

Teste de Software

Prof. Eiji Adachi

Objetivos

- O que é Teste de Software?
- Por que realizar Teste de Software?
- Quando realizar Teste de Software?
- O que testar?
- Terminologia básica

O que é Teste de Software?

O que é
**Teste de
Software?**

“Teste de Software consiste na averiguação de que o comportamento de um programa durante sua execução atende o que foi especificado.”

[Definição simples pelo professor]

Teste de Software

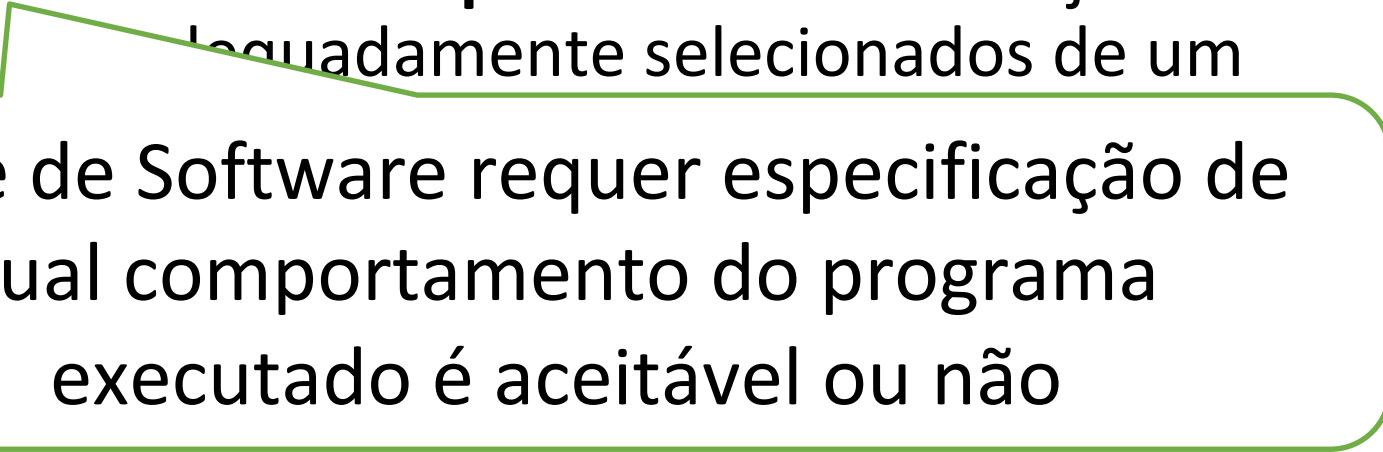
- Teste de Software consiste em verificar dinamicamente que um programa provê comportamentos esperados em um conjunto finito de casos de teste que são adequadamente selecionados de um domínio de execução tipicamente infinito [SWEBOK]

Teste de Software

- Teste de Software consiste em verificar **dinamicamente** que um programa provê comportamentos esperados em um conjunto finito de casos de teste que são derivados de uma especificação de um domínio de execução típico

Teste de Software requer a execução do programa sob análise

Teste de Software

- Teste de Software consiste em verificar dinamicamente que um programa provê **comportamentos esperados** em um conjunto finito de casos de teste que  aleatoriamente selecionados de um domínio de e

Teste de Software requer especificação de qual comportamento do programa executado é aceitável ou não

Teste de Software

- Teste de Software consiste em verificar dinamicamente que um programa provê comportamentos esperados em **um conjunto finito de casos de teste que são adequadamente selecionados** de um domínio de execução tipicamente infinito [SWEBOK]

Teste de Software requer a seleção de um sub-conjunto de casos de teste a ser executado

Teste de Software

- Teste de Software é também um processo realizado ao longo do processo de desenvolvimento do software
 - Planejamento e Controle
 - Análise e Projeto
 - Implementação e Execução
 - Avaliação e Relato
 - Finalização

Resumo: O que é Teste?

- Uma técnica de Verificação & Validação para averiguar dinamicamente o comportamento de um programa
- Um processo dentro do processo de desenvolvimento de software

Por que testar?

Por que
Teste de Software
é importante?

HERRAR

É

UANO

Causas de Erros Humanos

- Falta de comunicação ou comunicação ineficiente
- Documentação incorreta, incompleta ou ambígua
- Complexidade do problema ou da solução
- Inexperiência ou deficiência técnica
- Fadiga ou stress
- Pressão de tempo
- ...

Causas de Erros Humanos

- Omissão:
 - Não fez algo que deveria ter feito
- Comissão:
 - Fez algo que deveria ter sido feito, mas de modo incorreto

Falhas de software afetam
negativamente a qualidade
do software.

Custo de Falhas

*“According to the Consortium for Information and Software Quality, poor software quality cost US companies **\$2.08 trillion** in 2020.”*

Custo de Falhas

In February 2020, more than 100 flights were disrupted at Heathrow Airport because of a software “glitch.”

Falhas Famosas

- *NASA's Mars Climate Orbiter*
 - Nave espacial Climate Orbiter perdeu-se no espaço
 - Motivo: conversão incorreta de unidades do sistema britânico pro sistema métrico
 - Prejuízo: \$ 125 milhões

<http://www.cse.psu.edu/~gxt29/bug/softwarebug.html>

Falhas Famosas

- *Ariane 5 Flight 501*
 - Foguete não tripulado europeu explodiu logo após lançamento
 - Motivo: operação matemática causou um *integer overflow*
 - Prejuízo: \$ 500 milhões

<http://www.cse.psu.edu/~gxt29/bug/softwarebug.html>

Testes ao resgate!

- Estudo mostra que:
 - *“Aproximadamente 92% de falhas catastróficas resultam de erros não tratados adequadamente pelo sistema”*
 - *“Em 58% das falhas catastróficas, as causas poderiam ter sido detectadas com testes simples”*

Yuan, D., Luo, Y., Zhuang, X., Rodrigues, G. R., Zhao, X., Zhang, Y., ... & Stumm, M. (2014, October). Simple Testing Can Prevent Most Critical Failures: An Analysis of Production Failures in Distributed Data-Intensive Systems. In OSDI (pp. 249-265).

No entanto...

- Testes de Software podem consumir de 25% a 50% dos recursos de projetos de software

**Por que
Teste de Software
é difícil?**

Desafios de Testar Software

- Software é imaterial
- Software é complexo

Desafios de Testar Software

- Software evolui rapidamente

Desafios de Testar Software

- Condições possíveis de execução de um programa são infinitas

Resumo: Por que

- Por que testar?
 - Ganhar confiança:
 - Software não possui defeitos
 - Software está apto para uso
 - Software atende especificação

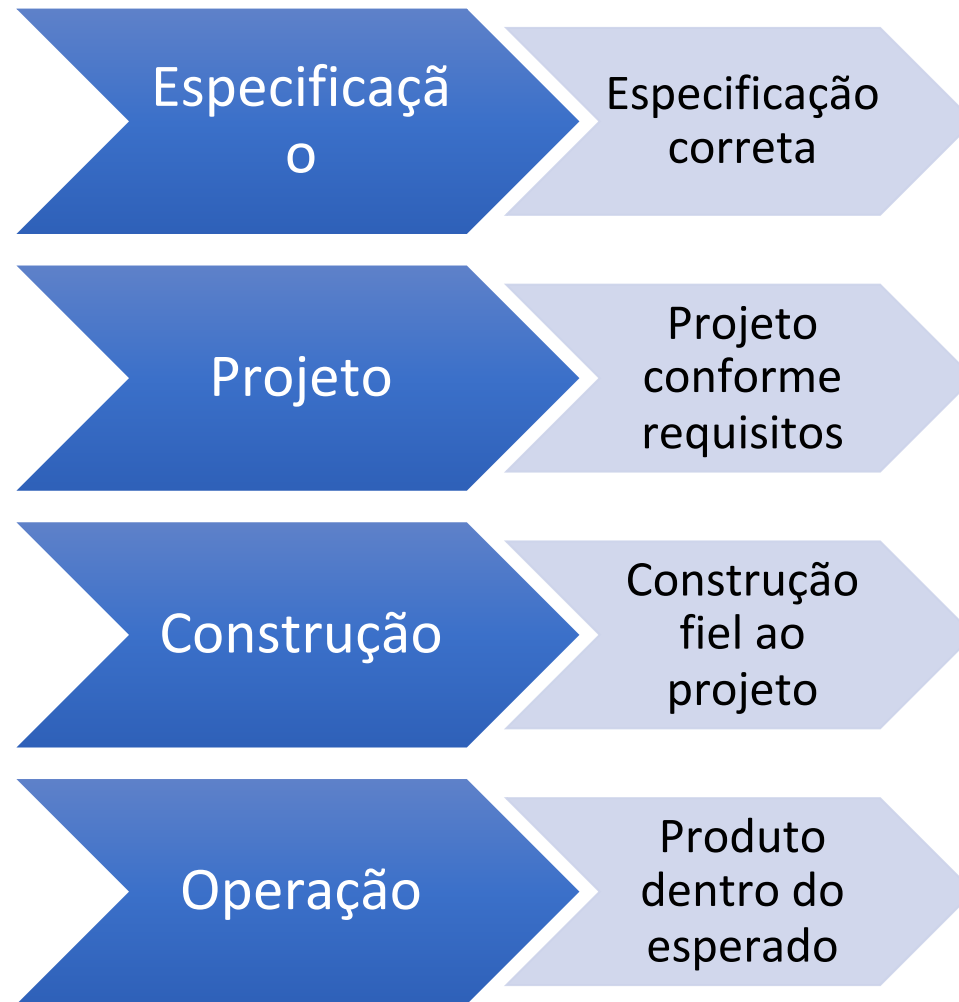
Quando testar?

Ciclo de Desenvolvimento

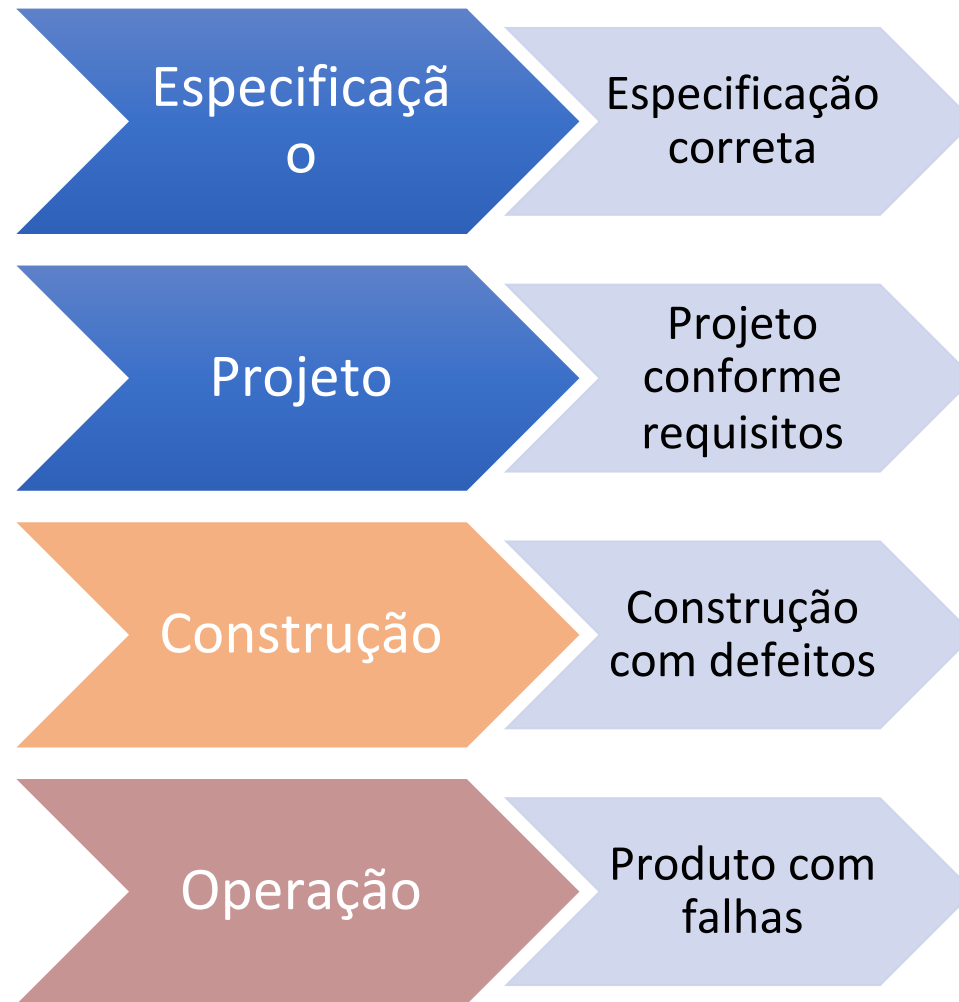
- Especificação
- Projeto
- Construção
- Operação

Em que
momentos
defeitos podem
ser introduzidos?

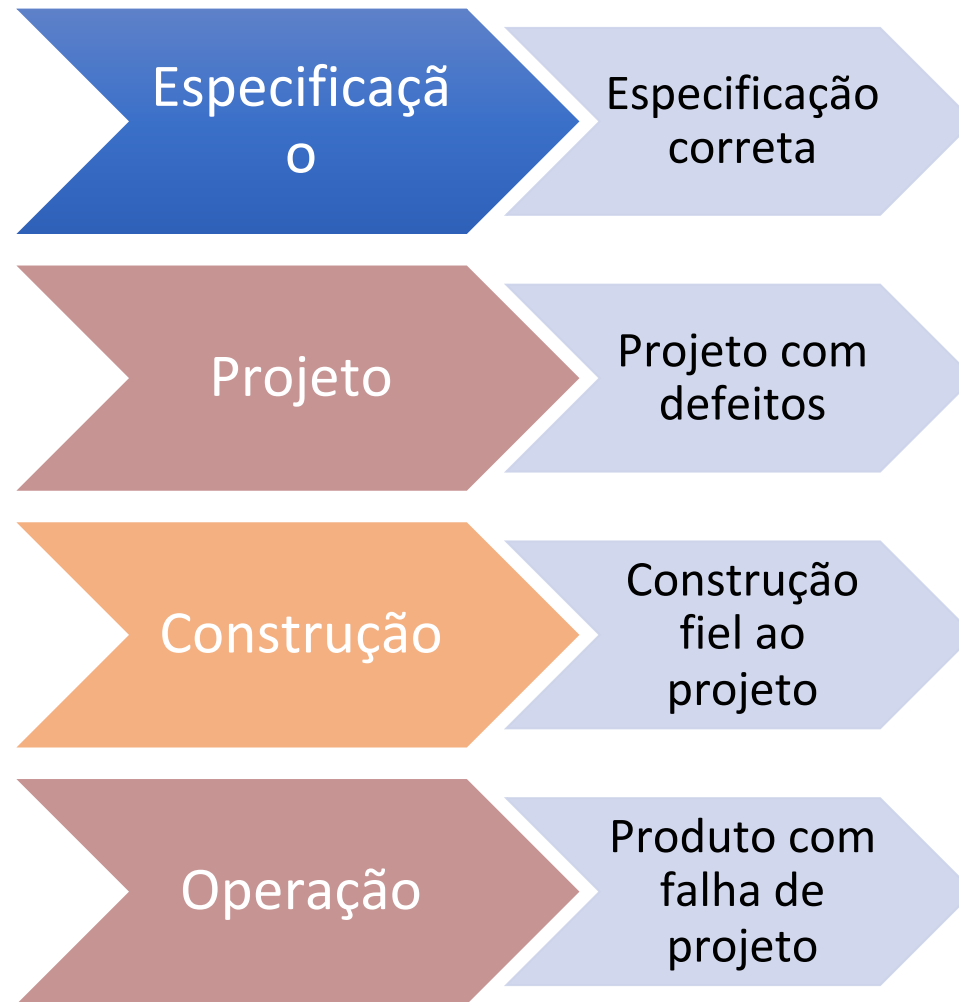
Mundo Ideal



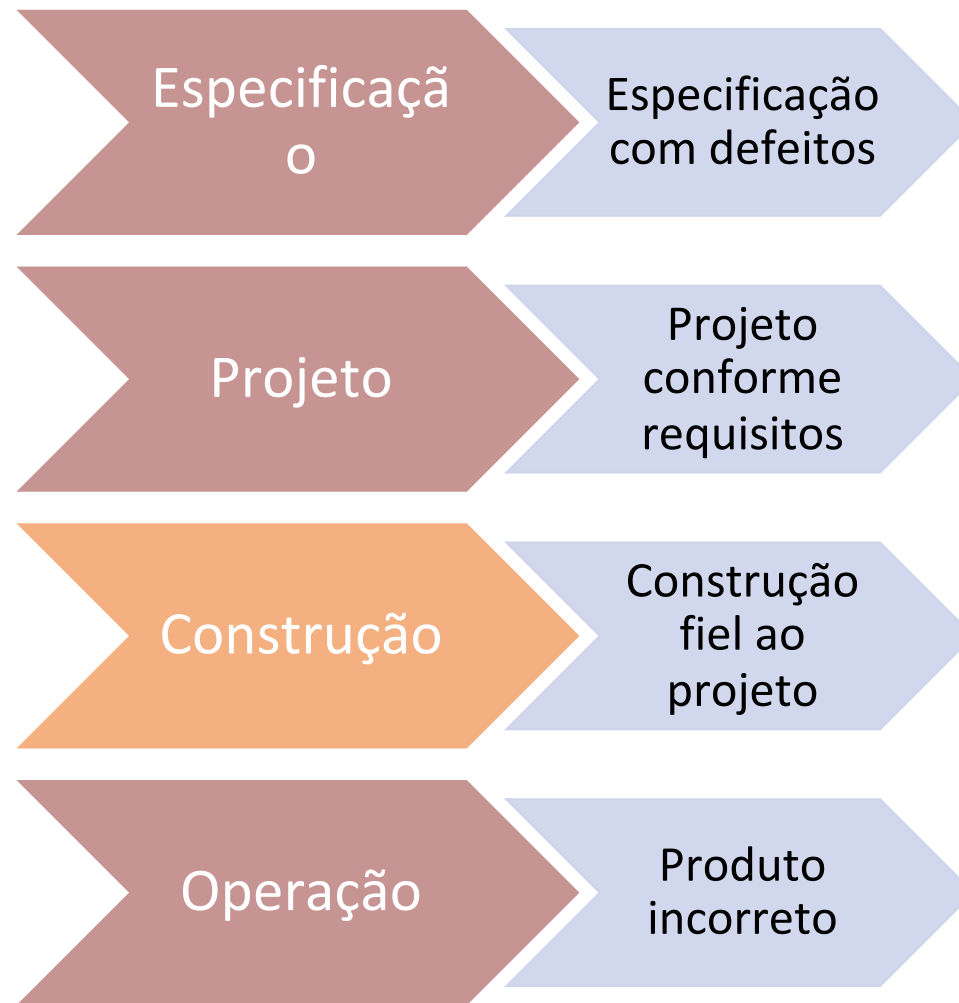
Mundo Real



Mundo Real



Mundo Real



Moral da História

- Defeitos podem ser inseridos no software em qualquer fase do desenvolvimento
- É importante realizar atividades de **controle de qualidade** ao longo de todas as fases do processo de desenvolvimento

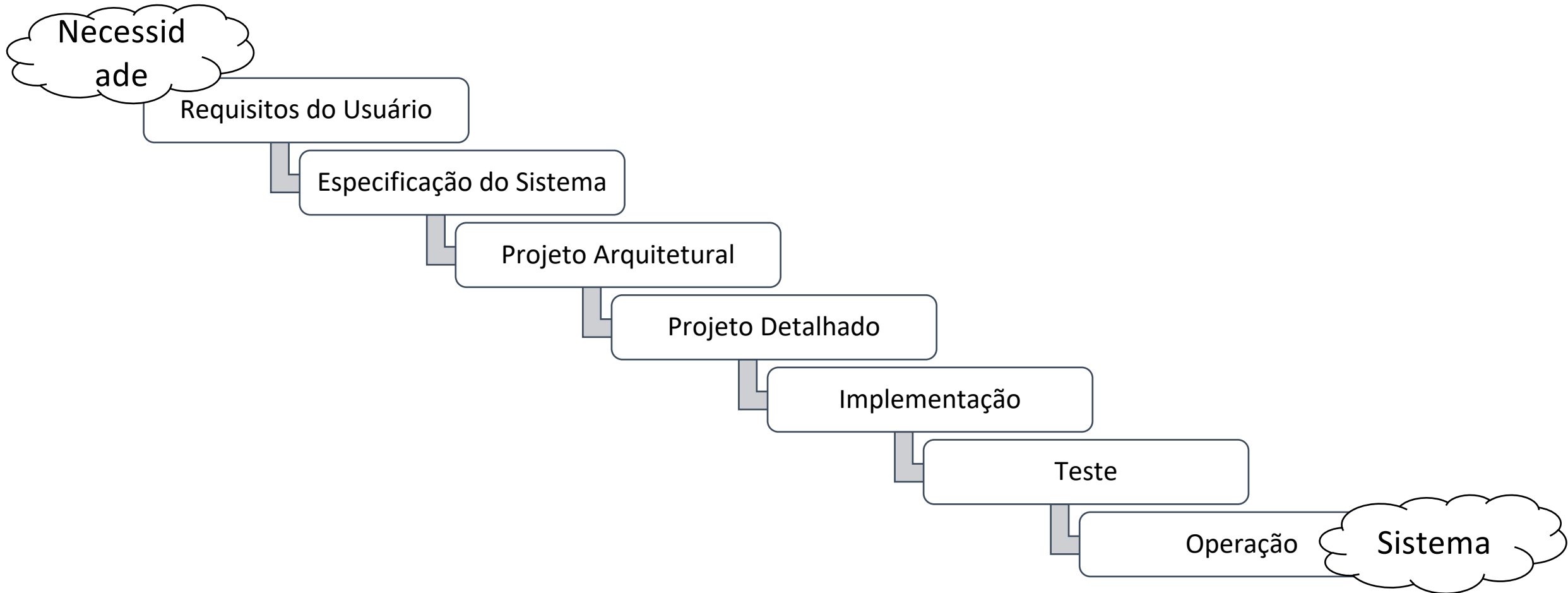
Resumo: Quando Testar

- Quando testar?
 - O teste em si, requer um componente de software executável. Logo, só podemos testar após alguma atividade de implementação ser concluída
 - Atividades de controle devem ser realizadas a cada etapa do desenvolvimento do software

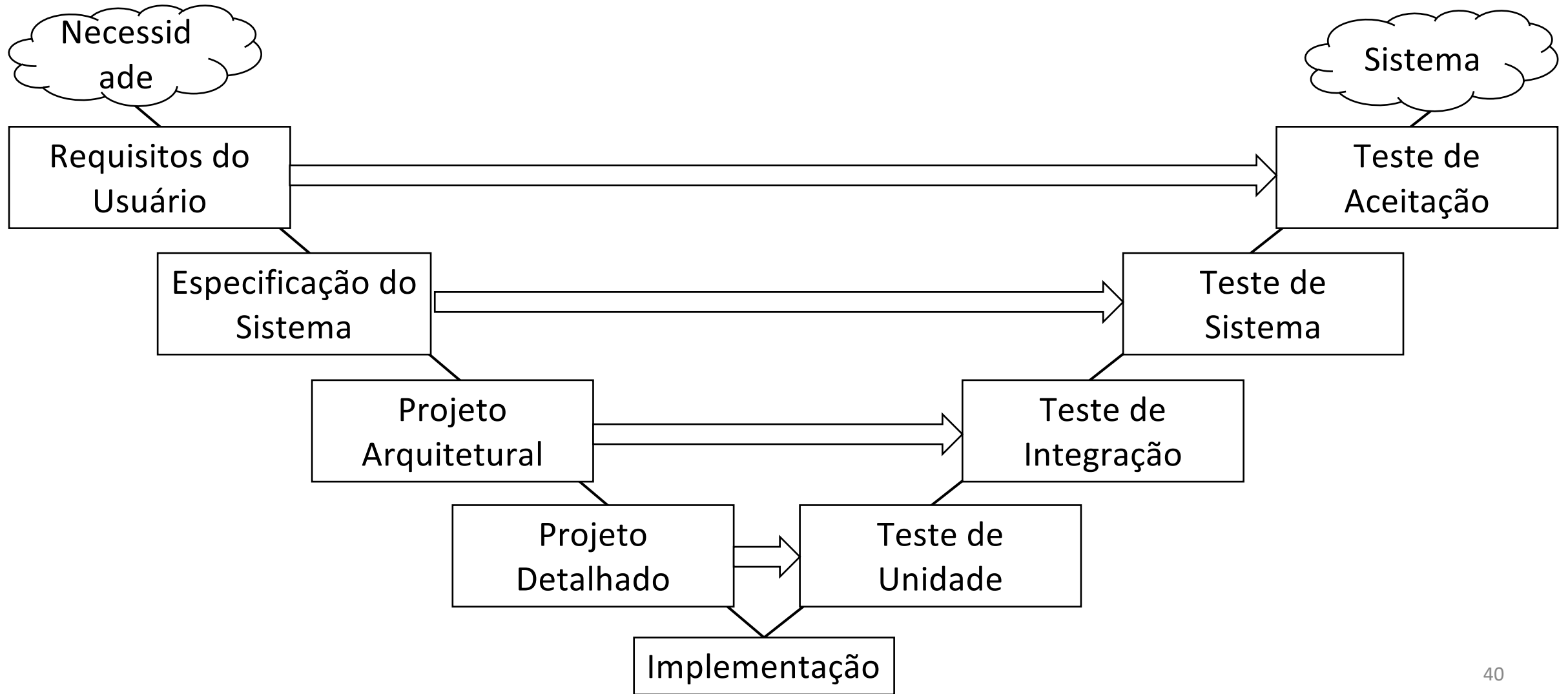
Níveis de Teste

O que testar?

Desenvolvimento “Cascata”



Modelo V



Níveis de Teste

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

Teste de Unidade

- Objetivo é identificar erros de lógica e de programação nas unidades
 - Unidade = Menor unidade de programação testável isoladamente
- Diferentes linguagens de programação possuem unidades diferentes
 - C: Função
 - Java: Métodos

Teste de Integração

- Objetivo é verificar se as unidades testadas individualmente se comunicam corretamente
 - Por que é necessário testar a integração, se isoladamente as unidades funcionam corretamente?

Teste de Sistema

- Objetivo é verificar se o programa em si interage corretamente com o sistema para o qual foi projetado
 - SO, banco de dados, hardware, etc
- Inclui teste de funcionalidade, usabilidade, segurança, confiabilidade, desempenho, stress, etc

Teste de Aceitação

- Objetivo é verificar se o programa atende as exigências do usuário final
 - Usabilidade
 - Usuários aprender a usar o sistema facilmente?
 - Usuários podem usar o sistema sem cometer enganos?
 - Utilidade
 - O usuário consegue realizar no sistema o que ele realmente precisa realizar?
 - Interface
 - O usuário gosta da aparência do sistema?

Teste de Aceitação

- Fases do teste de aceitação:
 - Teste alpha
 - Realizado no ambiente de desenvolvimento por usuários ou membros da organização
 - Os usuários realmente querem as funcionalidades que você implementou?
 - Teste beta
 - Realizado fora do ambiente de desenvolvimento, num ambiente de uso “mundo-real”, mas controlado e monitorado
 - Os usuários conseguem usar o sistema em seu ambiente operacional real?

Atributos de Qualidade

O que testar?

Atributos de Qualidade

- Definem os diferentes aspectos que podem ser levados em consideração para avaliar a qualidade de um produto de software

Atributos de Qualidade

- Funcionalidade
 - Grau com que um determinado produto de software provê funcionalidades satisfaçam o usuário em suas necessidades declaradas e implícitas, dentro de um determinado contexto de uso

Atributos de Qualidade

- Confiabilidade
 - Grau com que um determinado produto de software provê nível de desempenho nas condições estabelecidas

Atributos de Qualidade

- Usabilidade
 - Grau com que um determinado produto de software pode ser compreendido, seu funcionamento aprendido, ser operado e ser atraente ao usuário

Atributos de Qualidade

- Eficiência
 - Grau com que um determinado produto de software necessita de recursos para prover suas funcionalidades com o nível de desempenho esperado

Atributos de Qualidade

- Portabilidade
 - Grau com que um determinado produto de software pode ser transferido de um ambiente para outro

Resumo: O que Testar?

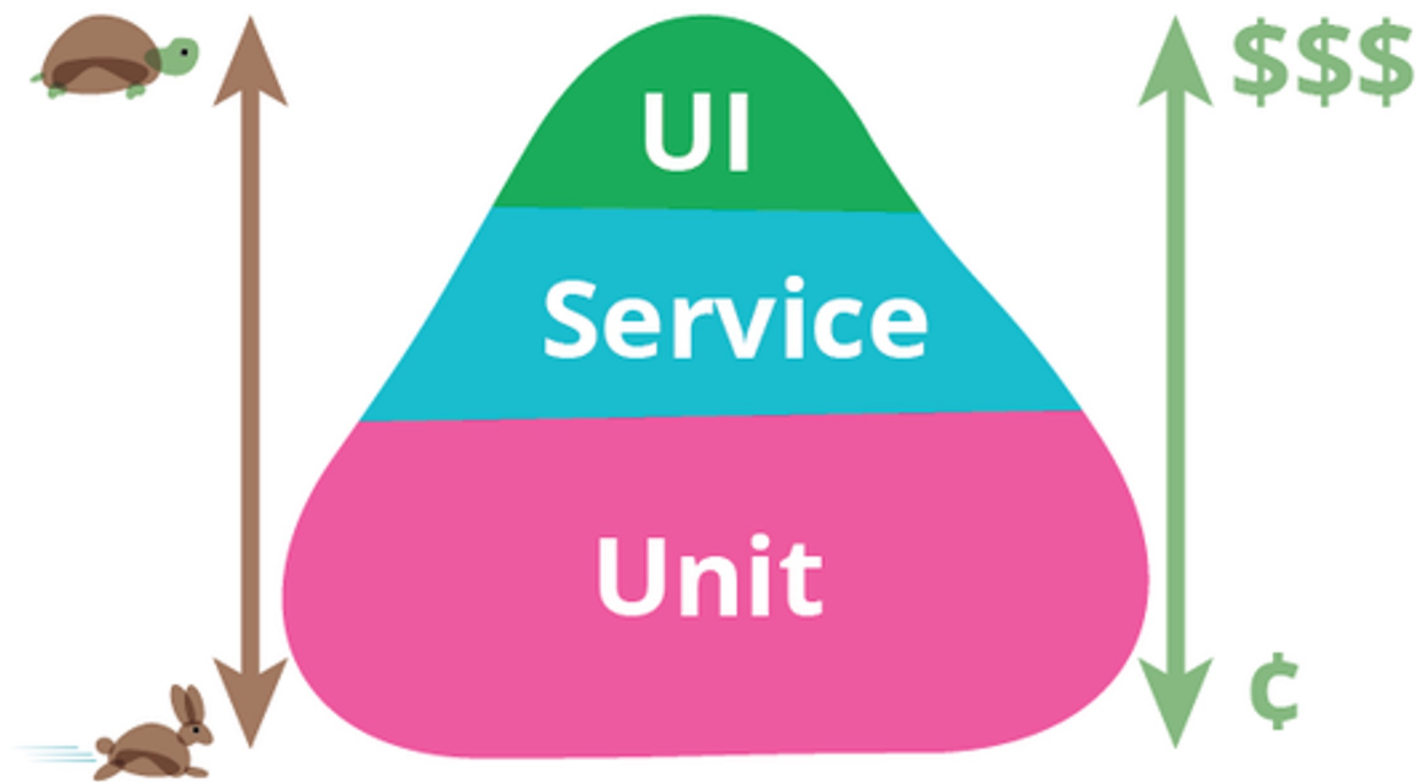
- Nível do Teste
 - Unidade, Integração, Sistema, Aceitação
- Atributos de qualidade
 - Confiabilidade, Usabilidade, Eficiência...

O quanto testar?

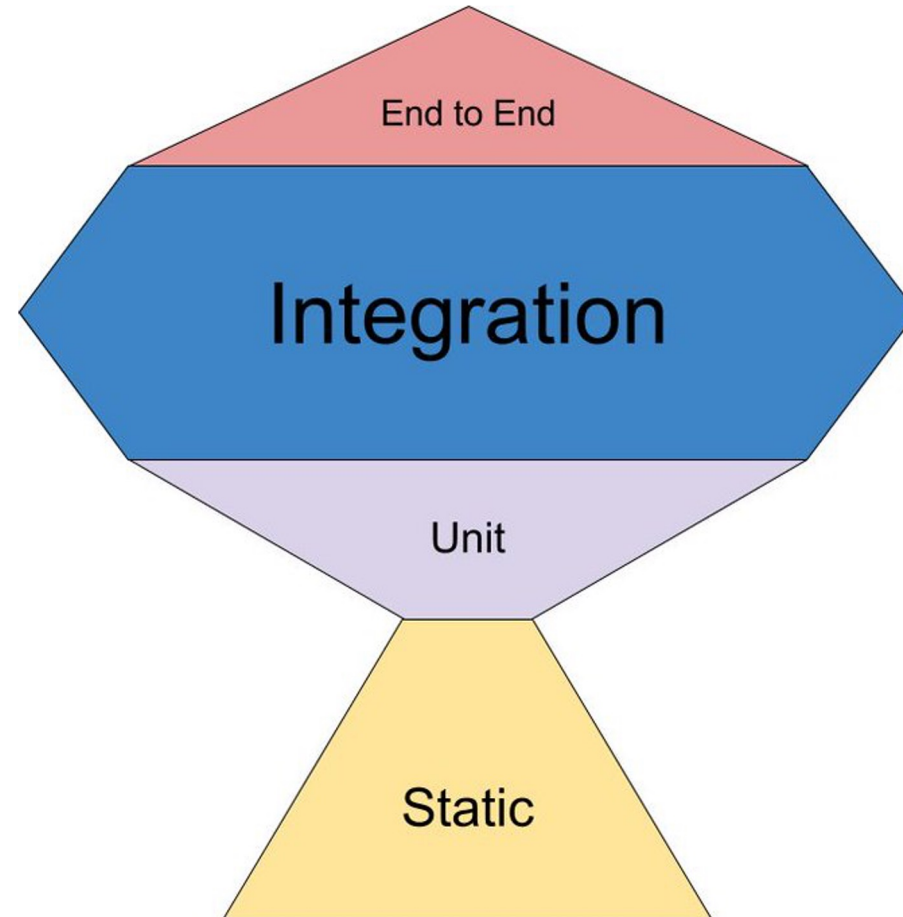
Ou dito de outra forma...

- Quanto de esforço em cada nível de teste?

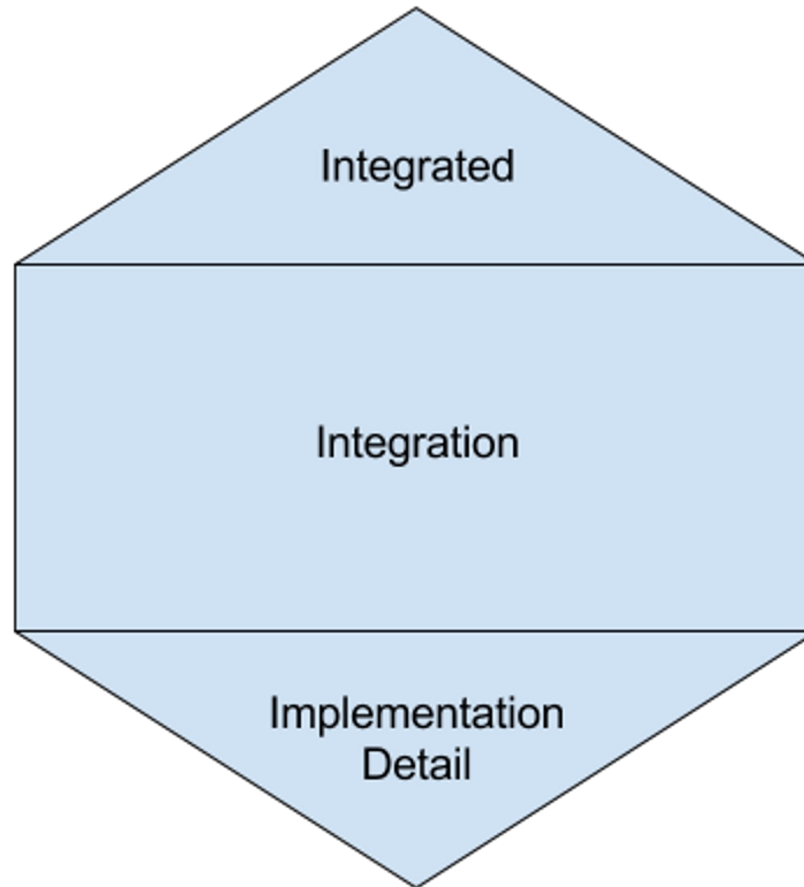
A Pirâmide de Testes



O Troféu de Testes



O Favo de Mel de Testes



E aí...?

- Quanto de esforço depende:
 - Características do sistema
 - Características do time
 - Disponibilidade de recursos
- E ainda existe toda a dimensão dos diversos atributos de qualidade

Terminologia de mal funcionamento

De modo geral, há muitos termos para descrever o mal funcionamento de um software.

Ex.: O que é um *bug*?

Diversidade de Termos

- Diversos termos são usados para descrever o mal funcionamento de um sistema:
 - *Bug, Defect, Error, Failure, Fault, Flaw, Glitch, Mistake, ...*
- Estes termos são muitas vezes usados inconsistentemente entre áreas diferentes, ou mesmo entre autores diferentes de uma mesma área

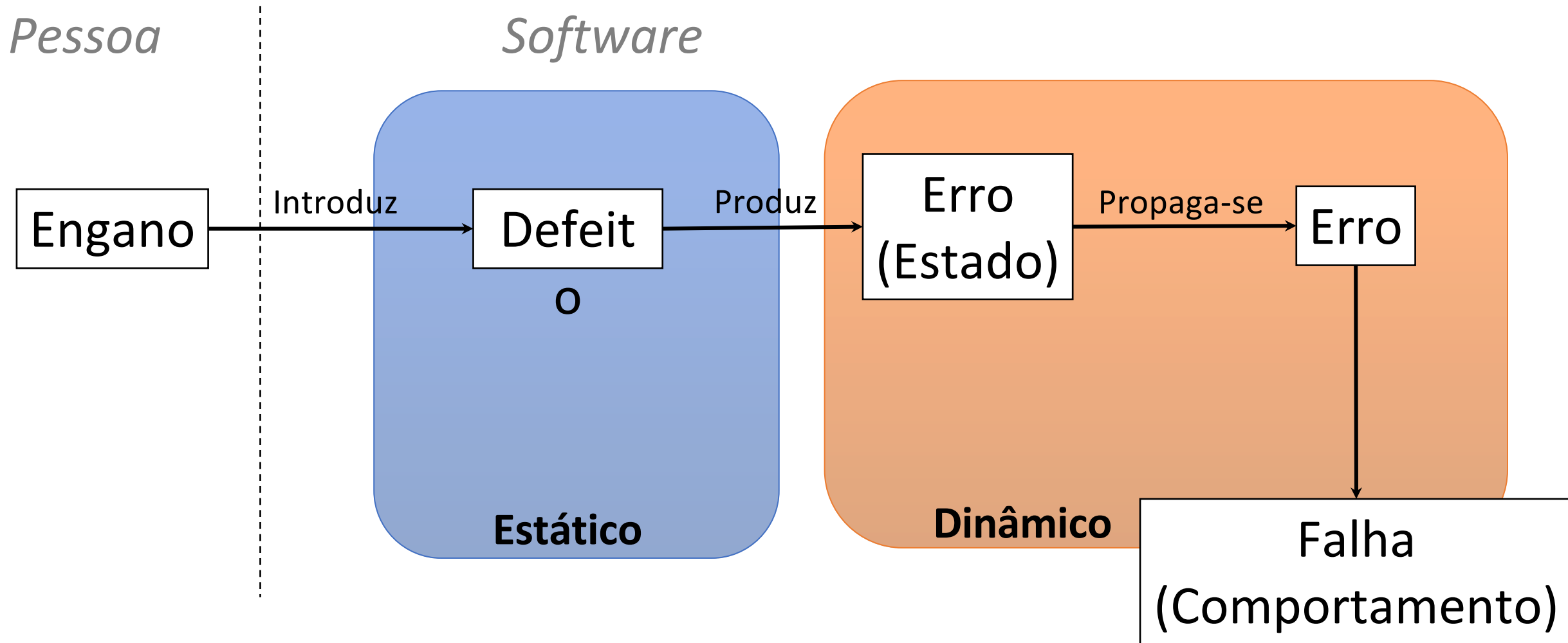
Terminologia Básica

- Engano (*Mistake*)
 - Ação humana que produz um defeito
- Defeito (*Fault* ou *Defect*)
 - Passo, processo ou definição de dado incorreto
- Erro (*Error*)
 - Estado inconsistente ou inesperado do sistema, quando comparado ao estado teoricamente correto
- Falha (*Failure*)
 - Inabilidade do sistema ou componente realizar a sua funcionalidade como esperado

Causa e Efeito

- Sobre o mal funcionamento de um sistema, é importante distinguir:
 - A **causa** do mal funcionamento, e
 - O **efeito** indesejado observado no serviço provido pelo sistema

Relação Entre os Termos



Relação Entre os Termos

- Um Engano pode introduzir um Defeito no código fonte
- Um Defeito no código, quando executado, pode produzir um estado de Erro no sistema
- Um Erro pode ser propagado pelo sistema, ocasionado outros Erros
- Quando um Erro é propagado até a saída do programa e é observada uma diferença no comportamento esperado, então ele ocasionou uma Falha

Exemplo

- Considere que em determinado trecho do código fonte, o correto seria implementar uma atribuição da seguinte forma: $x = a * b$

Exemplo

- Considere que em determinado trecho do código fonte, o correto seria implementar uma atribuição da seguinte forma: $x = a * b$
 - Engano: o programador teve uma interpretação incorreta da solução do problema, o que levou a uma formulação incorreta da equação

Exemplo

- Considere que em determinado trecho do código fonte, o correto seria implementar uma atribuição da seguinte forma: $x = a * b$
 - Engano: o programador teve uma interpretação incorreta da solução do problema, o que levou a uma formulação incorreta da equação
 - Defeito: a atribuição foi implementada como $x = a / b$

Exemplo

- Considere que em determinado trecho do código fonte, o correto seria implementar uma atribuição da seguinte forma: $x = a * b$
 - Engano: o programador teve uma interpretação incorreta da solução do problema, o que levou a uma formulação incorreta da equação
 - Defeito: a atribuição foi implementada como $x = a / b$
 - Se esta atribuição é executada com $b=1$, nenhum erro é produzido, pois $a/1 == a*1$

Exemplo

- Considere que em determinado trecho do código fonte, o correto seria implementar uma atribuição da seguinte forma: $x = a * b$
 - Engano: o programador teve uma interpretação incorreta da solução do problema, o que levou a uma formulação incorreta da equação
 - Defeito: a atribuição foi implementada como $x = a / b$
 - Se esta atribuição é executada com $b=1$, nenhum erro é produzido, pois $a/1 == a*1$
 - Falha: observa-se, em algum lugar do sistema, que o valor da variável x , ou alguma outra variável que foi calculada com base em x , é diferente do esperado

Exemplo

- Considere que em determinado trecho do código fonte, o correto seria implementar uma atribuição da seguinte forma: $x = a * b$
 - Engano: o programador teve uma interpretação incorreta da solução do problema, o que levou a uma formulação incorreta da equação
 - Defeito: a atribuição foi implementada como $x = a / b$
 - Se esta atribuição é executada com $b=1$, nenhum erro é produzido, pois $a/1 == a*1$
 - Erro: se o valor de b for igual a 2, por exemplo, o esperado seria $x = A*2$, mas o calculado seria $x = A/2$, portanto, o sistema estaria num estado de erro
 - Falha: observa-se, em algum lugar do sistema, que o valor da variável x , ou alguma outra variável que foi calculada com base em x , é diferente do esperado
 - Se a atribuição $x = a/b$ for executada com $b==0$, a falha seria imediata

Defeitos “dormentes”

- Nem todos os defeitos causam falhas: alguns defeitos podem ficar “dormentes” no código sem nunca ocasionar falhas
 - Nunca são executados
 - As condições que ocasionariam falhas nunca ocorrem

Outras Causas de Falhas

- Condições ambientais podem afetar um hardware, que, por sua vez, podem afetar o software
 - Chuva, raios, tempestade, calor excessivo

Outras Causas de Falhas

- Mal uso do software
 - Mal uso não-intencional: Usuários desatentos, ou despreparados
 - Mal uso intencional: Usuários maliciosos

Terminologia de testes

Terminologia de Testes

- Caso de Teste (Test Case)
 - Conjunto de entradas, saídas esperadas, e pré e pós condições
- Suíte de Testes (Test Suite)
 - Conjunto de casos de teste

Terminologia de Testes

- Objeto de Teste (Test Object), ou Sistema Sob Teste (System Under Test (SUT))
 - O produto de trabalho a ser testado
- Objetivo do Teste (Test Objective)
 - A razão ou propósito de realizar teste

Terminologia de Testes

- Teste Exaustivo (Exhaustive Testing)
 - Abordagem em que a suíte de testes é formada por todas possíveis combinações de entradas e pré-condições
- Explosão de Casos de Teste (Test Case Explosion)
 - Crescimento desproporcional no número de casos de teste

Terminologia de Testes

- Técnica de Teste (Test Technique)
 - Sistematização usada para projetar casos de teste
- Critério de Seleção de Teste (Test Selection Criteria)
 - Critério usado para guiar a geração ou seleção de casos de testes a fim de limitar o tamanho de uma suíte de testes

Terminologia de Testes

- Execução de Teste (Test Execution)
 - Processo de executar um componente ou sistema, produzindo resultados, para fins de teste
- Automação da Execução de Teste (Test Execution Automation)
 - Uso de ferramentas de software para automatizar a execução, a comparação dos resultados produzidos por um objeto de teste com os resultados esperados especificados, a configuração do ambiente de testes e o relato dos resultados

Teste de Software

Prof. Eiji Adachi