Teste Baseado em Modelo

Prof. André Takeshi Endo

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS CORNÉLIO PROCÓPIO

Teste Baseado em Modelo

- Critérios de teste caixa-preta
  - Particionamento em classes de equivalência
  - Análise de valor limite
  - Tabela de decisão
- **Teste baseado em modelo (TBM)**
  - Associado em geral ao teste caixa-preta e de sistema
  - No entanto, pode ser aplicado em qualquer nível ou mesmo para teste caixa-branca

Intuição

- TBM pode ser aplicado em diversos contextos, mas no geral, no cenário discutido a seguir.
- PCE, AVL e TD* são critérios que focam cenários com no máximo duas sequências de eventos
  - Por exemplo, (1) dados de formulário e (2) mensagens de erro ou sucesso!
- A funcionalidade a ser testada pode envolver uma série de eventos
  - Carrinho de compra em um e-commerce
  - Operações bancárias (vários passos)
  - Sistemas embarcados (climatizador)

*Particionamento em classes de equivalência (PCE)
Análise de valor limite (AVL)
Tabela de decisão (TD)

Teste baseado em modelo

- É aplicado em 4 passos principais:
  - **Modelagem**: elaborar de um modelo de teste referente a funcionalidade a ser verificada
  - **Geração de casos de teste**: usando modelo, casos de teste (sequências de teste) são geradas
  - **Concretização**: Os testes gerados são abstratos (codificar o teste para executar no SUT)
  - **Execução dos testes**: usando os casos de teste concretizados, os mesmos são executados no SUT

Exemplo

- *Uma pilha de strings com tamanho limitado (Moodle)*
- *Escreva uma implementação para a estrutura de dados pilha; tal pilha deve ser capaz de empilhar e desempilhar strings. Considere que:*
- *Tal pilha possui um tamanho limitado (que deve ser passado como parâmetro no construtor).*
- *Adicione métodos para empilhar, desempilhar e verificar se a pilha está vazia.*
- *Crie duas classes de exceção que devem ser do tipo "checked exception": PilhaVaziaException e PilhaCheiaException.*
- *PilhaVaziaException deve ser lançada caso tente desempilhar a pilha sem elementos.*
- *PilhaCheiaException deve ser lançada caso tente empilhar um elemento na pilha cheia.*

Como aplicar?

- **Modelagem:** elaborar de um modelo de teste referente a funcionalidade a ser verificada
- Usaremos:
  - Máquinas de estados* → diagrama de máquinas de estados da UML
- Existem outras técnicas:
  - *Modelos orientados a eventos* → *Event Sequence Graph* (ESG)
  - …..

*Serão usadas apenas as estruturas mais básicas.

Modelagem

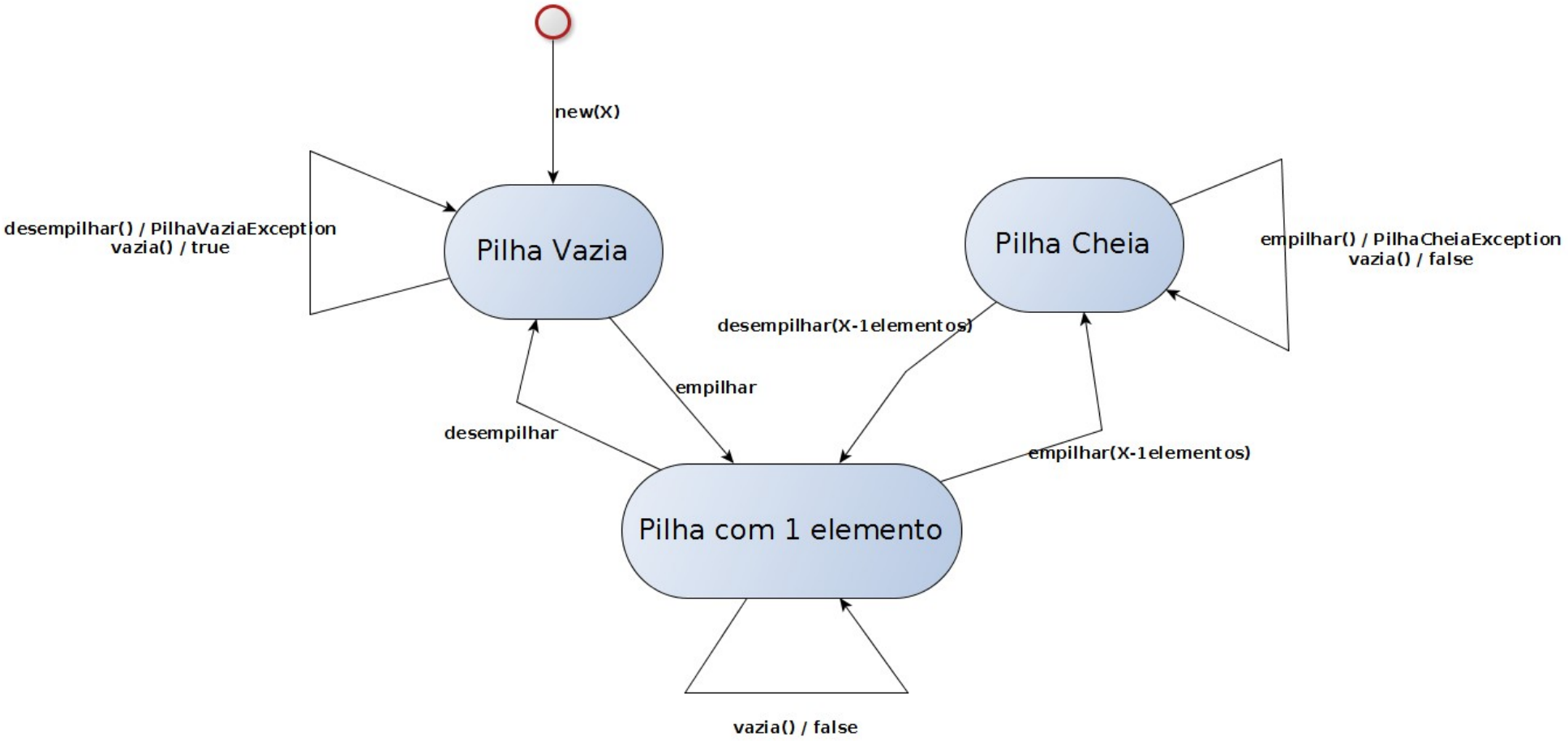
- Quais os eventos no exemplo?
- Baseado na lista de métodos públicos
  - New (construtor)
  - empilhar(objeto)
  - Objeto = desempilhar()
  - Boolean vazia()
  - PilhaCheiaException
  - PilhaVaziaException

Modelagem

- Máquina de estados (UML)
  - Quais os estados?
  - Quais os eventos?
  - O que faria a mudança de estados (transições)?

Modelagem

- Máquina de estados (UML) – possível solução
  - Sugestão: usar o yEd Graph Editor*



* https://www.yworks.com/downloads#yEd

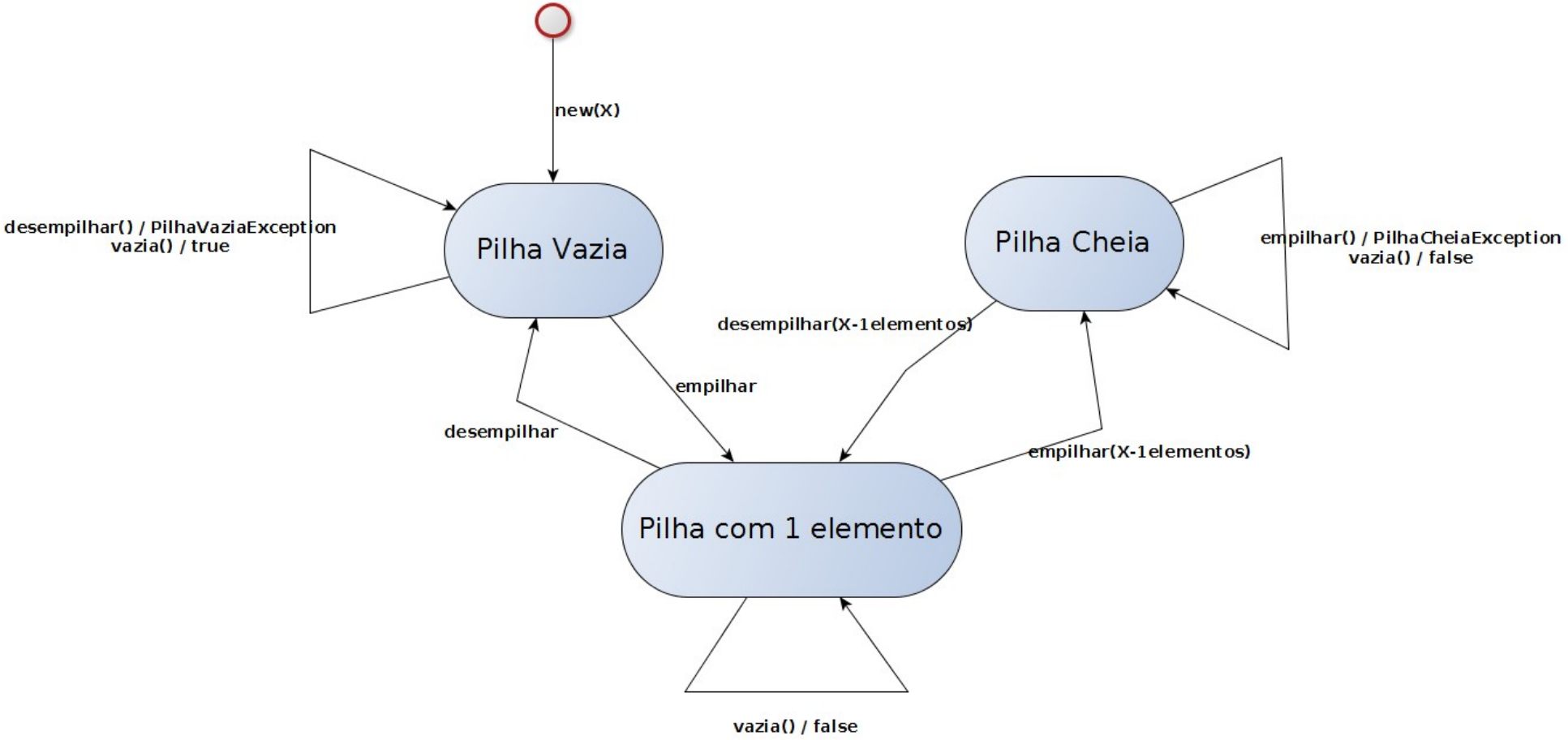
Como aplicar?

- **Geração de casos de teste:** usando modelo de teste, casos de teste (sequências de teste) são geradas
- Algoritmos e ferramentas podem ser aplicados para gerar automaticamente os casos de teste
- No caso do nosso modelo, queremos que cada transição seja executada ao menos uma vez
  - Critério de teste: todas as transições*

* Este critério é equivalente a todos-arcos usado no teste caixa-branca.

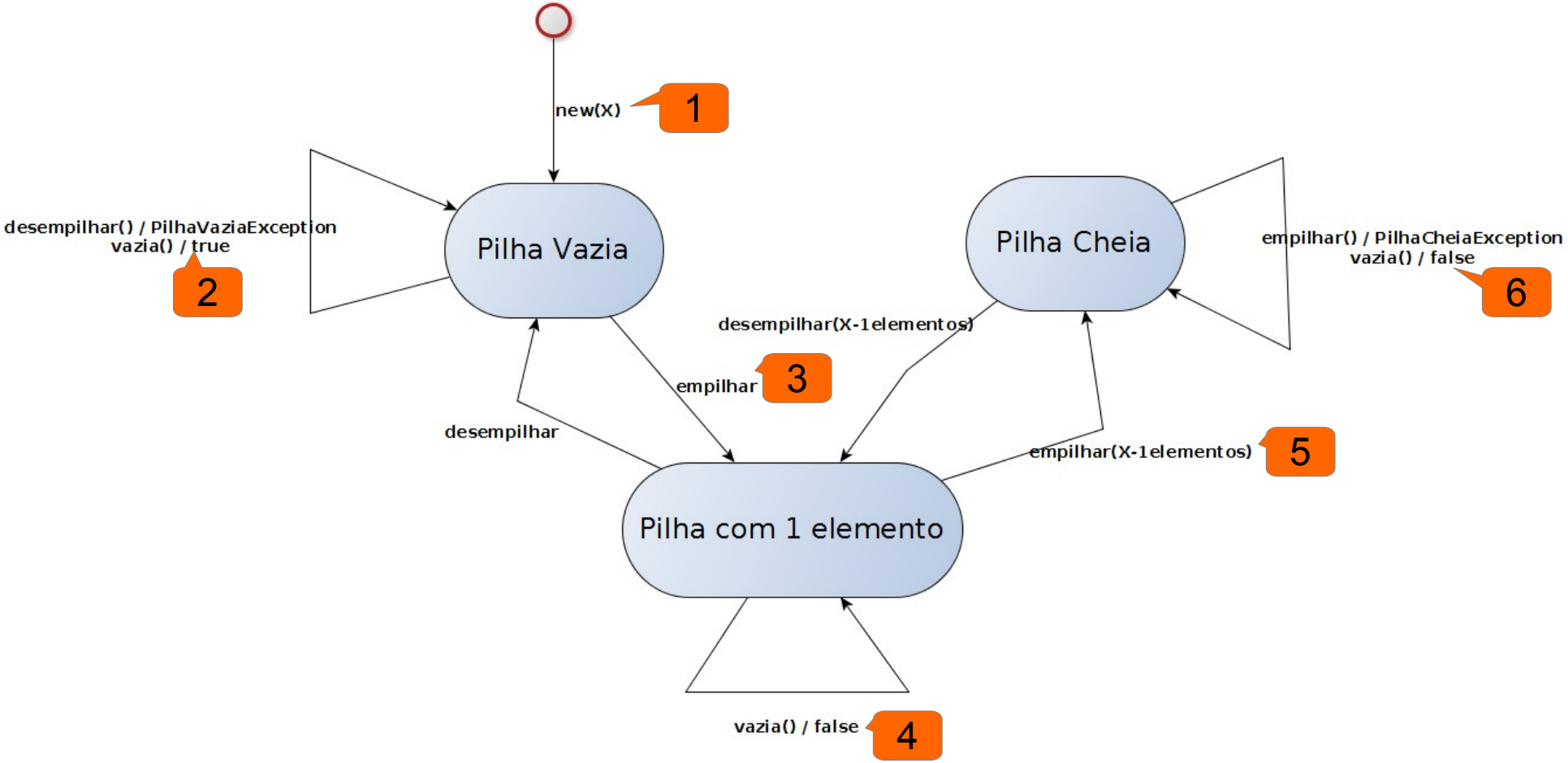
Geração de casos de teste

- Elabore dois casos de teste: (i) testar o método vazia() em cada estado e (ii) lançar as exceções



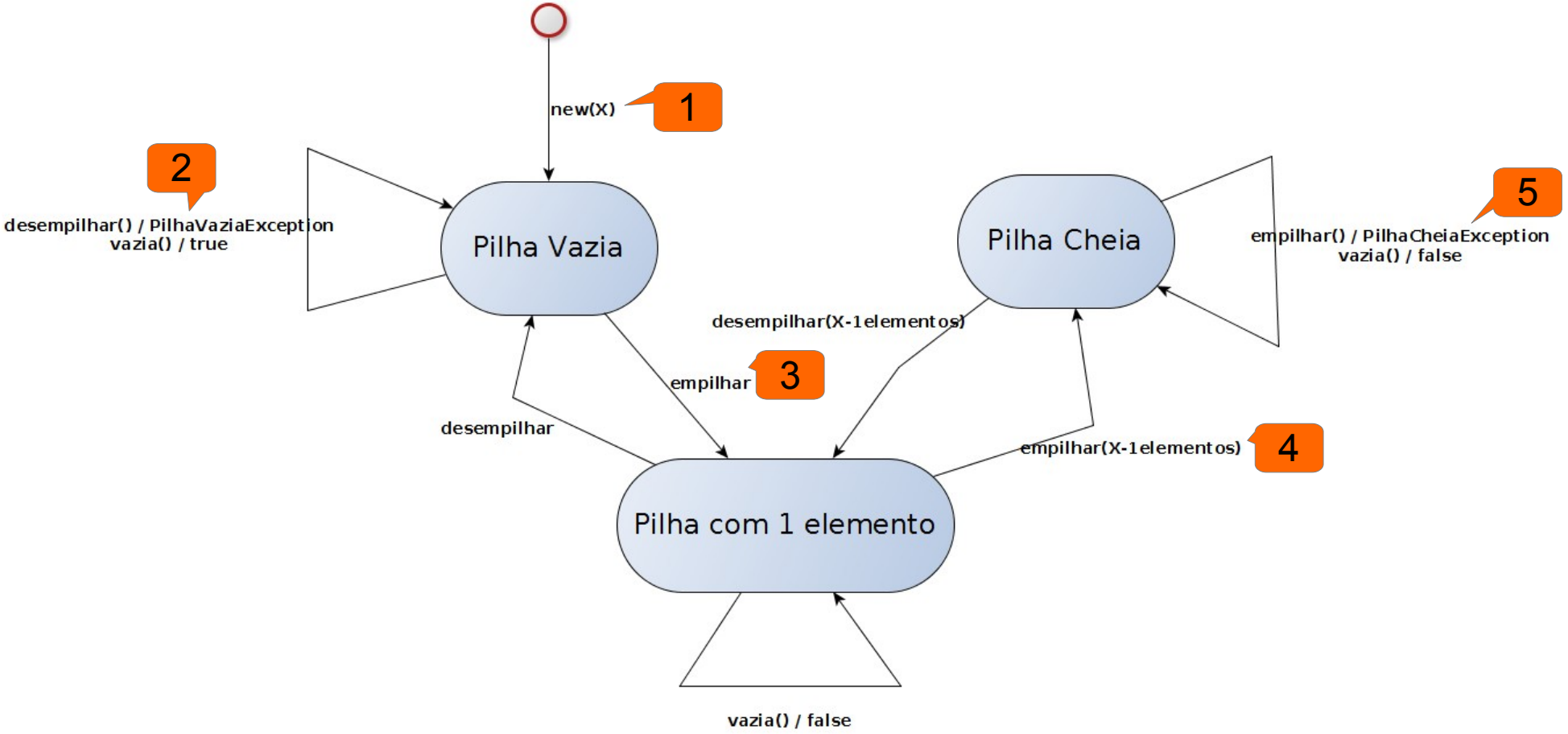
Geração de casos de teste

- Elabore dois casos de teste (i) testar o método vazia() em cada estado e (ii) lançar as exceções



Geração de casos de teste

- Elabore dois casos de teste (i) testar o método vazia() em cada estado e (ii) lançar as exceções



Geração de casos de teste

- Caso de teste 1
  - *new(x), vazia()/true, empilhar, vazia()/false, empilhar(x-1 elementos), vazia()/false*
- Caso de teste 2
  - *new(x), desempilhar/pilhaVaziaException, empilhar, empilhar(x-1 elementos), empilhar/pilhaCheiaException*

- ***Faltou cobrir alguma transição? Elabore um CT para cobrir esta transição***

Como aplicar?

- **Concretização:** Os testes gerados são abstratos (codificar o teste para executar no SUT)
- Dá para executar os CTs anteriores direto no SUT?
  - **Não! São abstratos**
- Existem várias estratégias
  - ***Vamos concretizar na forma de código usando JUnit!!!!***

Concretização

- Usando JUnit, como faríamos a concretização deste caso de teste?
  - Chamando métodos na ordem especificada e verificando as saídas com assertivas
- Caso de teste 1
  - *new(x), vazia()/true, empilhar, vazia()/false, empilhar(x-1 elementos), vazia()/false*

Como aplicar?

- **Execução dos teste:** usando os casos de teste concretizados, os mesmos são executados no SUT

Execução dos testes

- Caso um código automatizado seja desenvolvido, basta mandar executar o código!
- É possível, principalmente em teste de sistema, realizar a execução de maneira manual
  - Assim, a concretização e a execução é realizada manualmente pelo testador!

Exercício

- Concretize e execute os outros dois casos de teste para este exemplo.

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
- [IEEE90] “IEEE Standard Glossary of Software Engineering Terminology”, 1990.
  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.