

# Engenharia de Software

## Verificação e Validação



**Marcello Thiry**  
marcello.thiry@gmail.com

**LQPS**  
<http://www.univali.br/lqps>

## Gerência de qualidade de software

- ☐ **Objetivo:** atingir a satisfação do cliente pela monitoração da qualidade dos produtos e serviços no nível organizacional e dos projetos para garantir que satisfaçam os requisitos do cliente [ISO/IEC 12207]
- ☐ Inclui vários processos:
  - ☐ Garantia da qualidade
  - ☐ **Verificação & Validação**
    - ☐ Revisões técnicas
    - ☐ Auditoria

## Verificação e Validação (V&V)

- ☐ Dois objetivos principais:
  - ☐ Descobrir defeitos no software
  - ☐ Avaliar se o sistema de software é útil e usável numa situação operacional
- ☐ Deve garantir que o software é adequado para o seu propósito
  - ☐ Não significa, necessariamente, totalmente livre de defeitos
  - ☐ O software precisa ter uma qualidade suficiente para seu uso
  - ☐ O tipo de uso determinará o grau de confiança requerido

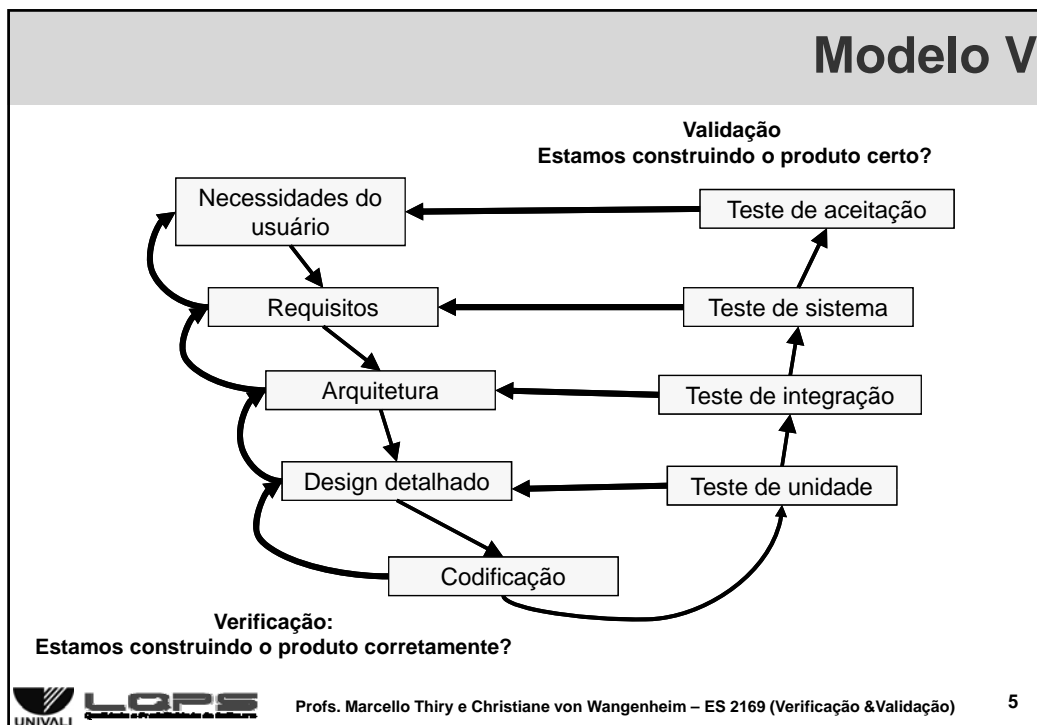
## Verificação x Validação

### **Verificação: Estamos construindo o produto corretamente?**

- ☐ O sistema de software deve estar em conformidade com a sua especificação

### **Validação: Estamos construindo o produto certo?**

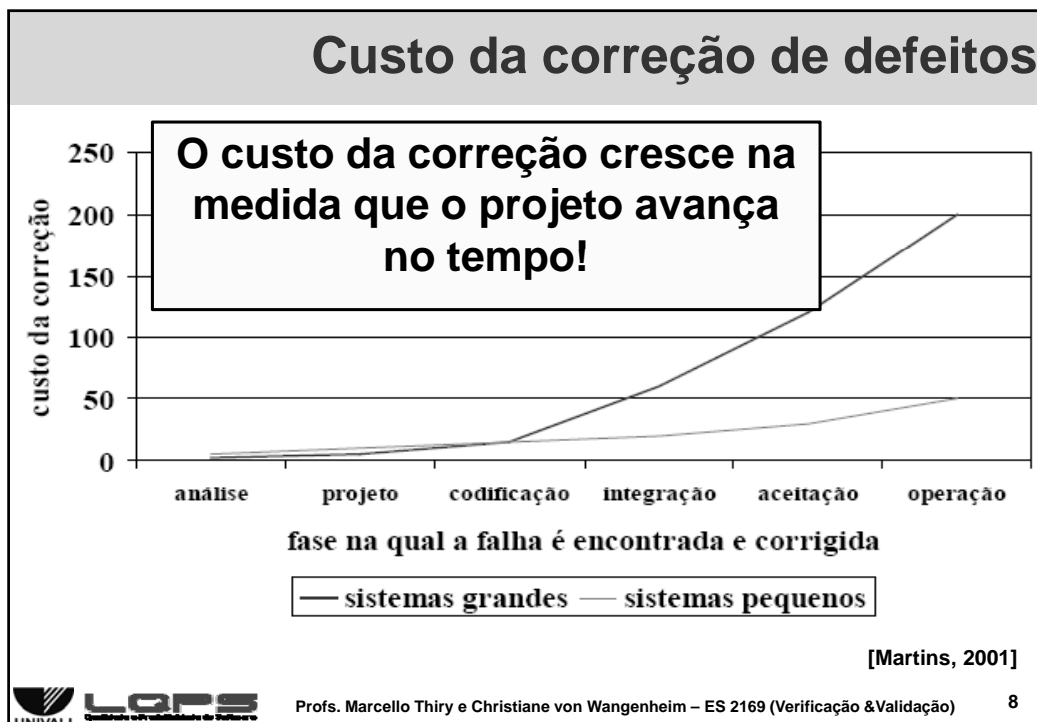
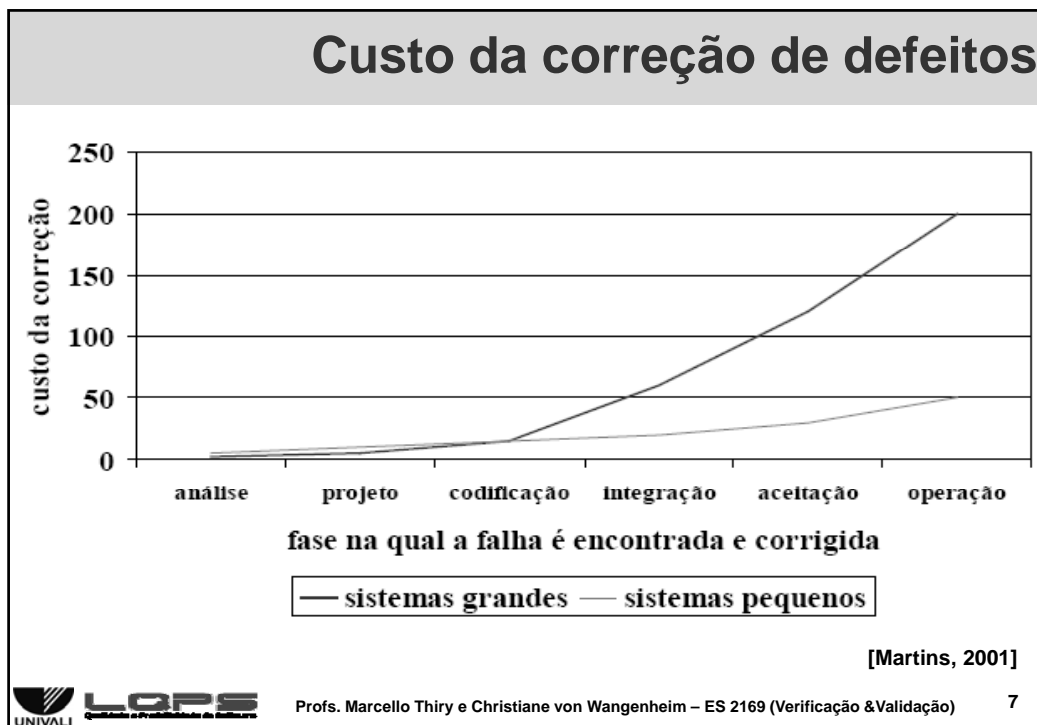
- ☐ O sistema de software deve fazer o que o usuário realmente precisa
- ☐ O foco está em entregar um produto de software com qualidade



### Custo da correção de defeitos

- ☐ O que acontece quando um defeito é encontrado no início do projeto?
- ☐ O que acontece quando um defeito é encontrado somente no final do projeto?
- ☐ Um defeito pode causar outros defeitos?
- ☐ Qual a relação do custo da correção de defeitos com a evolução do projeto?

UNIVALI LQPS Prof. Marcello Thiry e Christiane von Wangenheim – ES 2169 (Verificação & Validação) 6



## Vantagens da V&V

- ☐ Permite encontrar defeitos mais cedo
- ☐ Melhora a qualidade dos produtos
- ☐ Torna os requisitos mais estáveis
- ☐ Permite avaliação contínua da qualidade e da produtividade
  - ☐ Facilita o gerenciamento

## Vantagens da V&V

- ☐ MAS, como a qualidade é atingida?
- ☐ Pelas atividades de garantia de qualidade **durante todo o processo de desenvolvimento** e avaliado pela V&V
- ☐ *“Você não pode criar qualidade pela atividade de teste”*

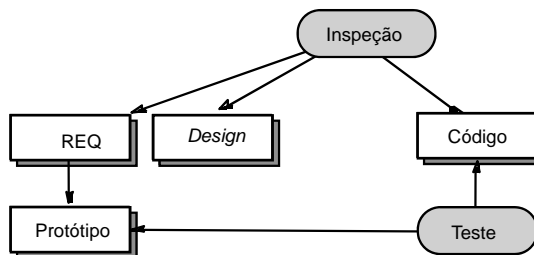
## Técnicas de V&V

### ❑ V&V estática:

- ❑ Análise de representação estática do produto para descobrir defeitos
- ❑ Não envolve a execução do produto
- ❑ Exemplos: Revisões e Inspeções

### ❑ V&V dinâmica:

- ❑ Exercitando e observando o comportamento do produto
- ❑ Envolve a execução do produto (código ou módulo executável)
- ❑ Exemplo: Testes e Simulação



## Defeito, Erro, Falha

- ❑ **Defeito (*Fault*):** um passo, processo ou definição de dados incorreta em um software que faz com que ele execute numa forma não pretendida ou prevista (*por exemplo, uma parte do código faltando ou incorreta*)
- ❑ **Erro (*Error*):** diferença entre o valor obtido e o valor esperado, ou seja, qualquer estado intermediário incorreto ou resultado inesperado na execução do software (*por exemplo, o valor de uma operação deveria ser 1, mas resultou em 0*)
- ❑ **Falha (*Failure*):** incapacidade de um software realizar suas funções requeridas de acordo com os requisitos especificados (*por exemplo, uma exceção em uma tela de cadastro*)

## Defeito, Erro, Falha

- ☐ Uma **falha** é a incapacidade demonstrada de um software realizar sua função requerida, ou seja, o mal funcionamento de um sistema evidenciado pela saída incorreta, terminação anormal ou não atendimento às restrições de tempo e espaço
- ☐ A causa de uma **falha** é um **defeito**
- ☐ Um **defeito** pode permanecer não detectado por muito tempo, até que algum evento o ative.
- ☐ Quando este evento ocorre, ele traz o software para um estado intermediário e instável, chamado **erro** (o qual, se e quando se propaga para a saída, eventualmente causa uma **falha**)
- ☐ **Defeito → Erro → Falha**



Profs. Marcello Thiry e Christiane von Wangenheim – ES 2169 (Verificação & Validação)

13

## Exercício

- ☐ Requisito:
  - ☐ Para um valor dado de X, imprimir o valor de  $(2 * X + 2)$
- ☐ Implementação:                      Execução

<code>Readln(X);</code>	<code>X = 3</code>
<code>Saida := 2 * X - 2;</code>	<code>Saida = 2 * 3 - 2 = 4</code>
<code>Writeln(X);</code>	<code>X = 3</code>
<code>Writeln(Saida)</code>	<code>Saida = 4</code>
- ☐ Saída:
  - ☐ X = 3
  - ☐ Saida = 4



Profs. Marcello Thiry e Christiane von Wangenheim – ES 2169 (Verificação & Validação)

14

## Exercício

### ❑ Requisito:

- ❑ Para um valor dado de X, imprimir o valor de  $(2 * X + 2)$

### ❑ Implementação:

```
Readln(X);
```

```
Saida := 2 * X
```

```
Writeln(X);
```

```
Writeln(Saida)
```

Defeito

- 2;

### Execução

```
X = 3
```

```
Saida =
```

```
2 * 3 - 2 = 4
```

Erro

```
X = 3
```

```
Saida =
```

```
4
```

Falha

### ❑ Saída:

- ❑ X = 3

- ❑ Saida = 4

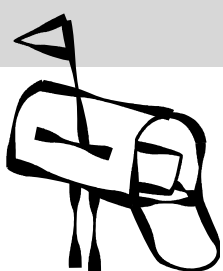

Falha

## Na prática ...

- ❑ Programadores experientes, na média, fazem um erro a cada 10 linhas de código
- ❑ Aproximadamente 50% destes defeitos são corrigidos durante a compilação do código
- ❑ Outros defeitos são detectados durante os testes
- ❑ Aproximadamente 15% dos defeitos ainda estão no sistema quando entregue ao cliente

[Watts Humphrey, 2000]



<b>Contato</b>	
	<p><b>Marcello Thiry</b> marcello.thiry@gmail.com</p> <p><b>LQPS</b> <a href="http://www.univali.br/lqps">http://www.univali.br/lqps</a></p>
<div style="display: flex; justify-content: space-between; align-items: center;"><div> <b>LQPS</b> <small>Laboratório de Qualidade e Produtividade de Software</small></div><div>Profs. Marcello Thiry e Christiane von Wangenheim – ES 2169 (Verificação &amp; Validação)</div><div>17</div></div>	