Teste de Software

Teste de Software com Objeto *Mock*

PUC Minas

Instituto de Ciências Exatas
e Informática

Prof. Lesandro Ponciano

Departamento de Engenharia de Software
e Sistemas de Informação (DES)

Objetivos da Aula

- Apresentar o conceito de *mock object*
- Contextualizar *mock* em teste de unidade
- Discutir vantagens, limitações, problemas com *mock*
- Apresentar um exemplo

Teste de Unidade (ou Unitário)

- Em um software, classes
  - não funcionam isoladamente: se comunicam com outros elementos da aplicação
  - delegam algumas das suas funções para outras classes

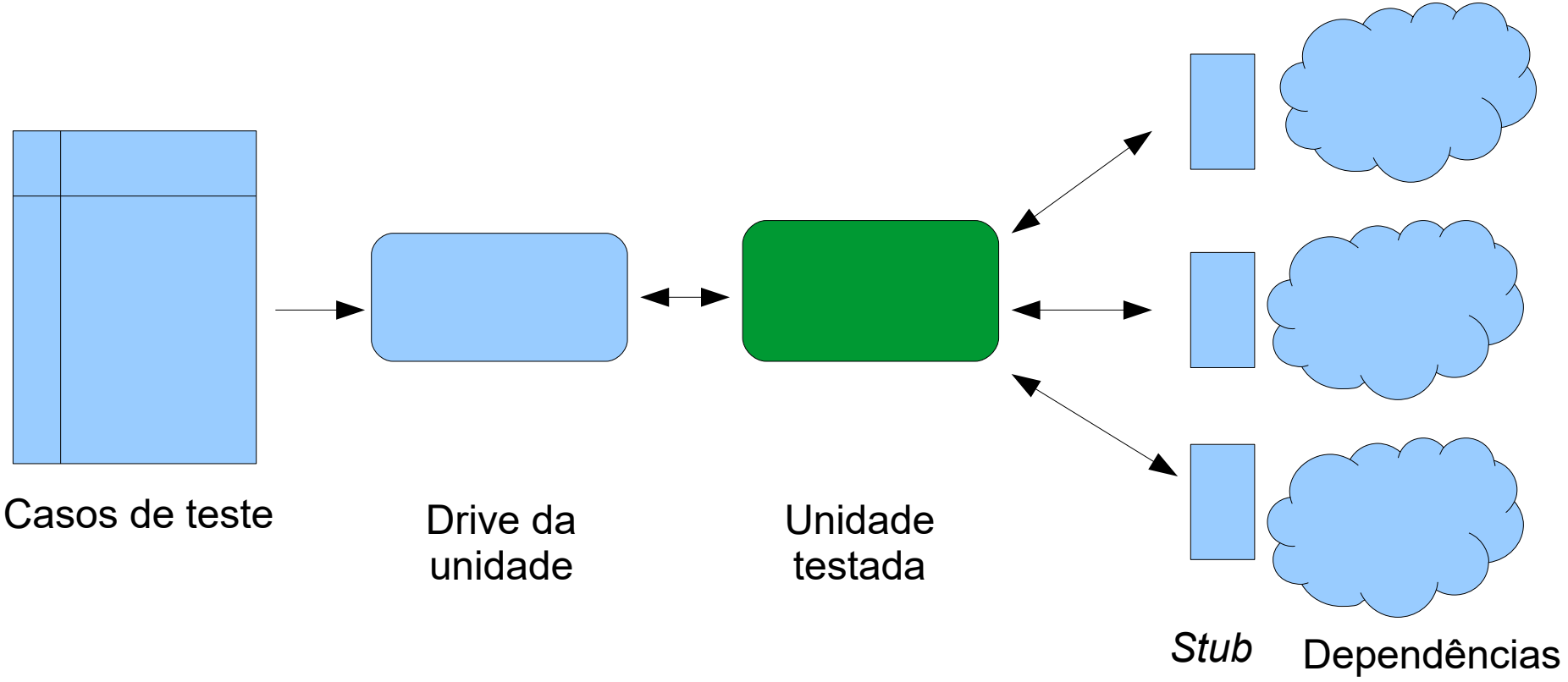
- Ao construir um teste unitário é necessário isolar a classe que está sendo testada para que nenhuma outra classe seja envolvida no teste

Problema em Teste Unitário

- Há diversos objetos cujos comportamentos são difíceis de serem incorporados em testes de unidade

- Exemplos
  - Chamada remota a um servidor
  - Objetos associados ao tempo
  - Objetos lentos (como acesso a banco de dados) que precisam ser instanciados antes dos testes

Visão Geral do Contexto



Objetos *Mock* (*mock objects*)

- Definição do Google
  - *Mock. Adjective: not authentic or real, but without the intention to deceive.*

- Precisamos de um objeto que
  1) finja ser o gerador
  2) devolva um resultado esperado

- Objetos *mock*
  - imitam objetos reais
  - simulam o comportamento de objetos reais complexos

Usando *Mock*

- Usar uma interface para descrever o objeto

- Implementar
  - a interface com código real para ser colocado em produção
  - a interface com um objeto *mock* para ser testado

Atividades ao se Usar *Mock*

1) Crie as instâncias dos objetos *mock*
2) Configure os estados dos objetos *mock*
3) Configure as expectativas dos objetos *mock*
4) Invoque o código de domínio passando como parâmetros os objetos *mock*
5) Realize as asserções (*assert()*, *fail()*) verificando a consistência dos objetos *mock*.

Objetos *Mock* no Teste

- Transparência para o código sendo testado

- O código a ser testado
  - refere-se ao objeto do qual ele depende por meio da interface
  - não tem idéia se está sendo usado o código real ou o código *mock*

Algumas Razões para usar *Mock*

- Adiar decisões ou implementações de infraestrutura ou classes
- Configuração de estados complexos de objetos
- Testar condições ou exceções difíceis pouco frequentes, difíceis de reproduzir
- Força o design por interface
- Diminui o tempo de execução dos testes
- Força a tornar os objetos mais coesos

Problemas/Limitações

- Existe o risco de objetos *mock* ou métodos de testes conterem defeitos
- Dificuldade, em alguns casos, de se criar objetos *mock* para bibliotecas externas.
- Criação manual de vários objetos *mock*

Referências

- Nakagawa, Elisa Yumi. (2015) Teste de Software - Parte 2. https://edisciplinas.usp.br/pluginfile.php/317501/mod_resource/content/1/Aula08_TesteSoftware_Parte2.pdf
- Dependable Software Systems. Topics in Mutation Testing and Program Perturbation. SERG https://slideplayer.com/slide/7421728/
- Myers, Glenford J. et al (2004) "The Art of Software Testing." 2ed. New York, NY, USA: John Wiley & Sons. .