# Introdução ao Teste de Software
## *Parte 2*

Prof. André Takeshi Endo

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS CORNÉLIO PROCÓPIO

# Desafios

- Tempo
- Muitas combinações de entrada
- Em um caso de teste, como determinar as saídas esperadas?
- Requisitos
- Não há treinamento
- Não há ferramentas
- Entender a necessidade do teste (não ser opcional)

# Desafios

- O que, formalmente, garantiria a ausência de defeitos?
  - **Teste exaustivo:** elaborar um conjunto de casos de teste que tratem de todas as combinações possíveis de entradas.
  - Dois grandes problemas, quais?

# Desafios

- Exemplo

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

# Desafios

- Exemplo

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

- **Problema 1**
- Entrada: um inteiro (2 bytes)
- [-32.768, +32.767] → 65.536 valores possíveis
- ***É factível gerar todos esses casos de teste para testar este método?***

# Desafios

- Exemplo

```
int blech(int j) {
    j = j - 1;   // deveria ser j = j + 1
    j = j / 3000
    return j;
}
```

- **Problema 2**
- Assumindo que seja possível executar as 65.536 entradas.
- ***E as saídas esperadas (o oráculo)?***

# Desafios

- Exemplo

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

- **Exercício:**
- Existem apenas 4 CTs capazes de revelar o defeito. Quais seriam?

# Desafios

- Exemplo

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

- **Exercício:**
- Existem apenas 4 CTs capazes de revelar o defeito. Quais seriam?
  - (-30000; 0) → saída real?
  - (-29999; 0) → saída real?
  - (30000; 1) → saída real?
  - (29999; 1) → saída real?

# Desafios

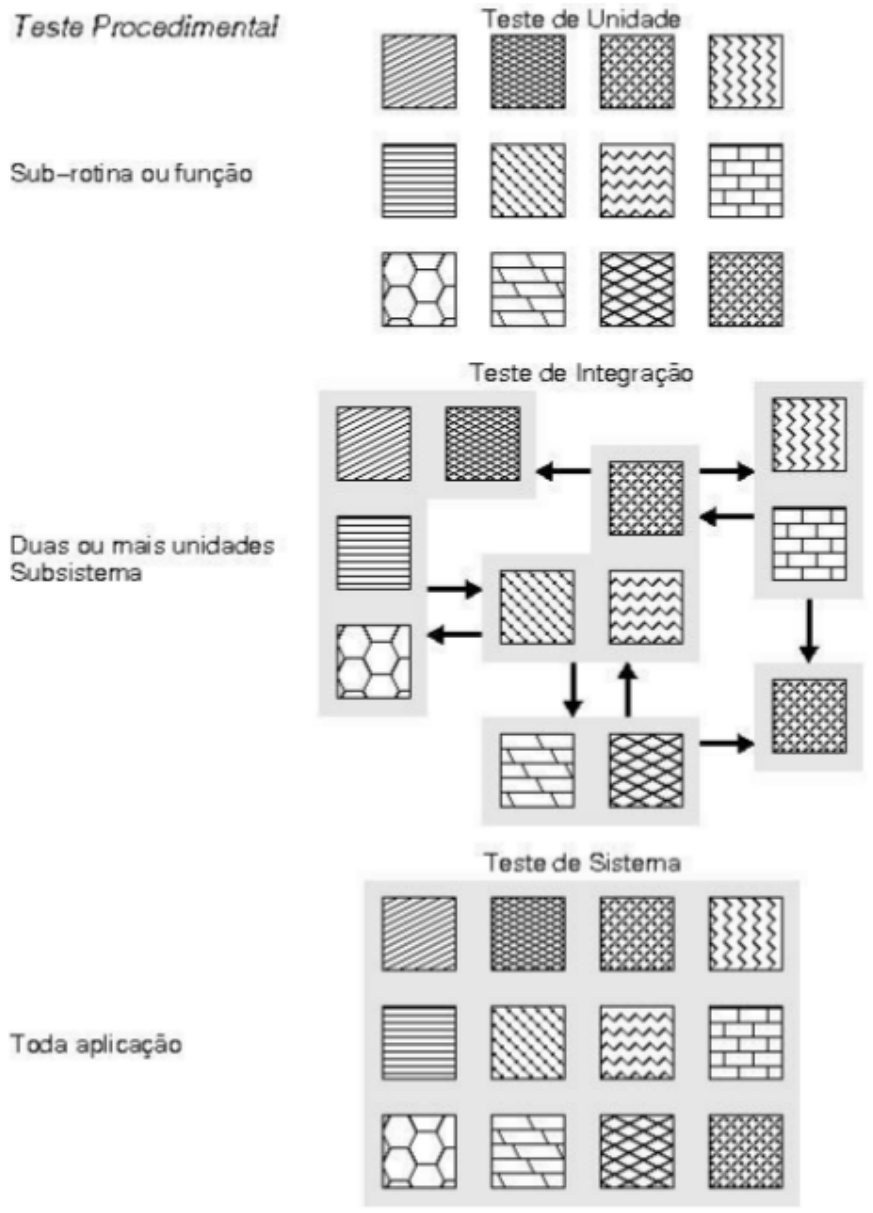
- Limitação principal
- Não existe um software **capaz de provar** a corretude de um programa.

Assim, Dijkstra (1970) disse tudo:

***"Teste pode ser usado para mostrar a presença de defeitos, mas nunca para mostrar sua ausência."***

# Níveis/Fases de Teste

- **Unidade**
- **Integração**
- **Sistema**



# Etapas de Teste

- A atividade de teste pode ser resumida basicamente em quatro etapas:
- (1) Planejamento
- (2) Projeto de casos de teste
- (3) Execução dos teste
- (4) Análise dos resultados

# Bibliografia


- [Pfleeger07] S. L. Pfleeger, “Engenharia de Software: Teoria e Prática”, 2007.
- [Pressman11] R. S. Pressman, “Engenharia de Software: uma abordagem profissional”, 2011.
- [Sommerville03] I. Sommerville, “Engenharia de Software”, 2003.
- [IEEE90] “IEEE Standard Glossary of Software Engineering Terminology”, 1990. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=159342
- [DMJ07] DELAMARO, Márcio Eduardo; MALDONADO, José Carlos; JINO, Mario. Introdução ao teste de software. Rio de Janeiro, RJ: Elsevier, 2007. 394 p. ISBN 9788535226348.
- [Pezze08] PEZZÈ, Mauro; YOUNG, Michal. Teste e análise de software: processo, princípios e técnicas. Porto Alegre, RS: Bookman, 2008. 512 p. ISBN 9780471455936.
- [Myers12] MYERS, Glenford J.; BADGETT, Tom; SANDLER, Corey. The art of software testing. 3rd ed. Hoboken, NJ.: John Wiley & Sons, c2012. xi, 240 p. ISBN 978118031964.

# Bibliografia


- [Rios11] RIOS, Emerson. Gerenciando projetos de teste de software. Niterói, RJ: Imagem art studio, 2011. 312 p. ISBN 9788599479223.
- [Kaner02] KANER, Cem; BACH, James; PETTICHORD, Bret. Lessons learned in software testing: a context-driven approach. New York, N.Y.: Wiley, c2002. xxvii, 286 p. ISBN 0471081124.
- [UUU] Materiais didáticos elaborados pelos grupos de engenharia de software do ICMC-USP, DC-UFSCAR e UTFPR-CP.
- Partes dessa apresentação foram adaptadas do material da profa. Ellen Francine e profa. Simone Souza.