

Teste de Software

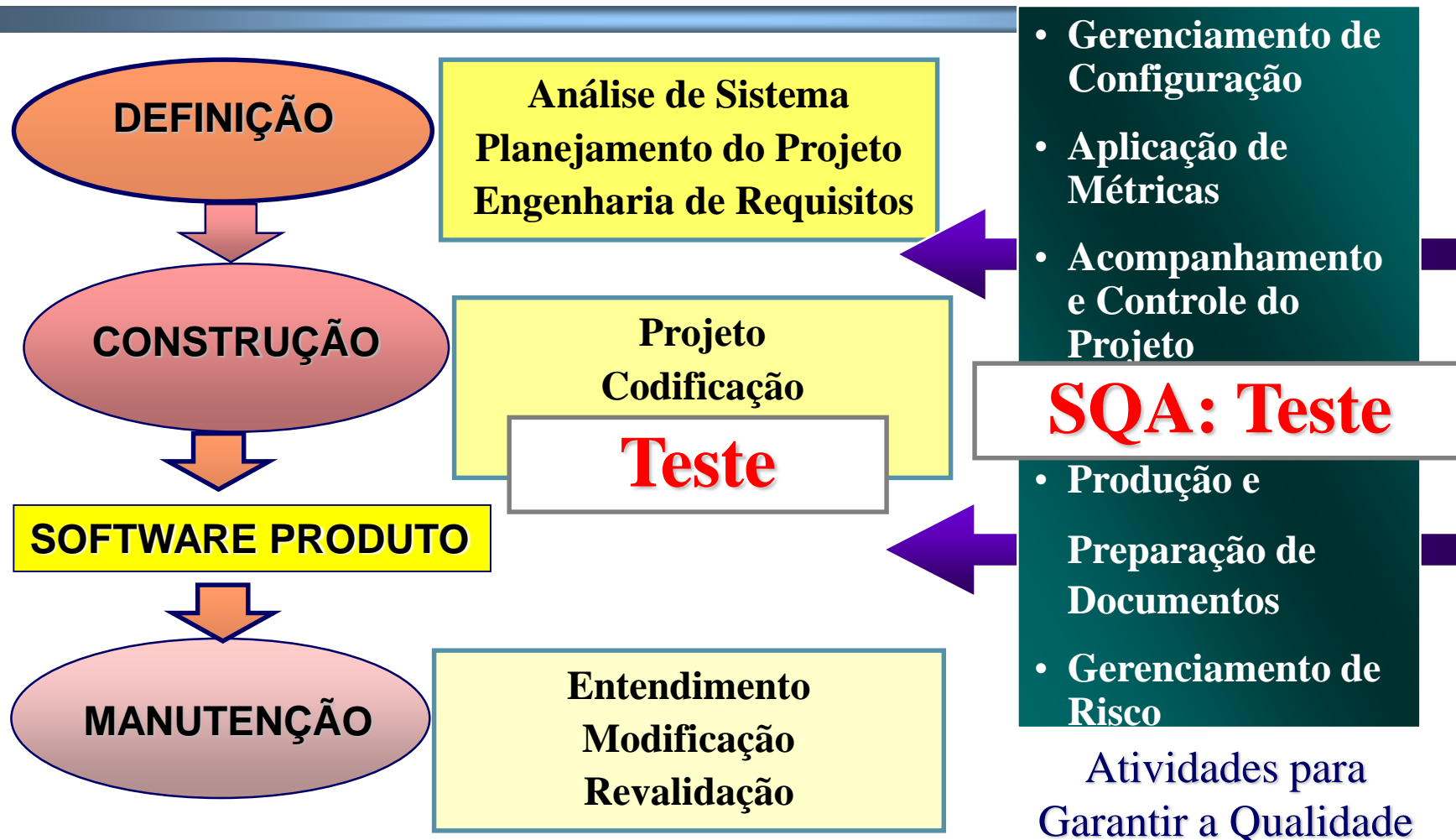
Two horizontal bars are positioned below the title. The top bar is a solid blue line, and the bottom bar is a solid grey line. Both bars span most of the width of the slide.

Simone Souza

Disciplina: Engenharia de Software
ICMC/USP

O Processo de Engenharia de Software

Atividades Genéricas



Introdução

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





Introdução

- 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

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

Exemplo

- Observe o exemplo a seguir:

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

Exemplo

- Observe o exemplo a seguir:

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

- Haverá tempo suficiente para se criar **65.536 casos de teste**?
 - E para programas maiores? Quantos casos de teste serão necessários?

Exemplo

- Quais valores escolher ?

Entrada	Saída esperada	Saída obtida
1	0	0
42	0	0
32000	1	1
-32000	-1	-1

Não revelam o erro !

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


Exemplo

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

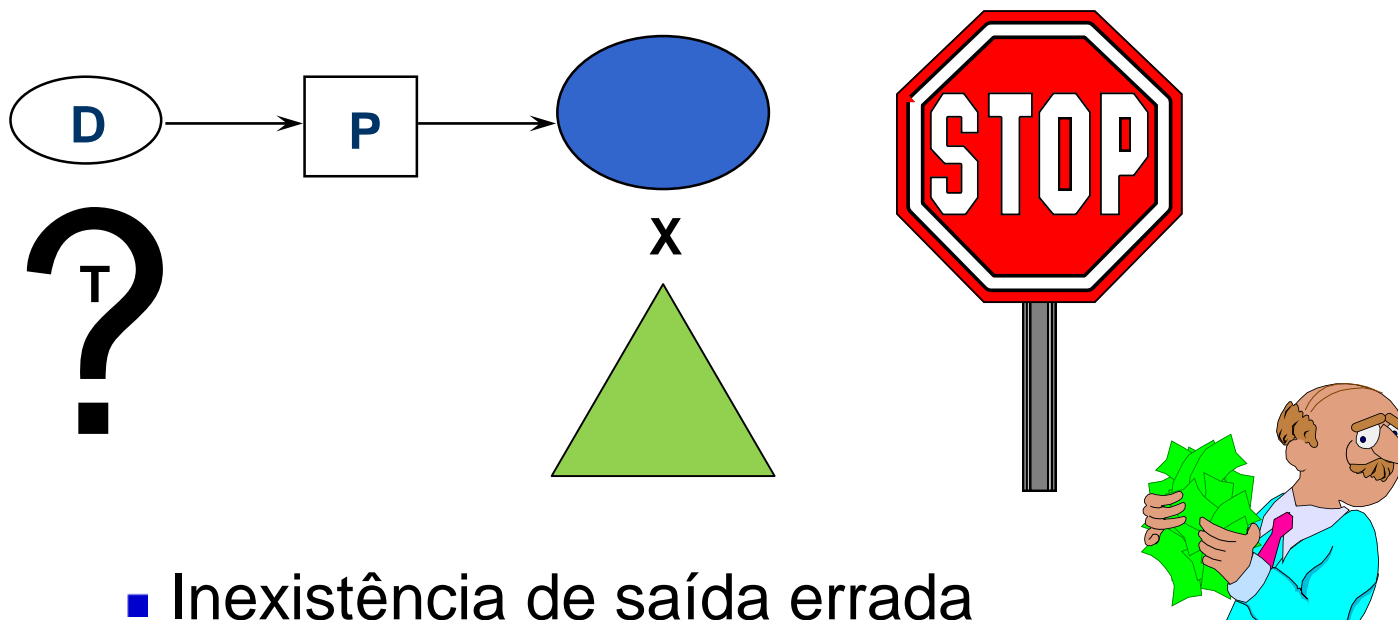
Entrada	Saída esperada	Saída obtida
-30000	0	-1
-29999	0	-1
30000	1	0
29999	1	0

Qual a chance de tais valores serem selecionados???

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

Objetivo do Teste

■ Revelar a presença de defeito



■ Inexistência de saída errada

- Software é de alta qualidade?
- T é de baixa qualidade?

Exemplo: gerar casos de teste para testar o programa TRI

- O programa TRI lê três valores inteiros que representam os lados de um triângulo. A partir dos valores, o programa informa se os lados formam um triângulo isósceles, escaleno ou equilátero.
- Condição: a soma de dois lados tem que ser maior que o terceiro lado.

Exemplo

1. Existe c.t. para triângulo escaleno válido?
2. Existe c.t. para triângulo isósceles válido?
3. Existe c.t. para triângulo equilátero válido?
4. Existem pelo menos 3 c.t. para isósceles válido contendo a permutação dos mesmos valores?
5. Existe c.t. com um valor zero?
6. Existe c.t. com um valor negativo?
7. Existe c.t. em que a soma de 2 lados é igual ao terceiro lado?
8. Para o item 7, tem um c.t. para cada permutação de valores?

Exemplo

9. Existe c.t. em que a soma de 2 lados é menor que o terceiro lado?
10. Para o item 9, tem um c.t. para cada permutação de valores?
11. Existe c.t. para os 3 valores iguais a zero?
12. Existe c.t. com valores não inteiros?
13. Existe c.t. com número de valores errados, por exemplo, 2 valores ao invés de 3?
14. Para cada c.t. você especificou a saída esperada para a entrada projetada?

Exemplo

- Questões baseadas em erros encontrados em implementações do programa TRI!!!

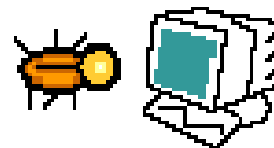
Teste de Software

- A qualidade da atividade de teste está fortemente ligada à qualidade do conjunto de casos de teste.

- *Como selecionar casos de teste?*

Teste de Software

- Etapas de teste
- Técnicas de teste
 - Critérios de teste
- Fases de teste

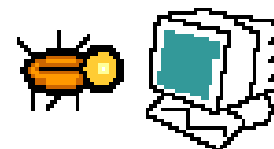


Etapas de Teste

- Planejamento.
 - Desenvolvimento **Plano de Testes.**
- Projeto de casos de teste.
 - Seleção e aplicação de critérios.
- Execução do programa.
- Análise de resultados.

Técnicas de Teste

- Teste Funcional
- Teste Estrutural
- Teste Baseado em Defeitos



Teste Funcional

- Funcional (Caixa Preta)
 - Os testes são baseados exclusivamente na **especificação do programa**.
 - Nenhum conhecimento de como o programa está implementado é requerido.
 - Não precisa ter o programa para derivar os testes

- Principais critérios:
 - Particionamento em classes de equivalência.
 - Análise do valor limite.

Exemplo: Teste Funcional

O programa deve determinar se um identificador é válido ou não em Silly Pascal. Um identificador válido deve começar com uma letra e conter apenas letras ou dígitos. Além disso, deve ter no mínimo 1 caractere e no máximo 6 caracteres de comprimento.

– Exemplo:

abc12 (válido);
cont*1 (inválido);

1soma (inválido);
a123456 (inválido)

Exemplo: Teste Funcional

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

■ Exemplo de Conjunto de Casos de Teste:

- $T_0 = \{(a1, \text{Válido}), (2B3, \text{Inválido}), (Z-12, \text{Inválido}), (A1b2C3d, \text{Inválido}), \emptyset\}$

```

/* 01 */ {
/* 01 */     char  achar;
/* 01 */     int   length, valid_id;
/* 01 */     length = 0;
/* 01 */     printf ("Digite identificador\n");
/* 01 */     printf ("seguido por <ENTER>: ");
/* 01 */     achar = fgetc (stdin);
/* 01 */     valid_id = valid_starter (achar);
/* 01 */     if (valid_id)
/* 02 */         length = 1;
/* 03 */     achar = fgetc (stdin);
/* 04 */     while (achar != '\n')
/* 05 */     {
/* 05 */         if (!(valid_follower (achar)))
/* 06 */             valid_id = 0;
/* 07 */         length++;
/* 07 */         achar = fgetc (stdin);
/* 07 */     }
/* 08 */     if (!valid_id)
/* 09 */         printf ("Invalido\n");
/* 10 */     printf ("Valido\n");
/* 11 */ }

```

O conjunto T_0 é suficiente para testar o programa?

- $T_0 = \{(a1, \text{Válido}), (2B3, \text{Inválido}), (Z-12, \text{Inválido}), (A1b2C3d, \text{Inválido}), \emptyset\}$

Execução do caso de teste: (a1,Válido)

```

/* 01 */ {
/* 01 */     char  achar;
/* 01 */     int   length, valid_id;
/* 01 */     length = 0;
/* 01 */     printf ("Digite identificador\n");
/* 01 */     printf ("seguido por <ENTER>: ");
/* 01 */     achar = fgetc (stdin);
/* 01 */     valid_id = valid_starter
/* 01 */     if (valid_id)
/* 02 */         length = 1;
/* 03 */     achar = fgetc (stdin);
/* 04 */     while (achar != '\n')
/* 05 */     {
/* 05 */         if (!(valid_follower (achar)))
/* 06 */             valid_id = 0;
/* 07 */         length++;
/* 07 */         achar = fgetc (stdin);
/* 07 */     }
/* 08 */     if (valid_id && (length >= 1) && (length < 6) )
/* 09 */         printf ("Valido\n");
/* 10 */     else
/* 10 */         printf ("Invalido\n");
/* 11 */ }

```

Alguns comandos
não foram
executados!

Execução do caso de teste: (2B3, Inválido)

```

/* 01 */ {
/* 01 */     char  achar;
/* 01 */     int   length, valid_id;
/* 01 */     length = 0;
/* 01 */     printf ("Digite identificador\n");
/* 01 */     printf ("seguido por <ENTER>: ");
/* 01 */     achar = fgetc (stdin);
/* 01 */     valid_id = valid_starter
/* 01 */     if (valid_id)
/* 02 */         length = 1;
/* 03 */     achar = fgetc (stdin);
/* 04 */     while (achar != '\n')
/* 05 */     {
/* 05 */         if (!(valid_follow
/* 06 */             valid_id =
/* 07 */             length++;
/* 07 */             achar = fgetc (stdin);
/* 07 */     }
/* 08 */     if (valid_id && (length >= 1) && (length < 6) )
/* 09 */         printf ("Valido\n");
/* 10 */     else
/* 10 */         printf ("Invalido\n");
/* 11 */ }

```

O comando em /*06*/ não foi executado pelos casos de teste.

Execução do caso de teste: (Z-12, Inválido)

```

/* 01 */ {
/* 01 */     char  achar;
/* 01 */     int   length, valid_id;
/* 01 */     length = 0;
/* 01 */     printf ("Digite identificador\n");
/* 01 */     printf ("seguido por <ENTER>: ");
/* 01 */     achar = fgetc (stdin);
/* 01 */     valid_id = valid_starter (achar);
/* 01 */     if (valid_id)
/* 02 */         length = 1;
/* 03 */     achar = fgetc (stdin);
/* 04 */     while (achar != '\n')
/* 05 */     {
/* 05 */         if (!(valid_follower (achar)))
/* 06 */             valid_id = 0;
/* 07 */         length++;
/* 07 */         achar = fgetc (stdin);
/* 07 */     }
/* 08 */     if (valid_id && (length >= 1) && (length < 6) )
/* 09 */         printf ("Valido\n");
/* 10 */     else
/* 10 */         printf ("Invalido\n");
/* 11 */ }

```

```

/* 01 */ {
/* 01 */     char  achar;
/* 01 */     int   length, valid_id;
/* 01 */     length = 0;
/* 01 */     printf ("Digite identificador\n");
/* 01 */     printf ("seguido por <ENTER>: ");
/* 01 */     achar = fgetc (stdin);
/* 01 */     valid_id = valid_starter (achar);
/* 01 */     if (valid_id)
/* 02 */         length = 1;
/* 03 */     achar = fgetc (stdin);
/* 04 */     while (achar != '\n')
/* 05 */     {
/* 05 */         if (!(valid_follower (achar)))
/* 06 */             valid_id = 0;
/* 07 */         length++;
/* 07 */         achar = fgetc (s
/* 07 */     }
/* 08 */     if (valid_id && (length
/* 09 */         printf ("Valido\
/* 10 */     else
/* 10 */         printf ("Invalid
/* 11 */ }

```

Todos os comandos foram executados e as saídas esperadas foram obtidas!

Pode-se afirmar que o programa está correto?

```

/* 01 */ {
/* 01 */     char  achar;
/* 01 */     int   length, valid_id;
/* 01 */     length = 0;
/* 01 */     printf ("Digite identificador\n");
/* 01 */     printf ("seguido por <ENTER>: ");
/* 01 */     achar = fgetc (stdin);
/* 01 */     valid_id = valid_starter (achar);
/* 01 */     if (valid_id)
/* 02 */         length = 1;
/* 03 */     achar = fgetc (stdin);
/* 04 */     while (achar != '\n')
/* 05 */     {
/* 05 */         if (!(valid_follower (achar)))
/* 06 */             valid_id = 0;
/* 07 */         length++;
/* 07 */         achar = fgetc (stdin);
/* 07 */     }
/* 08 */     if (valid_id && (length >= 1) && (length < 6) )
/* 09 */         printf ("Valido\n");
/* 10 */     else
/* 10 */         printf ("Invalido\n");
/* 11 */ }

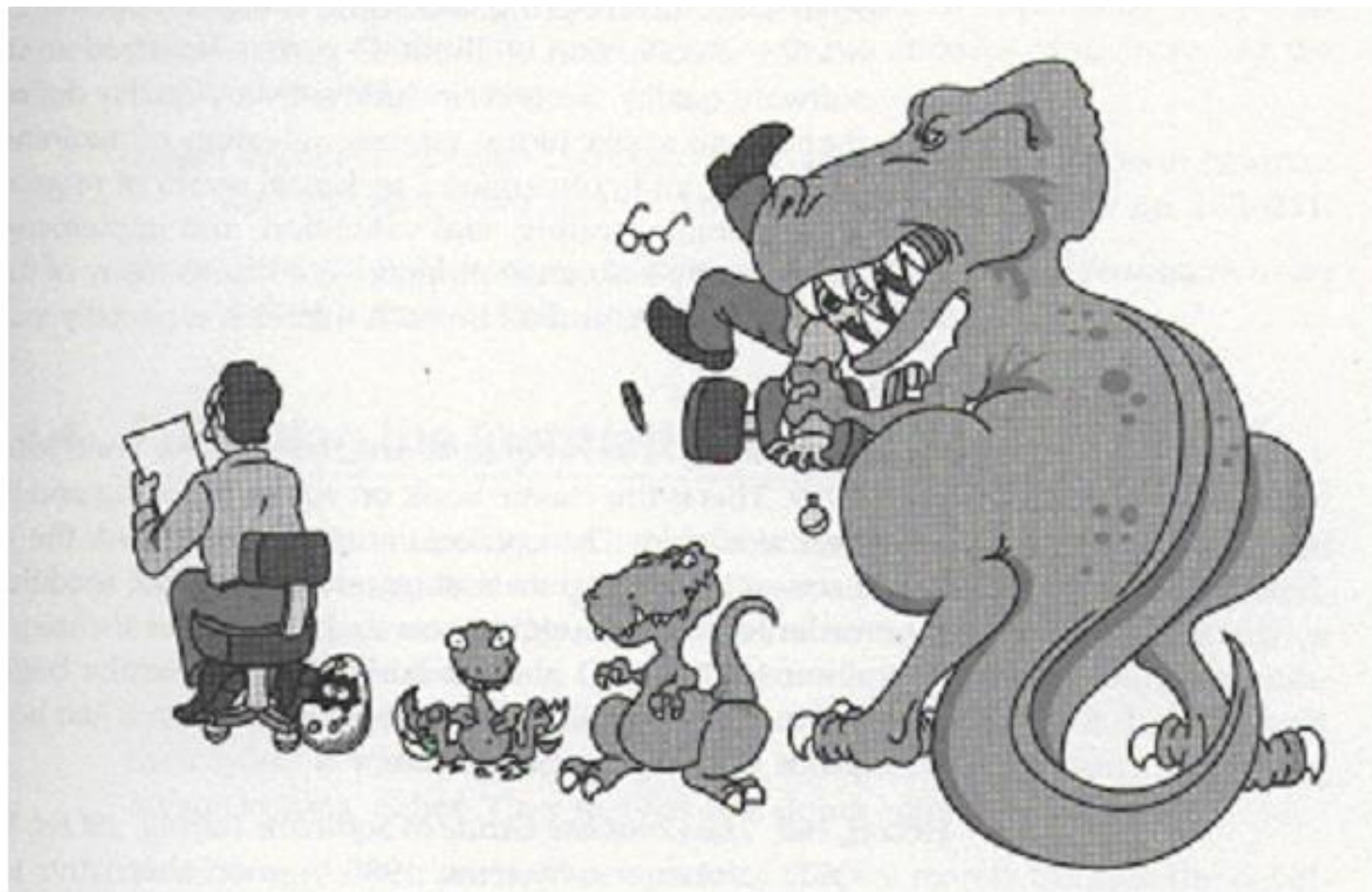
```

Existe um defeito não detectado!!!!!!!

```

/* 01 */ {
/* 01 */     char  achar;
/* 01 */     int   length, valid_id;
/* 01 */     length = 0;
/* 01 */     printf ("Digite identificador\n");
/* 01 */     printf ("seguido por <ENTER>: ");
/* 01 */     achar = fgetc (stdin);
/* 01 */     valid_id = valid_starter (achar);
/* 01 */     if (valid_id)
/* 02 */         length = 1;
/* 03 */     achar = fgetc (stdin);
/* 04 */     while (achar != '\n')
/* 05 */     {
/* 05 */         if (!(valid_follower (achar)))
/* 06 */             valid_id = 0;
/* 07 */         length++;
/* 07 */         achar = fgetc (stdin);
/* 07 */     }
/* 08 */     if (valid_id && (length >= 1) && (length < 6) )
/* 09 */         printf ("Valido\n");
/* 10 */     else
/* 10 */         printf ("Invalido\n");
/* 11 */ }

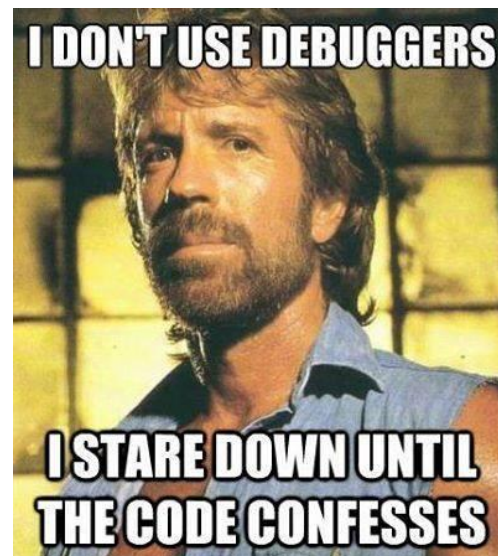
```



The longer defects remain undetected, the longer they take to fix . Correct defects when they are young and easy to control!

Teste de Software

- Não é possível garantir que o software está livre de defeitos!!!
 - Permite **aumentar a confiança** de que, para os valores testados, o programa está correto!



Técnicas de Teste

- Teste Funcional
- Teste Estrutural
- Teste Baseado em Erros

Teste Estrutural

- Estrutural (Caixa Branca)
 - Os testes são baseados na estrutura interna do programa, ou seja, na **implementação** do mesmo.

- Critérios de Fluxo de Controle
 - Testar todos os comandos.
 - Testar todos desvios condicionais.
 - Testar todos os caminhos básicos.

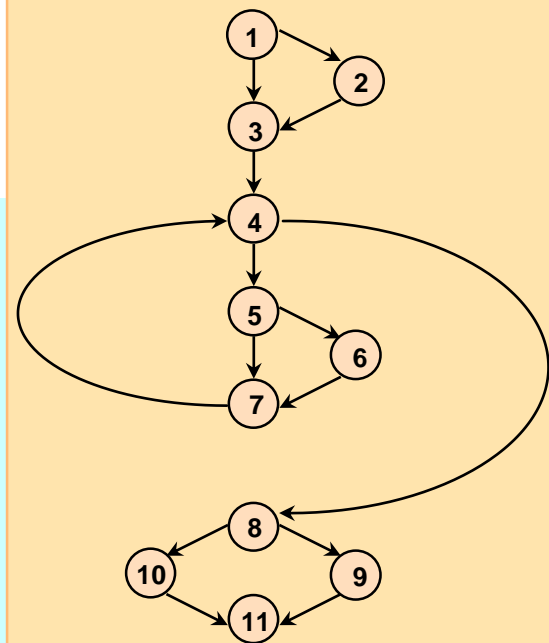
- Critérios de Fluxo de Dados
 - Testar todo uso a partir de uma definição de variável.

Exemplo: Teste Estrutural

```

/* 01 */ {
/* 01 */     char  achar;
/* 01 */     int   length, valid_id;
/* 01 */     length = 0;
/* 01 */     printf ("Digite identificador\n");
/* 01 */     printf ("seguido por <ENTER>: ");
/* 01 */     achar = fgetc (stdin);
/* 01 */     valid_id = valid_starter (achar);
/* 01 */     if (valid_id)
/* 02 */         length = 1;
/* 03 */     achar = fgetc (stdin);
/* 04 */     while (achar != '\n')
/* 05 */     {
/* 05 */         if (!(valid_follower (achar)))
/* 06 */             valid_id = 0;
/* 07 */         length++;
/* 07 */         achar = fgetc (stdin);
/* 07 */     }
/* 08 */     if (valid_id && (length >= 1) && (length < 6) )
/* 09 */         printf ("Valido\n");
/* 10 */     else
/* 10 */         printf ("Invalido\n");
/* 11 */ }

```

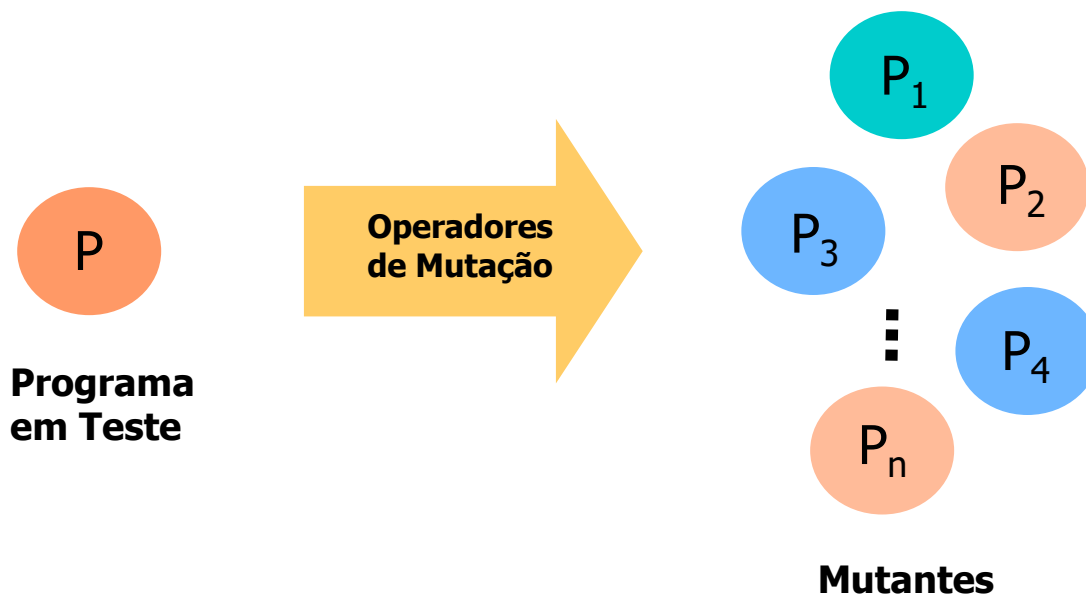


Técnicas de Teste

- Teste Funcional
- Teste Estrutural
- Teste Baseado em Defeitos

Teste Baseado em Defeitos

- Baseado nos **defeitos mais comuns** cometidos pelos programadores.
- Critério Análise de Mutantes



Exemplo: Teste Baseado em Defeitos

```
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"); }
```

Programa Fatorial

```
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"); }
```

Mutante Morto

Exemplo: Teste Baseado em Defeitos

```
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"); }
```

Programa Fatorial

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

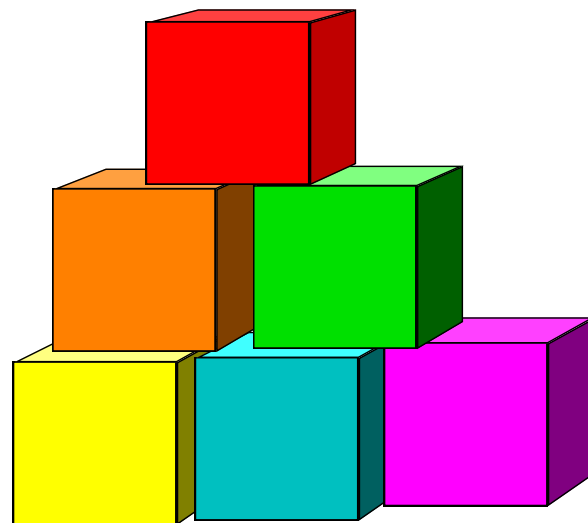
Mutante Equivalente

Teste Baseado em Defeitos

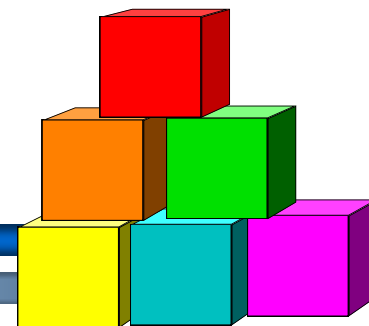
- Teste de mutação
 - Um dos melhores critérios de teste
 - Alto custo de aplicação
 - Necessidade de ferramenta de apoio
 - Necessidade de estratégias de teste

Fases do Teste

- Teste de Unidade
- Teste de Integração
- Teste de Sistema



Teste de Unidade

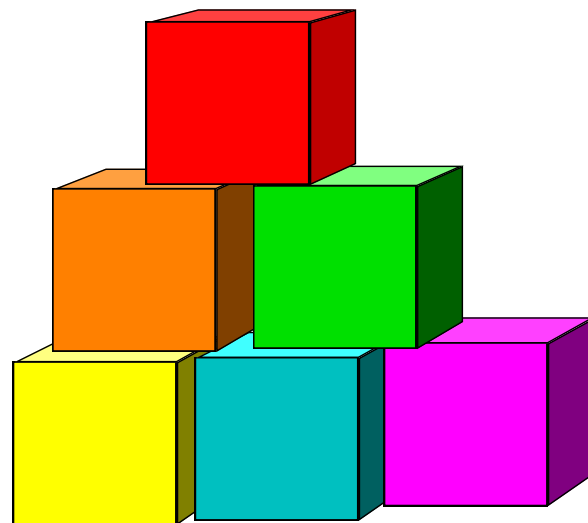


- Cada **unidade** do programa é explorado separadamente...
 - Identificar erros de lógica e de implementação.

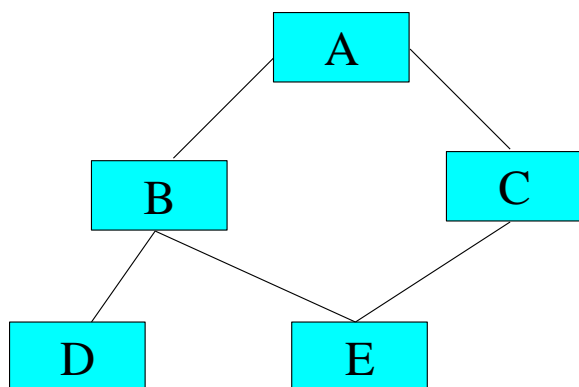
- Critérios de teste utilizados:
 - **Critérios de Fluxo de Controle**
 - **Critérios de Fluxo de Dados**
 - **Critério Análise de Mutantes**

Fases do Teste

- Teste de Unidade
- Teste de Integração
- Teste de Sistema

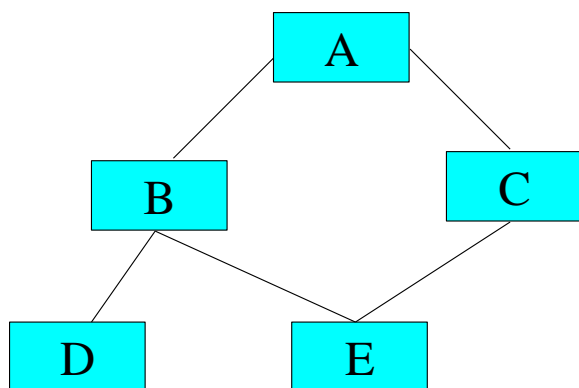
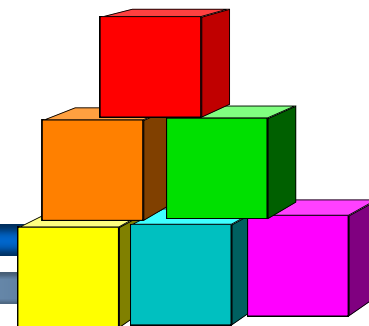


Teste de Integração



- Testar a comunicação entre as unidades funcionais do sistema
- Formas de comunicação:
 - **parâmetros de entrada.**
 - **Retorno** por meio de parâmetros de entrada (**passagem por referência**).
 - **Variáveis globais.**
 - **Comandos *return*.**

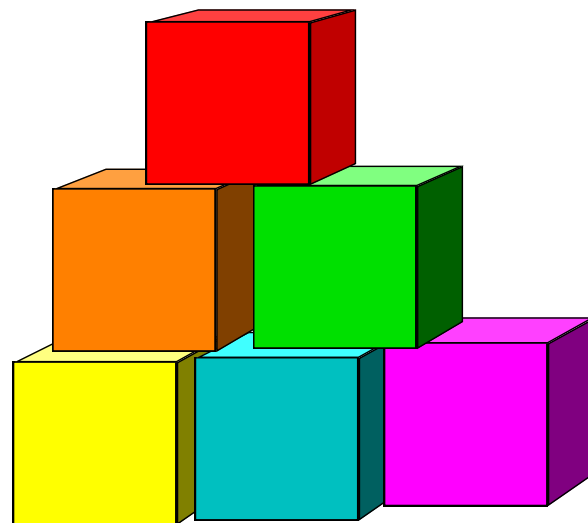
Teste de Integração



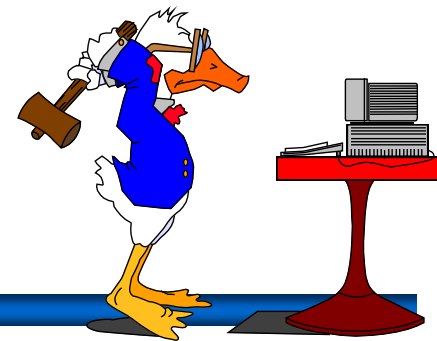
- Como testar a conexão entre A-B:
 - Testar pontos em que A chama B
 - Testar pontos dentro de B que usam dados enviados por A (relacionados com as variáveis de comunicação).

Fases do Teste

- Teste de Unidade
- Teste de Integração
- Teste de Sistema



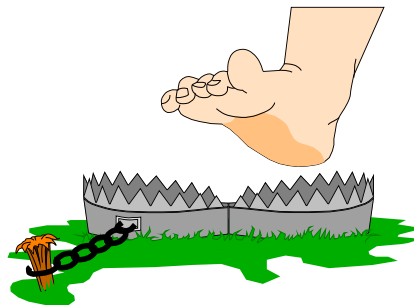
Teste de Sistema



- Após integrar (e testar!) todos os componentes do software...
 - O software deve ser capaz de funcionar apropriadamente no ambiente para o qual foi projetado.
- Apropriado para testar requisitos não funcionais

Exemplo de testes de sistema

- Teste de Recuperação e Estresse
- Teste de Segurança
- Teste de Desempenho
- Teste de Usabilidade



Exercício – parte 1

- Considere uma função que recebe como parâmetros um *vetor* de número inteiros e o tamanho t do *vetor* e retorna o vetor ordenado.
- Gere um conjunto de casos de teste, baseado no teste funcional.

Exercício – parte 2

- Analise se o conjunto de casos de teste gerado anteriormente executa todos os comandos (statements) do código se considerar:

- Ordenação pelo método *bubble sort*
- Ordenação pelo método *bucket sort*

Dica1: encontrar funções do método e executar os casos de testes, observando os comandos executados.

Dica2: para apoio à execução dos testes use uma ferramenta de automação de testes, por exemplo, junit

- Enviar relatório do resultado pelo edisciplinas até a próxima aula (qua 10/4)