

# Teste de integração, sistema, regressão e (...)

Prof. André Takeshi Endo

# Níveis/Fases de Teste

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

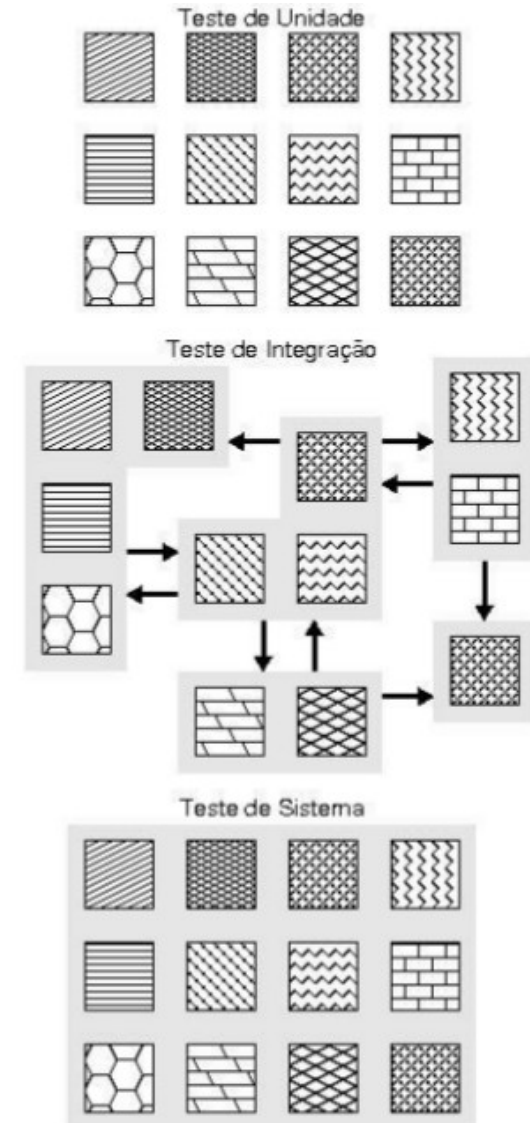
*Teste Procedimental*

Teste de Unidade

Sub-rotina ou função

Duas ou mais unidades  
Subsistema

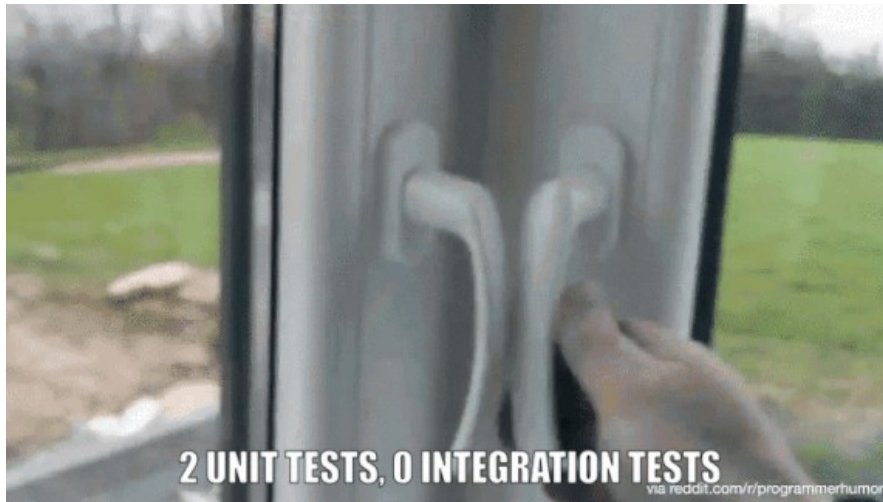
Toda aplicação



# Teste de Integração

- **Definição**
- Testes realizados durante a integração entre as unidades
- *Defeitos podem acontecer em função da integração (comunicação) entre as unidades*
- Em OO, testa a interação entre as classes

# Teste de Integração

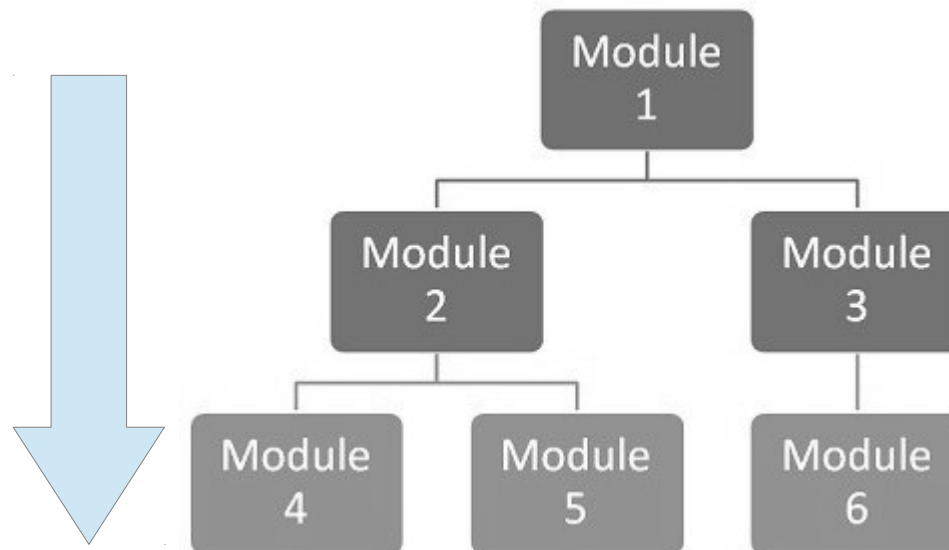


# Estratégias de Integração: Big-Bang

- Todas as unidades são integradas e testadas ao mesmo tempo
  - Após, é feito o teste do “todo” (sistema)
- Todas as unidades precisam estar prontas
  - “Roda pra ver!”
- Difícil de localizar um defeito (depuração)
  - Isolar a unidade que causa o defeito

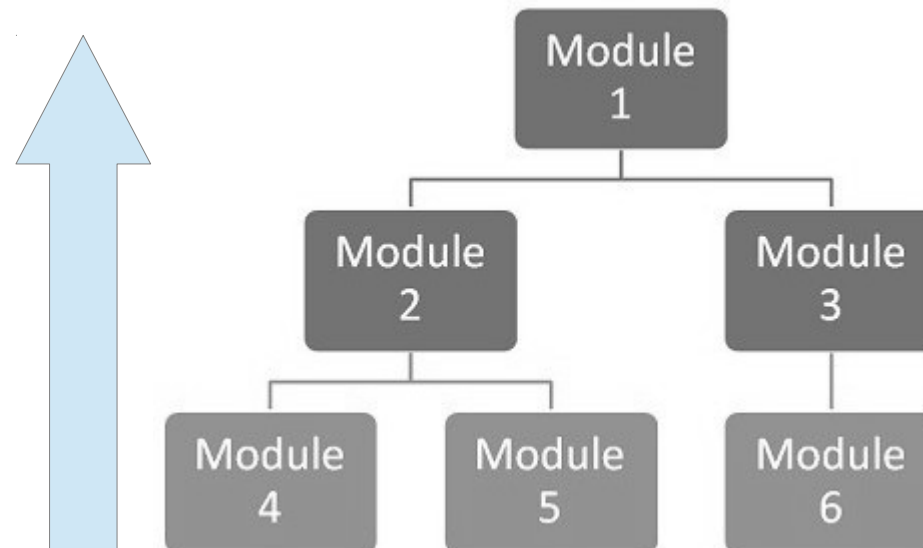
# Estratégias de Integração: Top-Down

- O teste é executado na unidade principal
- As unidades diretamente conectadas são “mockadas”
- Unidades são integradas aos poucos, substituindo os mocks pelas unidades reais



# Estratégias de Integração: Bottom-Up

- Começar os testes pelas unidades independentes
  - Sem dependências
- Passar as unidades com poucas dependências
- Subir na hierarquia das unidades
- Mocks são menos necessários



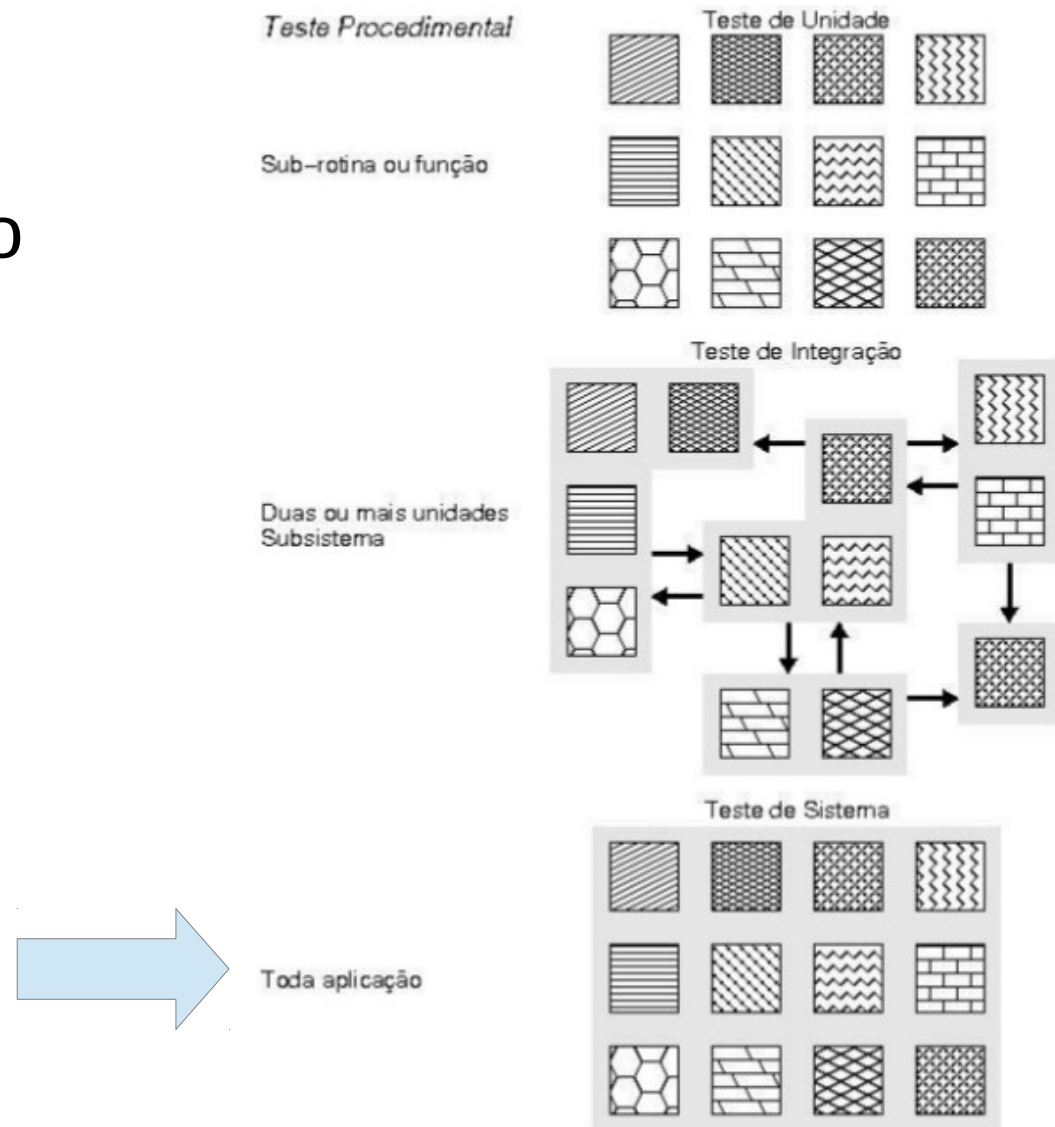
# Testando a Integração (prática)

- Não usar mocks, ou em apenas algumas partes
- Embora o teste da comunicação entre duas unidades (métodos) já é teste de integração ...
- Testes mais lentos
- ***Teste que depende de sistemas/ambientes externos***
  - Banco de dados (JPA)
  - Serviços na Internet (REST)
  - Comunicação com hardware
  - Leitura/escrita de arquivos
  - Acesso ao sistema operacional



# Níveis/Fases de Teste

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



# Teste de Sistema

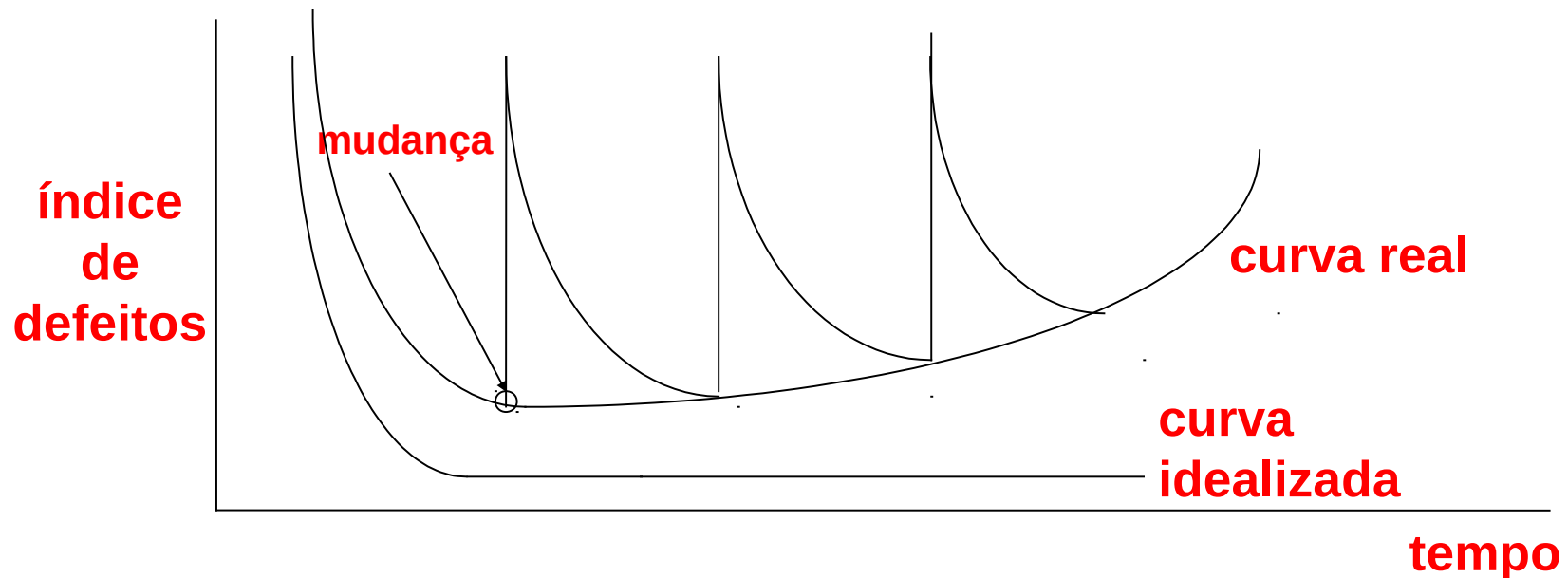
- Ao final da integração, o software está completo (pronto para uso)
  - Pode ser um **incremento** (versão funcional do software)
- O teste é baseado principalmente nos **requisitos**
- Verificar se o sw faz exatamente o que deveria
  - Teste exploratório!!!
- Baseado em requisitos funcionais e não-funcionais

# Teste de Sistema

- Foco em teste funcional (caixa-preta)
  - Critérios vistos podem ser aplicados
- Requisitos funcionais (casos de uso/estórias)
  - Teste de GUI (*graphical user interface*)
    - Como o usuário interage
- Requisitos não funcionais (especificações)
  - Teste de desempenho
  - Teste de usabilidade (IHC)
  - Teste de segurança
  - (...)

# Teste de Regressão

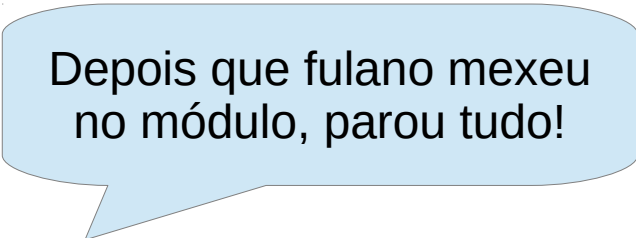
- Software evolui (sofre mudanças)
- **Como evitar que ele se deteriore?**



**CURVA DE DEFEITOS DO SOFTWARE**

# Teste de Regressão

- **Teste de regressão objetiva revelar defeitos que podem ser introduzidos acidentalmente com uma mudança**
- *Teste automatizado é um elemento-chave do teste de regressão. Por quê?*
- *É possível fazer teste de regressão com teste manual?*
- Minimizar “efeitos colaterais das mudanças”



Depois que fulano mexeu no módulo, parou tudo!



Estava funcionando antes!!!

# Teste de Aceitação

- O software é entregue ao usuário ou cliente para sua utilização
- O teste de aceitação é conduzido pelo usuário, focando principalmente na validação das funcionalidades
- Revisitar os modelos de processo cascata e incremental
  - Qual o benefício do teste de aceitação em ambos?

# Como fazer testes de aceitação?

- **Em métodos ágeis**, um teste é escrito e validado diretamente pelo cliente. O teste é então **executado pelo time de desenvolvimento**.
- *Testar diretamente com usuários*
- **Teste Alfa**
  - Acontece no local de desenvolvimento
  - Exemplo: feira de games
- **Teste Beta**
  - Acontece no ambiente real de uso
  - Exemplo: versões betas de SOs

# 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.
- [Brooks87] “No Silver Bullet: Essence and Accidents of Software Engineering”, 1987.  
[http://ieeexplore.ieee.org/xpls/abs\\_all.jsp?arnumber=1663532](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1663532)
- [IEEE90] “IEEE Standard Glossary of Software Engineering Terminology”, 1990.  
[http://ieeexplore.ieee.org/xpls/abs\\_all.jsp?arnumber=159342](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=159342)



# Bibliografia

- [Myers] G. J. Myers, T. Badgett, C. Sandler, “The art of software testing”, 2012.
- [Pezze] M. Pezze, M. Young, “Teste e análise de software: Processos, princípios e técnicas”, 2008.
- [DMJ07] DELAMARO, Márcio Eduardo; MALDONADO, José Carlos; JINO, Mario. Introdução ao teste de software. Rio de Janeiro, RJ: Elsevier, 2007. 394 p. ISBN 9788535226348.
- [UUU] Materiais didáticos elaborados pelos grupos de engenharia de software do ICMC-USP, DC-UFSCAR e UTFPR-CP.