

# Introdução ao Teste de Software

## Conceitos Básicos

Vinicius H. S. Durelli

✉ [durelli@ufsj.edu.br](mailto:durelli@ufsj.edu.br)



# Organização

- 1 Conceitos Básicos
  - Motivação
  - Terminologia: defeitos, erros e falhas
- 2 O que é teste de software?
  - O modelo “RIPR”
- 3 Objetivos do teste de software
- 4 Fases do teste de software
- 5 Considerações finais

## 1 Conceitos Básicos

- Motivação
- Terminologia: defeitos, erros e falhas

## 2 O que é teste de software?

- O modelo “RIPR”

## 3 Objetivos do teste de software

## 4 Fases do teste de software

## 5 Considerações finais

# Software é **essencial!**

- Software está **presente diariamente em nossas vidas.**
  - Software embarcado, e.g., aviões, controle de tráfego aéreo.  
(verifiquem os seus bolsos!)
  - O mercado de software vem crescendo cada vez mais!
  - Processos ágeis.
- ➡ Software precisa ser de qualidade, confiável. Caso contrário:

# Software é **essencial!**

- Software está **presente diariamente em nossas vidas.**
  - Software embarcado, e.g., aviões, controle de tráfego aéreo.  
(verifiquem os seus bolsos!)
  - O mercado de software vem crescendo cada vez mais!
  - Processos ágeis.
- ▢ ➡ Software precisa ser de qualidade, confiável. Caso contrário:
- **Sonda Mars Polar Lander:** falha causada por uma confusão no sistema de medidas americano e europeu.

# Software é essencial!

- Software está **presente diariamente em nossas vidas**.
  - Software embarcado, e.g., aviões, controle de tráfego aéreo.  
(verifiquem os seus bolsos!)
  - O mercado de software vem crescendo cada vez mais!
  - Processos ágeis.
- ➡ Software precisa ser de qualidade, confiável. Caso contrário:
- **Sonda Mars Polar Lander:** falha causada por uma confusão no sistema de medidas americano e europeu.
  - **THERAC-25 Radiation Machine:** teste inadequado causou a morte de 4 pessoas.

# Software é essencial!

- Software está **presente diariamente em nossas vidas**.
- Software embarcado, e.g., aviões, controle de tráfego aéreo.  
(verifiquem os seus bolsos!)
- O mercado de software vem crescendo cada vez mais!
- Processos ágeis.

➡ Software precisa ser de qualidade, confiável. Caso contrário:

- **Sonda Mars Polar Lander:** falha causada por uma confusão no sistema de medidas americano e europeu.
- **THERAC-25 Radiation Machine:** teste inadequado causou a morte de 4 pessoas.
- **Ariane 5:** problema com tratamento de exceções (problemas aritméticos).

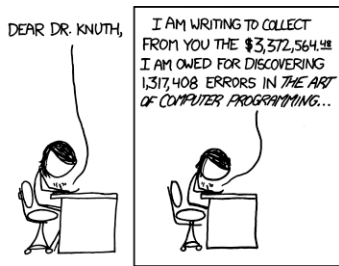
Ainda não está convencido da importância do teste de software?

Veja: <http://listverse.com/2012/12/24/10-seriously-epic-computer-software-bugs/>

## Ah, o “famoso” *bug*

- *Bug* é usado *informalmente*;
- Muitos cientistas da computação usam o termo *bug* para descrever **falhas**, **erros** e **defeitos** (a maioria deles não sabe a diferença).

*“Beware of bugs in the above code;  
I have only proved it correct, not  
tried it.” – Donald Knuth*





# Um pouco de terminologia (1)...

## Definição → Defeito (do inglês, *fault*)

Como sendo um passo, processo ou definição de dados incorretos, portanto, um sendo um conceito estático (Delamaro et al. 2016). Um problema estático no software (Ammann & Offutt 2008; Ammann & Offutt 2016).

\* O que causa defeitos?

Definição → Engano (do inglês, *mistake*)

Ação humana que produz um defeito.

# Um pouco de terminologia (1)...

## Definição → Defeito (do inglês, *fault*)

Como sendo um passo, processo ou definição de dados incorretos, portanto, um sendo um conceito estático (Delamaro et al. 2016). Um problema estático no software (Ammann & Offutt 2008; Ammann & Offutt 2016).

\* O que causa defeitos?

## Definição → Engano (do inglês, *mistake*)

Ação humana que produz um defeito.

## Um pouco de terminologia (2)...

O **estado de um programa** (i.e., execução do programa) em um determinado instante é dado por:

- Valor da memória (variáveis do programa); e
- Apontador de instruções.

### Definição → Erro (do inglês, *Error*)

Estado incorreto ou inconsistente causado pela existência de um defeito (Delamaro et al. 2016).

### Definição → Falha (do inglês, *Failure*)

Quando resultado da execução é diferente do esperado: uma inconsistência pode levar a uma falha (Delamaro et al. 2016). De acordo com Ammann e Offutt, um comportamento (externo) diferente do esperado.

# Entendendo os conceitos por meio de uma analogia...

Uma analogia envolvendo teste de software e medicina

- A lista de **sintomas**.
- A causa da **doença**.
- **Condições internas anômalas**.
  - e.g., hipertensão.



# Entendendo os conceitos por meio de uma analogia...

Uma analogia envolvendo teste de software e medicina

- A lista de **sintomas**. ➡ Falhas
- A causa da **doença**. ➡ Defeito
- **Condições internas anômalas**. ➡ Erros



## Um exemplo mais técnico

```
public static int numberOfZeroes(int [] x) {  
    // Effects: if x is null throw NullPointerException  
    // else return the number of occurrences of 0 in x  
    int count = 0;  
    for (int i = 1; i < x.length; i++) {  
        if (x[i] == 0)  
            count++;  
    }  
    return count;  
}
```

Caso de Teste:

Erro:

Falha:

Saída Esperada:

Saída:

## Um exemplo mais técnico

```
public static int numberOfZeroes(int[] x) {  
    // Effects: if x is null throw NullPointerException  
    // else return the number of occurrences of 0 in x  
    int count = 0;  
    for (int i = 1; i < x.length; i++) {  
        if (x[i] == 0)  
            count++;  
    }  
    return count;  
}
```

Caso de Teste:

Erro:

Falha:

Saída Esperada:

Saída:

## Um exemplo mais técnico

```
public static int numberOfZeroes(int [] x) {  
    // Effects: if x is null throw NullPointerException  
    // else return the number of occurrences of 0 in x  
    int count = 0;  
    for (int i = 1; i < x.length; i++) {  
        if (x[i] == 0)  
            count++;  
    }  
    return count;  
}
```

Caso de Teste: [3, 9, 0]

Erro: ?

Falha: ?

Saída Esperada: ?

Saída: ?



## Um exemplo mais técnico

```
public static int numberOfZeroes(int[] x) {  
    // Effects: if x is null throw NullPointerException  
    // else return the number of occurrences of 0 in x  
    int count = 0;  
    for (int i = 1; i < x.length; i++) {  
        if (x[i] == 0)  
            count++;  
    }  
    return count;  
}
```

Caso de Teste: [3, 9, 0]

Erro: ?

Falha: ?

Saída Esperada: 1

Saída: 1

## Um exemplo mais técnico

```
public static int numberOfZeroes(int[] x) {  
    // Effects: if x is null throw NullPointerException  
    // else return the number of occurrences of 0 in x  
    int count = 0;  
    for (int i = 1; i < x.length; i++) {  
        if (x[i] == 0)  
            count++;  
    }  
    return count;  
}
```

Caso de Teste: [3, 9, 0]

Erro: i é 1 não 0

Falha: nenhuma

Saída Esperada: 1

Saída: 1

## Um exemplo mais técnico

```
public static int numberOfZeroes(int[] x) {  
    // Effects: if x is null throw NullPointerException  
    // else return the number of occurrences of 0 in x  
    int count = 0;  
    for (int i = 1; i < x.length; i++) {  
        if (x[i] == 0)  
            count++;  
    }  
    return count;  
}
```

Caso de Teste: [3, 9, 0]; [0, 7, 1]

Erro: ?

Falha: ?

Saída Esperada: 1

Saída: 0

## Um exemplo mais técnico

```
public static int numberOfZeroes(int[] x) {  
    // Effects: if x is null throw NullPointerException  
    // else return the number of occurrences of 0 in x  
    int count = 0;  
    for (int i = 1; i < x.length; i++) {  
        if (x[i] == 0)  
            count++;  
    }  
    return count;  
}
```

Caso de Teste: [3, 9, 0]; [0, 7, 1]

Erro: i é 1 não 0

Falha: count contém o valor 0 até a instrução return

Saída Esperada: 1

Saída: 0

- 1 Conceitos Básicos
  - Motivação
  - Terminologia: defeitos, erros e falhas
- 2 O que é teste de software?
  - O modelo "RIPR"
- 3 Objetivos do teste de software
- 4 Fases do teste de software
- 5 Considerações finais

# Quiz

**Pergunta ①:** O que é teste de software?

**Pergunta ②:** Com base nas definições anteriores, o que é *debugging* (depuração)?



## Quiz

**Pergunta ①:** O que é teste de software?

### Definição → Teste de Software

Avaliar software por meio de sua execução (Ammann & Offutt 2016).

**Pergunta ②:** Com base nas definições anteriores, o que é *debugging* (depuração)?



## Quiz

**Pergunta ①:** O que é teste de software?

### Definição → Teste de Software

Avaliar software por meio de sua execução (Ammann & Offutt 2016).

**Pergunta ②:** Com base nas definições anteriores, o que é *debugging* (depuração)?

### Definição → Depuração

Dado uma falha, depuração é o processo de localizar o(s) defeito(s) (Ammann & Offutt 2016).



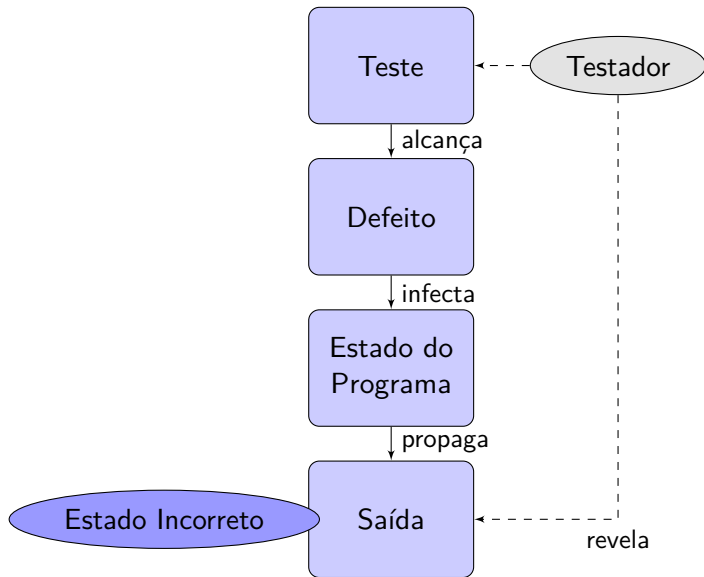


**Nem todas as entradas “disparam” um determinado defeito,** produzindo assim uma falha (saída inconsistente). Esse é o principal tema abordado por teste de software.

Além disso, na maioria das vezes é complicado **relacionar uma falha ao defeito associado.**

- **Reachability:** a localização do defeito deve ser atingida.
- **Infection:** em seguida, o estado do programa deve se tornar incorreto.
- **Propagation:** o estado incorreto deve ser propagado para a saída.
- **Revealability:** deve haver uma intersecção entre a parte incorreta do programa e a parte do programa observada pelo testador.

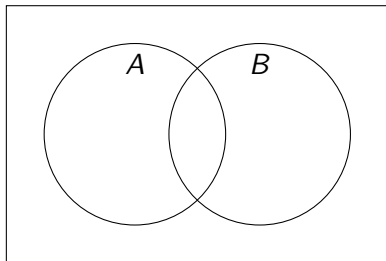
# Modelo RIPR: visão geral



## Detalhando *revealability*...

*Revealability* só acontece quando o testador observa a parte da saída que representa a propagação do estado interno. Por exemplo, se um conjunto  $A$  representa a **porção incorreta do estado interno** e  $B$  contém a **parte da saída observada pelo testador**, é possível denotar *revealability* como:

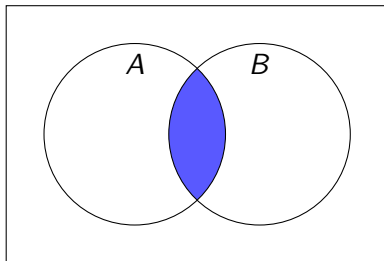
$$A \cap B = \{r \mid r \in A \text{ e } r \in B\}$$



## Detalhando *revealability*...

*Revealability* só acontece quando o testador observa a parte da saída que representa a propagação do estado interno. Por exemplo, se um conjunto  $A$  representa a **porção incorreta do estado interno** e  $B$  contém a **parte da saída observada pelo testador**, é possível denotar *revealability* como:

$$A \cap B = \{r \mid r \in A \text{ e } r \in B\}$$



- 1 Conceitos Básicos
  - Motivação
  - Terminologia: defeitos, erros e falhas
- 2 O que é teste de software?
  - O modelo “RIPR”
- 3 **Objetivos do teste de software**
- 4 Fases do teste de software
- 5 Considerações finais

# Objetivos do teste de software

Com base em um modelo de processo de maturidade

Nível 0

- Não existe diferença entre teste e depuração.

Nível 1

- O propósito da atividade de teste é **mostrar corretude**.

Nível 2

- O propósito da atividade de teste é **mostrar que o software não funciona**.

Nível 3

- O propósito da atividade de teste **não é provar nada específico**, e sim **reduzir o risco de se utilizar o software**.

Nível 4


- Teste é **um estado mental** que ajuda profissionais de TI a **produzir software de qualidade**.

## Mentalidade nível 0...

*“It's a long way to the top  
If you wanna ~~rock 'n' roll~~ software testing”*

*It's A Long Way To The Top (If You Wanna Rock 'n' Roll) by AC/DC*

- Teste de software e depuração são a mesma coisa.
  - Indivíduos nesse nível não sabem a diferença entre **comportamento incorreto** e **problemas em programas**.
  - Não desenvolvem programas que são **confiáveis** (*reliable*).

➡ Estudantes que nunca cursaram essa disciplina encontram-se nesse nível. 

## Mentalidade nível 1...

Em tal nível, desenvolvedores/testadores acreditam que o propósito das atividades de teste é **provar a corretude do software**.

- É impossível provar corretude;
- O que podemos dizer se o software não apresenta problemas?
  - O software é de qualidade ou os casos de teste são horríveis?
- Neste nível, testadores não têm:
  - Uma **meta** adequada;
  - Uma **regra de parada**;
  - Não seguem nenhuma **técnica formal**.




## Mentalidade nível 2

Neste nível, acredita-se que o propósito de atividades de teste é **revelar falhas**.

- Entretanto, a atividade de encontrar falhas é **vista de forma negativa** por outros membros da equipe.
  - Desenvolvedores e testadores vêem uns aos outros como **adversários**.



➡ A maioria dos profissionais encontram-se neste nível. 

## Mentalidade nível 3

Neste nível, acredita-se que teste de software é capaz de **relevar somente a presença de falhas**.

A utilização de software sempre acarreta em riscos.

- Risco baixo, **consequências irrelevantes**;
- Risco elevado, **consequências catastróficas**.



➡ Poucas empresas encontram-se nesse nível.



## Mentalidade nível 4

### Teste de software é...

Visto como uma **disciplina/estado mental** para aprimorar a qualidade do software.

- Teste é **uma das formas** de se aprimorar a qualidade do software.
- Testadores podem se tornar líderes de projeto.
- É responsabilidade dos testadores **avaliar e aprimorar a qualidade** do software.



➡ Teste melhora a habilidade dos desenvolvedores.



## Quiz (2)

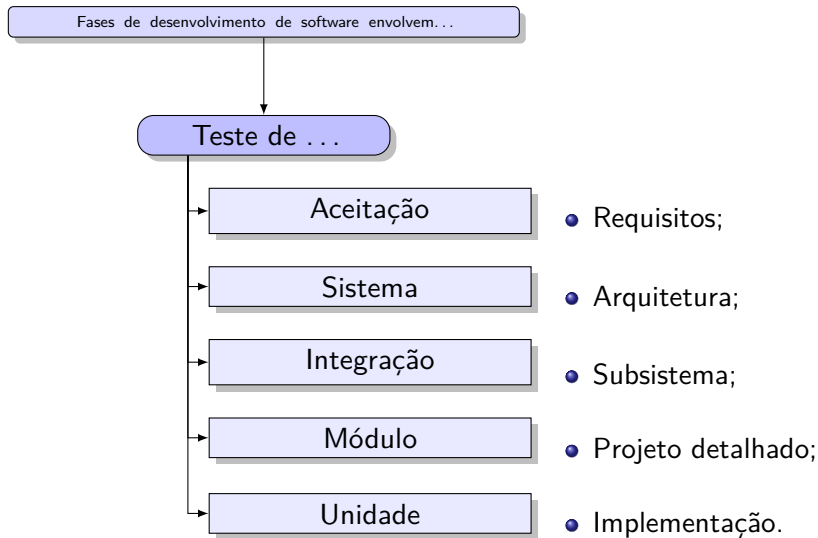
**Pergunta ③:** Em qual nível de mentalidade/maturidade de teste você está (estão)?

**Pergunta ④:** Em qual nível a(s) empresa(s) que você trabalhou ou trabalha atualmente está?



- 1 Conceitos Básicos
  - Motivação
  - Terminologia: defeitos, erros e falhas
- 2 O que é teste de software?
  - O modelo “RIPR”
- 3 Objetivos do teste de software
- 4 Fases do teste de software
- 5 Considerações finais

# Fases do teste de software



- 1 Conceitos Básicos
  - Motivação
  - Terminologia: defeitos, erros e falhas
- 2 O que é teste de software?
  - O modelo “RIPR”
- 3 Objetivos do teste de software
- 4 Fases do teste de software
- 5 Considerações finais**

## Considerações finais. . .

Na aula de hoje nós vimos:

- Conceitos básicos;
  - Definições básicas: teste de software e depuração.
- Propósito do teste de software;
  - Níveis de mentalidade de teste de software;
- Fases do teste de software.

Na **próxima aula**:

- **conceitos básicos (parte 2);**



Ammann, Paul & Jeff Offutt (2008). *Introduction to Software Testing*. 1st ed. Cambridge University Press, p. 346.

– (2016). *Introduction to Software Testing*. 2nd ed. Cambridge University Press, p. 364.

Delamaro, Marcio, Mario Jino, & Jose Carlos Maldonado (2016). *Introdução ao Teste de Software*. 2nd ed. Elsevier, p. 394.

😊 **Agora: exercício(s) sobre o conteúdo da aula de hoje!** 😊

“Devil” icon by mungang kim “Rock Climbing” icon by Paul Phillips, “Buddha” icon by Cass Reese from the Noun Project (<https://thenounproject.com/>).