Módulo I – Fundamentos Computacionais
Sessão 5 - Aula 2 – Teste de Software – Pairwise – Parte 1

Prof. Dr. Luiz Alberto Vieira Dias
Prof. Dr. Lineu Mialaret

ITAú Analytics – Fundamento de Computação – CEDS 111
2º SEMESTRE 2018

Teste de Software
Pairwise Testing - Parte 1

CEDS -111 - Teste de Software - Sessão 5 - Aula 2

Prof. Dr. Luiz Alberto **Vieira Dias** (ITA)
Prof. Dr. Adilson Marques da **Cunha** (ITA)
Prof. Dr. **Lineu** Mialaret (ITA/IFSP)
Prof. Dr. Paulo Marcelo **Tasinaffo** (ITA)

A preparação destes slides contou com o apoio dos Alunos do ITA

- Lucas Nadalete, MSc
  - D. Montini, DSc
  - G. Matuck, MSc
  - E. Mineda, MSc

Considerações iniciais para *Pairwise Testing* – Exemplo 1

- Um *website* deve operar corretamente com **8 *browsers*** diferentes:
  - Internet Explorer 5.0, 5.5, 6.0
  - Netscape 6.0, 6.1, 7.0
  - Mozilla 1.1
  - Opera 7
- Com **3 plug-ins**: RealPlayer, Media Player ou nenhum
- Com **6 sistemas operacionais (SO) cliente:** Windows 98, ME, NT, 2000, XP e Vista
- Recebendo páginas de ***3 servers*** diferentes: IIS, Apache e Weblogic
- Rodando em **3 SO para servers** diferentes: Windows NT, 2003 e Linux

Considerações iniciais para Pairwise Testing

- Browsers: 8
- Plug-ins: 3
- SO cliente: 6
- Servers: 3
- SO servers: 3

- Quantas escolhas (combinações, permutações) são possíveis?

Arranjos

- "**Arranjos** simples de n elementos tomados p a p (p ≤ n) são os diferentes agrupamentos ordenados que se podem formar com p dos n elementos de um conjunto dado"

$$A_{n,p} = n! /(n-p)!$$

- Em **arranjos** a ordem **importa**, em **combinações** a ordem **não importa**

Cadeado com “combinação”?

Na verdade arranjo!

Permutações e Combinações

- **Combinações:** a ordem não importa
- **Arrajos e Permutações:** a ordem conta
- Cada um pode ser:
  - Com repetições
  - Sem repetições

Regras gerais de Análise Combinatória

"**Regra do Produto:** A Regra do Produto diz que se um elemento H pode ser escolhido de m formas diferentes e se depois de cada uma dessas escolhas, um outro elemento M pode ser escolhido de n formas diferentes, a escolha do par (H,M) nesta ordem poderá ser realizada de m.n formas"

Número de escolhas para Exemplo 1

$$\text{Escolhas} = n_1 x\ n_2\ x \ldots x\ n_n =$$

$$8 \times 3 \times 6 \times 3 \times 3 = 1296$$

Para o Teste Exaustivo (Exemplo 1) são necessários = **1296** casos de teste (possibilidades)!

Considerações iniciais para *Pairwise Testing* (3) – Exemplo 2

- Um banco criou um novo sistema de processamento de dados. Este banco tem clientes de 4 tipos: consumidores comuns (CMU), “very important consumers” (VIC), comerciantes (COM) e sem fins lucrativos (SFL)
- Ele tem 5 tipos de contas: corrente, poupança, hipoteca, empréstimo ao consumidor e comercial
- Opera em 6 estados (USA): CA, NV, UT, AR, NM, AK

Exemplo 2

- **4** Tipos de clientes (CMU, VIC, COM, SFL)
- **5** Tipos de contas (COR, POUP,HIP,CONS,COM)
- Opera em **6** estados americanos (CA, NV, UT, AR, NM, AK)

- Casos de Teste = 4 x 5 x 6 = **120**

- **Total = 120 Casos de Teste(possibilidades)!**

Resumo

- Escolhas da Web = 1296
- Escolhas do banco = 120

O que estas diferentes situações têm em comum?

- Cada uma tem um número "grande" de possibilidades de escolha (Casos de Teste) e cada uma precisa ser testada
- Pode ser muito arriscado não testar todas as possibilidades (Foguete Ariane 5, Wall Street em 1987)
- Cada uma tem um número de possibilidades tal que não permite ao usuário (falta de "grana" ou tempo) construir e rodar todos os casos de teste necessários; simplesmente eles (os testes) são muitos!

O que fazer? (Política)

- É preciso selecionar um subconjunto, de porte “razoável” (subjetivo), de tal modo que seja possível testar o SUV (Software Under Test) ao máximo, dentro das possibilidades do usuário

Como fazer? (Estratégia)(1/3)

- Gerar uma lista de possibilidades, da pior para a melhor:

Lista (em ordem de melhora – pior antes):

1. Não testar
2. Testar todas as possibilidades e atrasar o projeto. Pode resultar no fechamento das portas da empresa!

Como fazer? (Estratégia)(2/3)

3. Escolher um ou dois testes e torcer para que funcione
4. Escolher testes que já rodaram, talvez como parte do desenvolvimento (alfa teste). Incorporá-los no Artefato Plano de Teste (ou equivalente) e rodá-los de novo. Impressiona os clientes, mas não resolve
5. Escolher os que sejam fáceis de criar e os rode. Ignorar se isto traz alguma informação útil sobre a qualidade do produto

Como fazer? (Estratégia)(3/3)

6. Fazer uma lista de todas as possibilidades possíveis e rodar algumas poucas do início da lista
7. Fazer uma lista de todas as possibilidades e escolher um subconjunto randômico
8. De alguma maneira **mágica**, selecionar um subconjunto pequeno que irá determinar um grande número de defeitos! Mais do que seria esperado!

A tática (*scheme*) vencedora

- A possibilidade 8 é claramente vencedora, mas como escolher tal subconjunto **mágico** de casos de teste?

A resposta

- Não tentar testar todas as possibilidades sem repetição possíveis de variáveis (ou funções) e os valores que elas podem tomar, mas testar todos os PARES DE VARIÁVEIS (FUNÇÕES). Isto é ***pairwise testing*** !

- Esta técnica reduz significantemente o número de casos de teste que devem ser criados e rodados

Base matemática

- **Não há** (para o aparecimento de defeitos no SW), mas para Matrizes Ortogonais, OA, há uma bem formulada teoria! O resultado é comprovado estatisticamente

- **Hipótese:** a maioria dos defeitos é single-mode (a função sob teste não funciona – usa-se o PyUnit) ou double-mode (é o par: **esta** função/método com **aquela** função/método que não funciona, embora os outros pares funcionem adequadamente)

Base matemática 2

- ***Pairwise*** é uma técnica que determina **todas** as funções/métodos que devem ser testadas para localização de defeitos de SW gerados por *single* e *double-modes*

- O **sucesso** desta técnica, verificado estatisticamente, documentado e não documentado, é a grande motivação para seu uso

FIM Parte 1

Arranjos,Permutações e Combinações

- **Combinações:** a ordem não importa
- **Arranjos e permutações** (caso particular de aranjos): a ordem conta
- Cada um pode ser:
  - Com repetições
  - Sem repetições

Exemplo 3 – Arranjos com repetições

- Um sistema tem 4 parâmetros de entrada e cada um deles pode tomar um de 3 valores diferentes.

- Caso de arranjo com repetições, a ordem conta!

$$A_{rep}(m,p) = m^p$$

Arranjos com repetições

- Sendo ***n*** o número de coisas a escolher, e você escolhe ***r*** dentre elas (Repetição permitida, a ordem é levada em conta)

$$A_R = n^r$$

- No caso do Exemplo 4:

$$A_R = (Val)^{par}$$

- O número total de arranjos c/ repetição, ou os caso **casos de teste**, é:

$$3^4 = 81$$

- Utilizando o **pairwise testing** apenas **9 (nove) casos de teste** são necessários! **Como?**

Exemplo 4

- Um sistema tem 13 parâmetros e cada um deles pode tomar um de 3 valores diferentes
- O número de combinações é 3^{13}
- **$3^{13} = 1.594.323$** combinações!
- Utilizando o pairwise testing apenas 27 casos de teste são necessários, segundo Copeland (2007). **Ver as tabelas de Taguchi, na Internet!**

Permutações e Combinações

- **Combinações:** a ordem não importa
- **Arranjos e Permutações:** a ordem conta
- Cada um pode ser:
  - Com repetições
  - Sem repetições

Taguchi Orthogonal Array Selector

Number of Levels

Number of Parameters

	2	3	4	5
2	P=2, L=2	P=2, L=3	P=2, L=4	P=2, L=5
3	P=3, L=2	P=3, L=3	P=3, L=4	P=3, L=5
4	P=4, L=2	P=4, L=3	P=4, L=4	P=4, L=5
5	P=5, L=2	P=5, L=3	P=5, L=4	P=5, L=5
6	P=6, L=2	P=6, L=3	P=6, L=4	P=6, L=5
7	P=7, L=2	P=7, L=3	P=7, L=4	P=7, L=5
8	P=8, L=2	P=8, L=3	P=8, L=4	P=8, L=5
9	P=9, L=2	P=9, L=3	P=9, L=4	P=9, L=5
10	P=10, L=2	P=10, L=3	P=10, L=4	P=10, L=5
11	P=11, L=2	P=11, L=3		P=11, L=5
12	P=12, L=2	P=12, L=3		P=12, L=5
13	P=13, L=2	P=13, L=3		
14	P=14, L=2	P=14, L=3		
15	P=15, L=2	P=15, L=3		
16	P=16, L=2	P=16, L=3		
17	P=17, L=2	P=17, L=3		
18	P=18, L=2	P=18, L=3		
19	P=19, L=2	P=19, L=3		
20	P=20, L=2	P=20, L=3		
21	P=21, L=2	P=21, L=3		
22	P=22, L=2	P=22, L=3		
23	P=23, L=2	P=23, L=3		
24	P=24, L=2			
25	P=25, L=2			
26	P=26, L=2			
27	P=27, L=2			
28	P=28, L=2			
29	P=29, L=2			
30	P=30, L=2			
31	P=31, L=2			

THESE COMBINATIONS ARE NOT AVAILABLE – TRY RUNNING A SMALLER EXPERIMENT

Directions: Click on the appropriate square, according to the number of parameters and levels in your experiment.

Site: **www.freequality.org**

TAGUCHI ARRAY SELECTOR

Levels->

Parameters

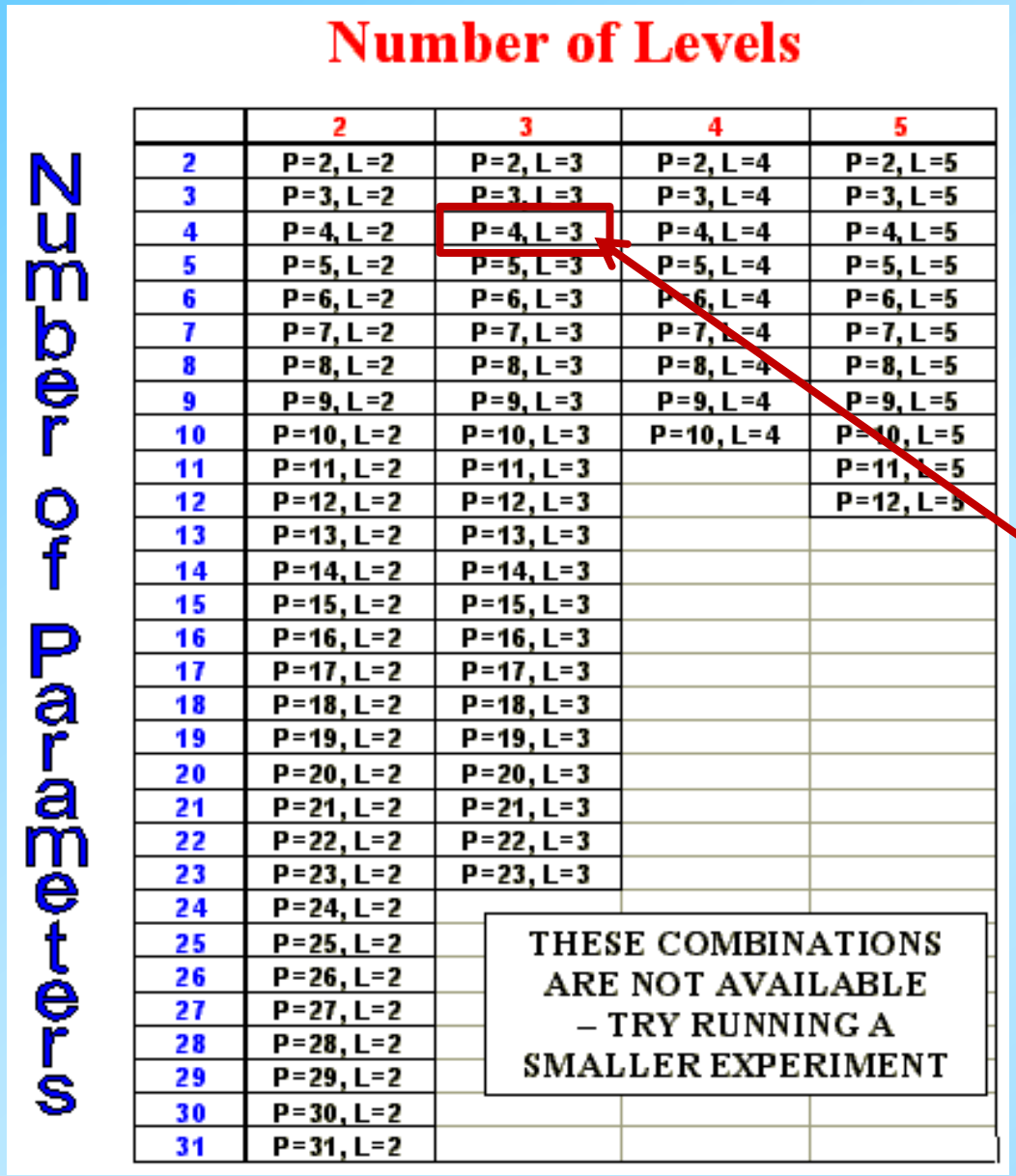
	2	3	4	5
2	4	9	16	25
3	4	9	16	25
4	8	9	16	25
5	8	18	16	25
6	8	18	32	25
7	8	18	32	50
8	12	18	32	50
9	12	27	32	50
10	12	27	32	50
11	12	27		50
12	16	27		50
13	16	27		
14	16	36		
15	16	36		
16	32	36		
17	32	36		
18	32	36		
19	32	36		
20	32	36		
21	32			
22	32			
23	32			
24	32			
25	32			
26	32			
27	32			
28	32			

Orthogonal Array

$L_9(3^4)$ ➪ Resultado = 9 experimentos

(4 parâmetros: 3 valores (*levels*) para cada parâmetro)

Experimentos	Coluna 1 – P1	Coluna 2 – P2	Coluna 3 – P3	Coluna 4 – P4
1	1	1	1	1
2	1	2	2	2
3	1	3	3	3
4	2	1	2	3
5	2	2	3	1
6	2	3	1	2
7	3	1	3	2
8	3	2	1	3
9	3	3	2	1



	2	3	4	5
2	P=2, L=2	P=2, L=3	P=2, L=4	P=2, L=5
3	P=3, L=2	P=3, L=3	P=3, L=4	P=3, L=5
4	P=4, L=2	P=4, L=3	P=4, L=4	P=4, L=5
5	P=5, L=2	P=5, L=3	P=5, L=4	P=5, L=5
6	P=6, L=2	P=6, L=3	P=6, L=4	P=6, L=5
7	P=7, L=2	P=7, L=3	P=7, L=4	P=7, L=5
8	P=8, L=2	P=8, L=3	P=8, L=4	P=8, L=5
9	P=9, L=2	P=9, L=3	P=9, L=4	P=9, L=5
10	P=10, L=2	P=10, L=3	P=10, L=4	P=10, L=5
11	P=11, L=2	P=11, L=3		P=11, L=5
12	P=12, L=2	P=12, L=3		P=12, L=5
13	P=13, L=2	P=13, L=3		
14	P=14, L=2	P=14, L=3		
15	P=15, L=2	P=15, L=3		
16	P=16, L=2	P=16, L=3		
17	P=17, L=2	P=17, L=3		
18	P=18, L=2	P=18, L=3		
19	P=19, L=2	P=19, L=3		
20	P=20, L=2	P=20, L=3		
21	P=21, L=2	P=21, L=3		
22	P=22, L=2	P=22, L=3		
23	P=23, L=2	P=23, L=3		
24	P=24, L=2			
25	P=25, L=2			
26	P=26, L=2			
27	P=27, L=2			
28	P=28, L=2			
29	P=29, L=2			
30	P=30, L=2			
31	P=31, L=2			

4 parâmetros

3 valores (***levels***) podem ser assumidos por cada parâmetro

$L_9(3^4)$

9 Casos deTeste!

Number of Levels

Number of Parameters	2	3	4	5
2	P=2, L=2	P=2, L=3	P=2, L=4	P=2, L=5
3	P=3, L=2	P=3, L=3	P=3, L=4	P=3, L=5
4	P=4, L=2	P=4, L=3	P=4, L=4	P=4, L=5
5	P=5, L=2	P=5, L=3	P=5, L=4	P=5, L=5
6	P=6, L=2	P=6, L=3	P=6, L=4	P=6, L=5
7	P=7, L=2	P=7, L=3	P=7, L=4	P=7, L=5
8	P=8, L=2	P=8, L=3	P=8, L=4	P=8, L=5
9	P=9, L=2	P=9, L=3	P=9, L=4	P=9, L=5
10	P=10, L=2	P=10, L=3	P=10, L=4	P=10, L=5
11	P=11, L=2	P=11, L=3		P=11, L=5
12	P=12, L=2	P=12, L=3		P=12, L=5
13	P=13, L=2	P=13, L=3		
14	P=14, L=2	P=14, L=3		
15	P=15, L=2	P=15, L=3		
16	P=16, L=2	P=16, L=3		
17	P=17, L=2	P=17, L=3		
18	P=18, L=2	P=18, L=3		
19	P=19, L=2	P=19, L=3		
20	P=20, L=2	P=20, L=3		
21	P=21, L=2	P=21, L=3		
22	P=22, L=2	P=22, L=3		
23	P=23, L=2	P=23, L=3		
24	P=24, L=2			
25	P=25, L=2			
26	P=26, L=2			
27	P=27, L=2			
28	P=28, L=2			
29	P=29, L=2			
30	P=30, L=2			
31	P=31, L=2			

THESE COMBINATIONS ARE NOT AVAILABLE – TRY RUNNING A SMALLER EXPERIMENT

P = 13; L=3 ;

$L_x(3^{13})$ -> $L_{27}(3^{13})$

x = 27

OBS:

Testando todas as possibilidades = 1.594.323 permutações!