Critérios de Teste Caixa-Preta

Prof. André Takeshi Endo

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS CORNÉLIO PROCÓPIO

Técnicas de Teste

- **Critérios de Teste**
- Define uma ***maneira sistemática*** e planejada para conduzir testes
- Quais casos de teste com **maior chance** de revelar defeitos

Técnicas de Teste

- **Teste Caixa-Preta**
  - Testes baseados na especificação (de requisitos)
  - A funcionalidade testada é considerada uma caixa preta
- **Teste Caixa-Branca**
  - Testes baseados na estrutura interna do programa
  - Analisar o código fonte (caixa branca)

Teste Caixa-Preta

- Vários critérios de teste
- **Particionamento em classes de equivalência**
- **Análise de valor limite**
- Tabela de decisão

Part. em Classes de Equivalência

- **Intuição (i)**
- Domínio de entrada do programa sob teste
- Exemplo: método de uma classe
  - Parâmetros de entrada
  - Variáveis globais (atributos da classe)
  - Objetos representando o estado atual

Part. em Classes de Equivalência

- **Intuição (ii)**
- Particionar o domínio em regiões
- Regiões têm valores igualmente úteis do ponto de vista de teste
- Valores são selecionados de cada região
- Se o programa funciona para um caso dessa região, ***é improvável*** que outro caso de teste dessa mesma região irá revelar um novo defeito
- Exemplo: triângulo (caso de teste para o triângulo equilatero)

Part. em Classes de Equivalência

- ***Como projetar os casos de teste usando particionamento em classes de equivalência? [Myers]***
- (1) Identificar as classes de equivalência
- (2) Definir os casos de teste

Part. em Classes de Equivalência

- **(1) Identificar as classes de equivalência**
- Identificar uma condição de entrada
- Particionar em 2 ou mais classes
- Identificar 2 grupos de classes
  - Válidas e inválidas
- *Identificar as classes é um processo heurístico*
  - Cada testador terá um resultado diferente

Part. em Classes de Equivalência

- **(1) Identificar as classes de equivalência**
- Processo heurístico (diretrizes):
- (i) Condição de entrada especifica limite de entradas
- Exemplo: o número de itens pode ser de 1 até 999.
  - Uma classe válida
  - Duas classes inválidas

Part. em Classes de Equivalência

- **(1) Identificar as classes de equivalência**
- Processo heurístico (diretrizes):
- Outro exemplo: “Um até seis proprietários podem ser listados por automóvel”.
  - Uma classe válida
  - Duas classes inválidas

Part. em Classes de Equivalência

- **(1) Identificar as classes de equivalência**
- Processo heurístico (diretrizes):
- (ii) Se a condição de entrada especifica um conjunto fechado de valores e há razão para acreditar que o programa trata cada um de forma diferente
- Exemplo: “Tipo de veículo deve ser ônibus, caminhão, carro e moto”.
  - Uma classe válida para cada valor
  - Uma classe inválida (fora do conjunto)

Part. em Classes de Equivalência

- **(1) Identificar as classes de equivalência**
- Processo heurístico (diretrizes):
- (iii) Condição de entrada especifica uma situação *"deve ser"*
- Exemplo: "O primeiro caracter *deve ser* uma letra".
  - Uma classe válida
  - Uma classe inválida

Part. em Classes de Equivalência

- **(1) Identificar as classes de equivalência**
- Processo heurístico (diretrizes):
- (iv) Se, por alguma razão, acredita-se que o programa não irá tratar todos os elementos da classe igualmente, divida a classe de equivalência em outras classes menores.
  - Exemplo: triângulo isósceles

Part. em Classes de Equivalência

- **(1) Identificar as classes de equivalência**
- Processo heurístico (diretrizes):
- (v) as classes de equivalência podem ser mapeadas com base nas saídas.
  - Exemplo: programa que recebe um inteiro e responde se o mesmo é primo ou não

Part. em Classes de Equivalência

- **(2) Definir os casos de teste**
- Derivar casos de teste das classes de equivalência
- (i) Atribuir um valor para cada classe
- (ii) Escrever casos de teste que cubram o máximo de classes válidas (até cobrir todas)
- (iii) Escrever um caso de teste para cobrir *uma, e somente uma,* classe inválida

Por que o passo (iii) cobre só uma?

Part. em Classes de Equivalência

- Uma forma de enumerar as classes de equivalência

Condição de entrada	Classes válidas	Classes Inválidas

Exemplo

- Exemplo [adaptado de DMJ07]
- *Um programa solicita do usuário um inteiro positivo no intervalo entre 1 e 20 e então solicita uma cadeia de caracteres desse comprimento. Após isso, o programa solicita um caracter e retorna a posição na cadeia em que o caracter é encontrado pela primeira vez ou uma mensagem indicando que o caracter não está presente na cadeia. Para qualquer entrada inválida, uma mensagem expressiva de erro deve ser mostrada e a entrada deve ser solicitada novamente.*

Exemplo

- **Condições de entrada**
  - *Comprimento da cadeia (int)*
  - *String*
  - *Caracter a ser buscado*

Exemplo

- **Enumerando as classes de equivalência**

Condição de entrada	Classes válidas	Classes Inválidas
Comprimento da cadeia (cc)	1 <= cc <= 20	cc < 1 cc > 20
String (s)	s no tamanho fornecido	s menor que cc s maior que cc
Caracter a ser buscado (c)	c em s	c não está em s

Exemplo

- **Atribuindo valores para cada classes**

Condição de entrada	Classes válidas	Classes Inválidas
Comprimento da cadeia (cc)	1 <= cc <= 20 **7**	cc < 1 **-20** cc > 20 **35**
String (s)	s no tamanho fornecido **abcdefg**	s menor que cc **abc** s maior que cc **abcefgh**
Caracter a ser buscado (c)	c em s **c**	c não está em s **y**

Exemplo

- **Elaborar CTs que cubram as classes válidas**
- **Um CT para cada classe inválida**

Condição de entrada	Classes válidas	Classes Inválidas
Comprimento da cadeia (cc)	1 <= cc <= 20 **7**	cc < 1 **-20** cc > 20 **35**
String (s)	s no tamanho fornecido **abcdefg**	s menor que cc **abc** s maior que cc **abcefgh**
Caracter a ser buscado (c)	c em s **c**	c não está em s **y**

Exemplo

- **Resposta (entradas; saídas esperadas):**
- (7, “abcdefg”, 'c'; “posição 3”)
- (-20, “abcdefg”, 'c'; “comprimento inválido”)
- (35, “abcdefg”, 'c'; “comprimento inválido”)
- (7, “abc”, 'c'; “cadeia com tamanho inconsistente”)
- (7, “abcdefgh”, 'c'; “cadeia com tamanho inconsistente”)
- (7, “abcdefgh”, 'y'; “caracter não está na cadeia fornecida”)

Part. em Classes de Equivalência

- **Aviso!!!!!**
- *"Just because two items/values are supposed to be in the same class and behave the same, it does not mean they DO behave the same."* (J. Soderstrom, in stackoverflow.com)

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.