem da linguagem alvo
 - Definem as regras das alterações a serem aplicadas a P
 - A seleção de operadores de mutação devem ser:
 - Abrangentes: capaz de modelar a maior parte dos erros
 - Pequena cardinalidade: quanto maior o número de operadores, maior o número de mutantes gerados
 - Aumento do custo da aplicação do critério
- Existem também mutantes instrumentados
 - Que não modelam defeitos típicos

1. Geração dos Mutantes

Exemplos de operadores de mutação

Mutação de Comandos

SSDL Retira um comando de cada vez do programa

SWDD Troca o comando *while* por *do-while*

SMTC Interrompe a execução do laço após duas execuções

Mutação de Operadores

ORRN Troca operador relacional por operador relacional

OLBN Troca operador lógico por operador *bitwise*

OASN Troca operador aritmético por operador de deslocamento

Mutação de Constantes

Ccsr Troca referências escalares por constantes

Cccr Troca constante por constante

Mutação de Variáveis

VTWD Troca referência escalar pelo sucessor e predecessor

VDTR Requer valor negativo, positivo e zero para cada referência escalar

1. Geração dos Mutantes

O Programa Mutante

Programa modificado, resultante da aplicação dos operadores de mutação sobre o programa original.


- Apenas uma mutação de cada vez
 - Ou seja, apenas uma transformação sintática
 - Devido ao efeito de acoplamento
 - Mas pode-se usar K transformações sintáticas
 - K -mutante

1. Geração dos Mutantes

Exemplo de programa mutante

- Operador de mutação ORRN
 - Troca do operador relacional $<$ pelo operador relacional $<=$

```
{
    char  achar;
    int length, valid_id;
    length = 0;
    printf ("Identificador: ");
    achar = fgetc (stdin);
    valid_id = valid_s(achar);
    if (valid_id)
        length = 1;
    achar = fgetc (stdin);
    while (achar != '\n')
    {
        if (!(valid_f(achar)))
            valid_id = 0;
        length++;
        achar = fgetc (stdin);
    }
    if (valid_id && (length >= 1) && (length <= 6))
        printf ("Valido\n");
    else
        printf ("Invalido\n");
}
```



Programa *Identifier* (função main)

2. Execução do Programa

- O programa é executado com os casos de teste

3. Execução dos Mutantes

- Execução dos mutantes com os casos de teste
 - Mutantes mortos
 - Mutantes vivos

3. Execução dos Mutantes

Mutantes Mortos

*O resultado do mutante e o do programa original
diferem entre si para algum caso de teste*

- Significa que o erro modelado pelo operador de mutação **não** está presente no programa original.

4. Análise dos Mutantes Vivos

- Análise dos mutantes vivos
 - Mutante equivalente
 - Inclusão de novos casos de teste
 - Escore de mutação
 - Medida de cobertura do teste de mutação!

4. Análise dos Mutantes Vivos

Mutantes Vivos: equivalente

*O mutante e o programa original apresentam **sempre** o mesmo resultado, para qualquer caso de teste pertencente ao domínio de entrada*

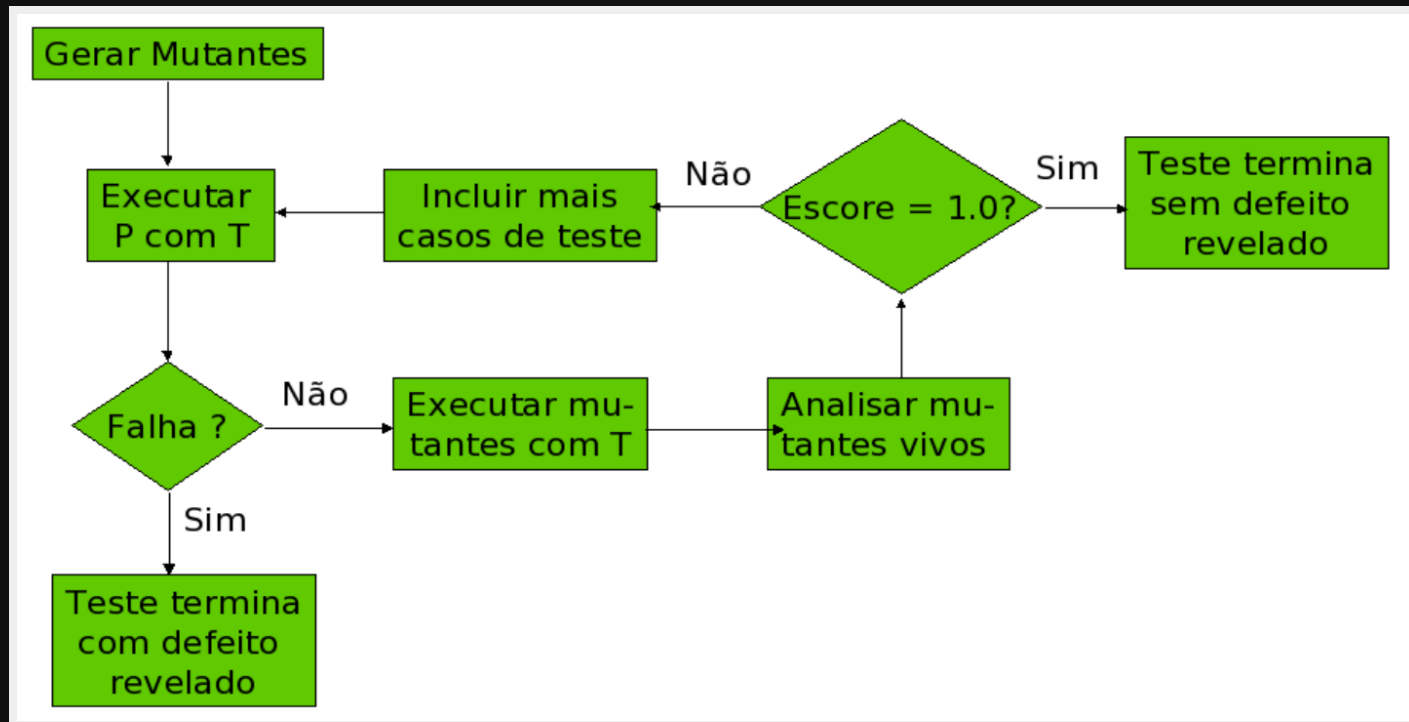
- A equivalência é um problema indencidível

4. Análise dos Mutantes Vivos

Mutantes Vivos: escore de mutação

- Medida objetiva a respeito do nível de confiança da adequação dos casos de teste utilizados.
- Varia no intervalo entre 0 e 1.
 - Quanto **maior** o escore, mais adequado é o conjunto de casos de teste.
 - $EM(P, T) = \frac{MM(P, T)}{MG(P) - ME(P)}$, no qual:
 - EM é o escore de mutação
 - $MM(P, T)$ é o total de mutantes mortos pelo conjunto de casos de teste T
 - $MG(P)$ é o total de mutantes gerados a partir do programa P
 - $ME(P)$ é o total de mutantes equivalentes ao programa P

Resumo da Aplicação do Critério



Considerações sobre a Análise de Mutantes

- Alta eficácia em revelar presença de erros.
- Alto custo de aplicação.
 - Grande número de mutantes gerados
 - Equivalência entre programas
 - Abordagens alternativas:
 - Mutação aleatória: uma porcentagem limitada de mutantes é considerada
 - Mutação seletiva: operadores de mutação que geram muitos mutantes não são aplicados
 - Mutação restrita: operadores de mutação específicos são escolhidos
 - Mutação restrita ao conjunto essencial de operadores: escolha sistemática de operadores de mutação
 - Mutação de interface: teste de integração

Considerações Finais



Pontos importantes

Técnica de Teste Baseada em Mutação

- Erros mais frequentes
 - Operadores de mutação
 - Mutantes: vivos, mortos e equivalentes
 - Escore de mutação
- Aplicabilidade
 - Programa: unidade e interface
 - Especificações
- Limitações
 - Alto custo de aplicação
 - Abordagens alternativas
- Ferramentas