SSC 0721 – Teste e Inspeção de Software

Critérios de fluxo de dados

Prof. Marcio E. Delamaro

delamaro@icmc.usp.br

Critérios Baseados em Fluxo de Dados

- Critérios pertencentes à Técnica de Teste Caixa Branca.
- Complementares aos critérios baseados em fluxo de controle.
- Busca testar o uso das variáveis em um programa, ou seja, como os dados são usados nas computações.

Motivação

- Teste de fluxo de dados é uma ferramenta poderosa para o uso incorreto de valores resultante de erros de codificação.
- Tornou-se popular com a publicação do trabalho de Rapps e Weyuker
  “"It is our belief that, just as one would not feel confident about a program without executing every statement in it as part of some test, one should not feel confident about a program without having seen the effect of using the value produced by each and every computation."

Critérios de fluxo de dados

- Usam informação sobre como as variáveis são utilizadas
- Cada vez que um valor é atribuído a uma variável, é necessário verificar se esse valor está correto
- Essa verificação é feita quando esse valor for acessado
- Tipos de utilização de variáveis

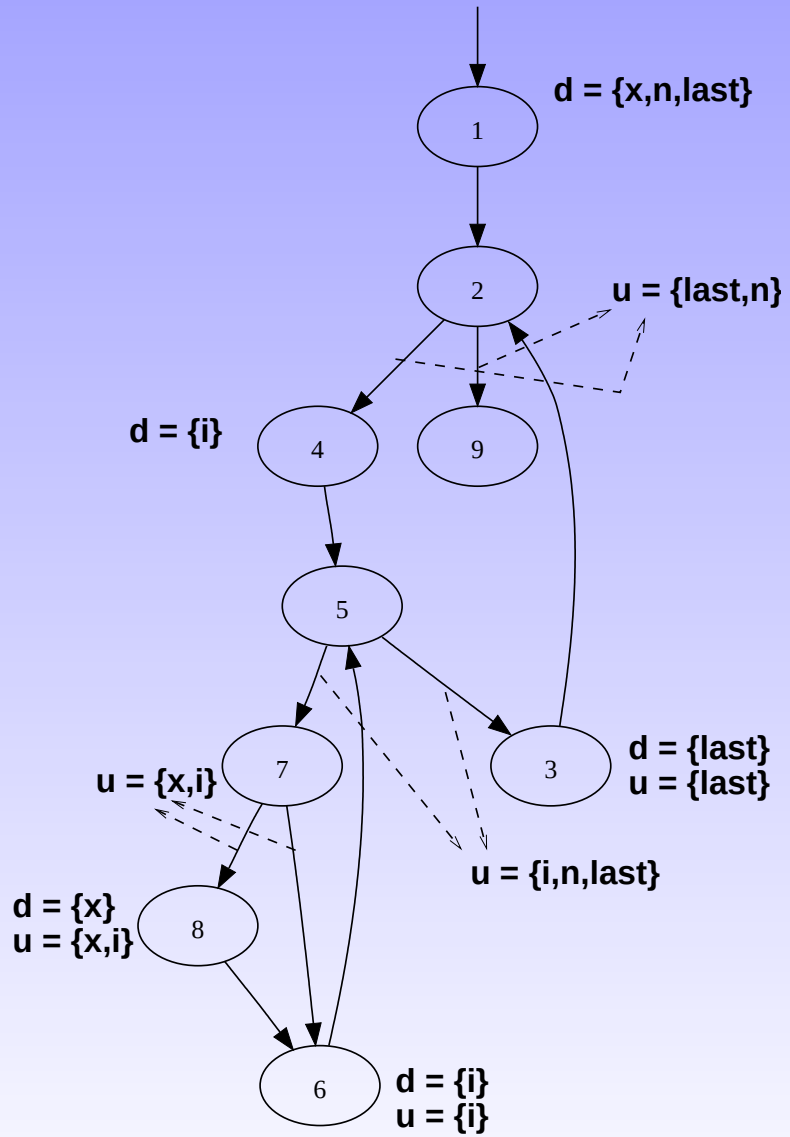
Critérios de fluxo de dados

- Tipos de utilização de variáveis
  - **Definição**: é toda referência feita a uma variável que faz com que o valor dessa variável possa ser alterado
    - a = 10;
    - read(a);
  - **Uso**: são todas as demais referências a variáveis, ou seja, quando o valor armazenado na variável é utilizado mas não modificado.
    - predicativo ou p-uso: `if (a > 10 )`
    - computacional ou c-uso: `a = b * c`

Grafo def-uso

- Estende o GFC
- Adiciona anotações de uso e definição de variáveis
- Cada c-uso é associado ao vértice do comando onde o uso ocorre
- Cada p-uso é associado às arestas que ele determina
- Uso e definição global

Grafo def-uso – exemplo



Modelo de fluxo de dados

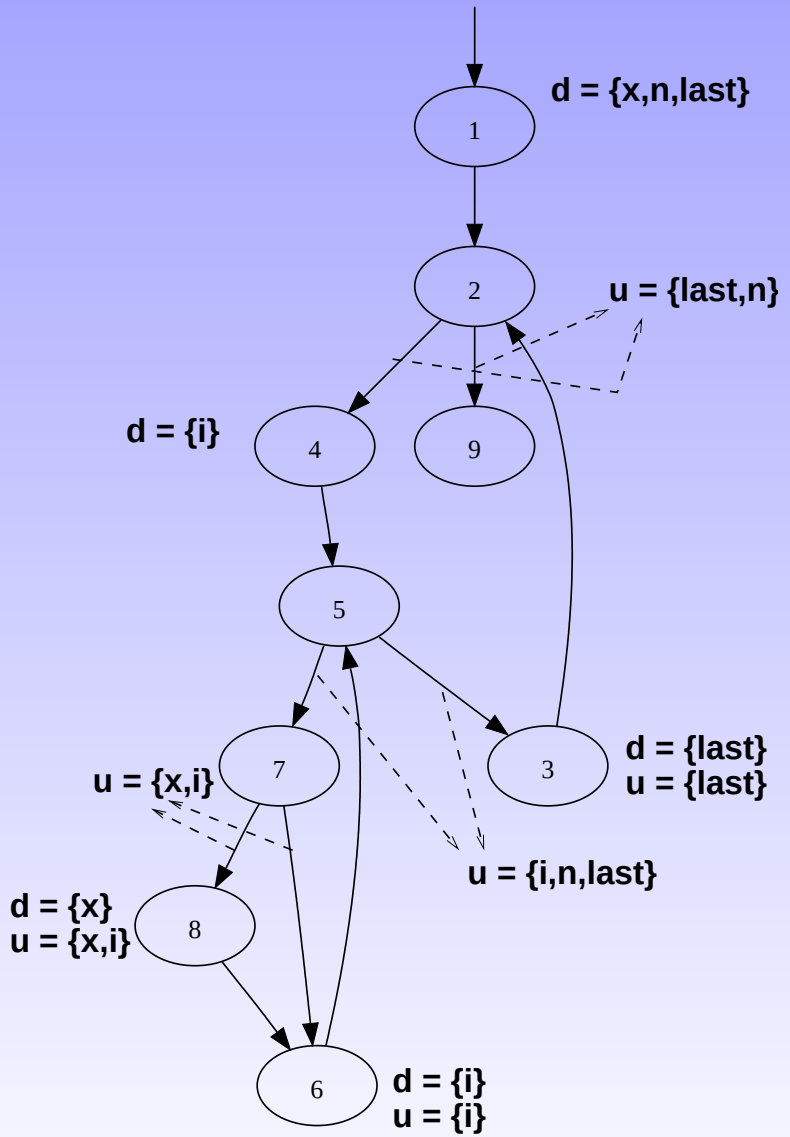
- Como se dão as definições e os usos
- Definição e uso de arrays. `v[i] = 10;`
- Utilização de ponteiros ou referências a objetos. `x->idade = 10`
- Chamada de subrotinas. Passagem de parâmetros por referência
- Uso de variáveis globais. Onde se dá a definição?

Conceitos

- Um **caminho livre de definição** em relação a uma variável `x` do vértice `i` ao vértice `j` ou à aresta (j, k) é uma caminho que não possui nenhuma definição de `x`, exceto, possivelvente, no vértice inicial, `i` ou no vértice `k`
- Uma **associação definição/uso de uma variável `x`** é uma tripla $\langle x, i, j \rangle$ ou $\langle x, i, (j, k) \rangle$ tal que `x` é definida no nó `i` e possui um uso no nó `j` ou na aresta `(j, k)`. Além disso deve existir um caminho livre de definição c.r.a. `x` dessa definição até esse uso.

Associação – exemplo

- ✓ $\langle i, 4, 8 \rangle$
- ✓ $\langle i, 4, (7, 6) \rangle$
- ✓ $\langle i, 4, (7, 8) \rangle$
- ✓ $\langle x, 8, 8 \rangle$
- ✓ $\langle x, 8, (7, 6) \rangle$
- ✓ $\langle x, 8, (7, 8) \rangle$



Critérios

- Critérios são baseados na execução das associações def-uso
- Critério **todas-definições** requer que para cada definição de variável, um uso seja exercitado
- Critério **todos-usos** requer que para cada definição de variável, todos os usos existentes sejam exercitados
- Outros: todos-p-usos-algum-c-uso, todos-c-usos-algum-p-uso, ...

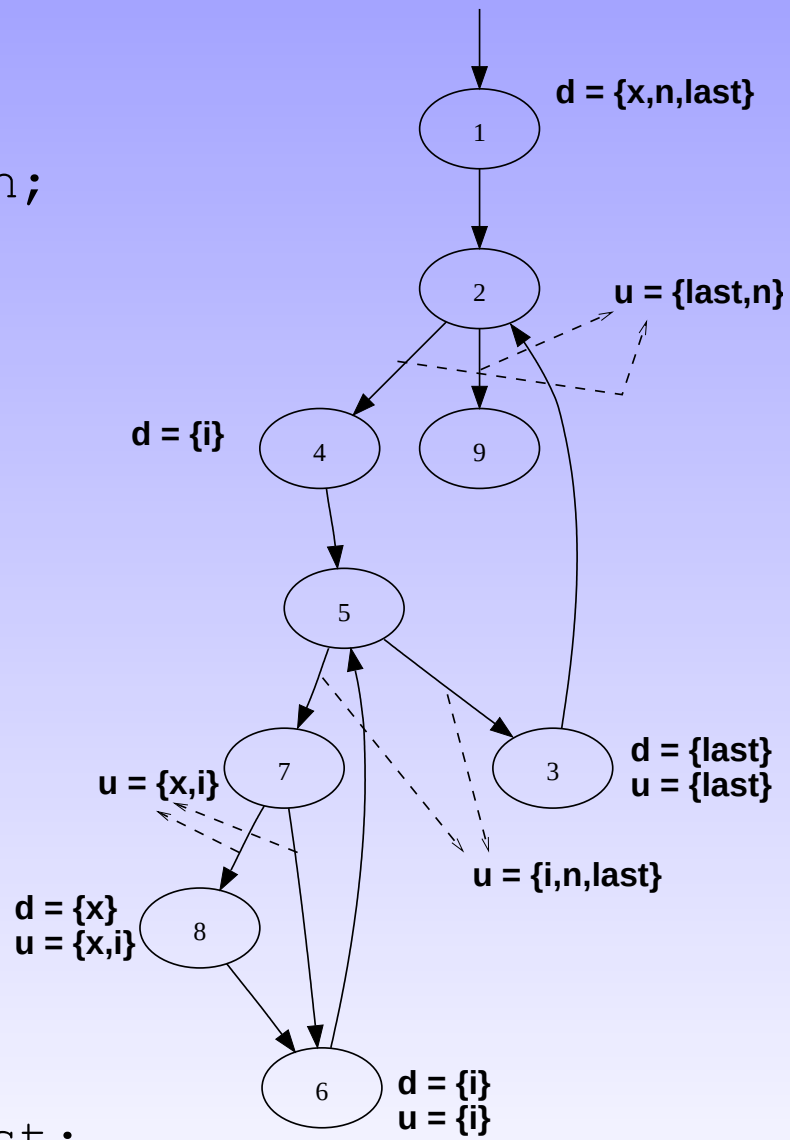
Critérios – exemplo

Variável	Definição	Uso
i	4	6
i	4	8
i*	4	(5,3)
i	4	(5,7)
i	4	(7,6)
i	4	(7,8)
i	6	8
i	6	(5,3)
i	6	(5,7)
i	6	(7,6)
i	6	(7,8)
last	3	(5,3)
last	3	(5,7)
last	3	(2,4)
last	3	(2,9)

Variável	Definição	Uso
last	1	3
last	1	(5,3)
last	1	(5,7)
last	1	(2,4)
last*	1	(2,9)
n	1	(5,7)
n	1	(5,3)
n	1	(2,4)
n	1	(2,9)
x	0	8
x	0	(7,6)
x	0	(7,8)
x	8	8
x*	8	(7,6)
x	8	(7,8)

Critérios – exemplo

- ✓ $\langle last, 1, (2, 9) \rangle$
  - ★ Não entrar no `for`
  - ★
    ```
    for (int last = 1; last < n;
    ```
  - ★ Array de tamanho 0 ou 1
- ✓ $\langle x, 8, (7, 6) \rangle$
  - ★ Trocar um elemento e não trocar o seguinte
  - ★ Def no nó 8 e passar direto na interação seguinte (p-uso na aresta)
  - ★ Entrada: [2, 1, 3]
- ✓ $\langle i, 4, (5, 3) \rangle$
  - ★ Não entrar no `for`
  - ★
    ```
    for (int i = 0; i < n - last;
    ```
  - ★ Não executável



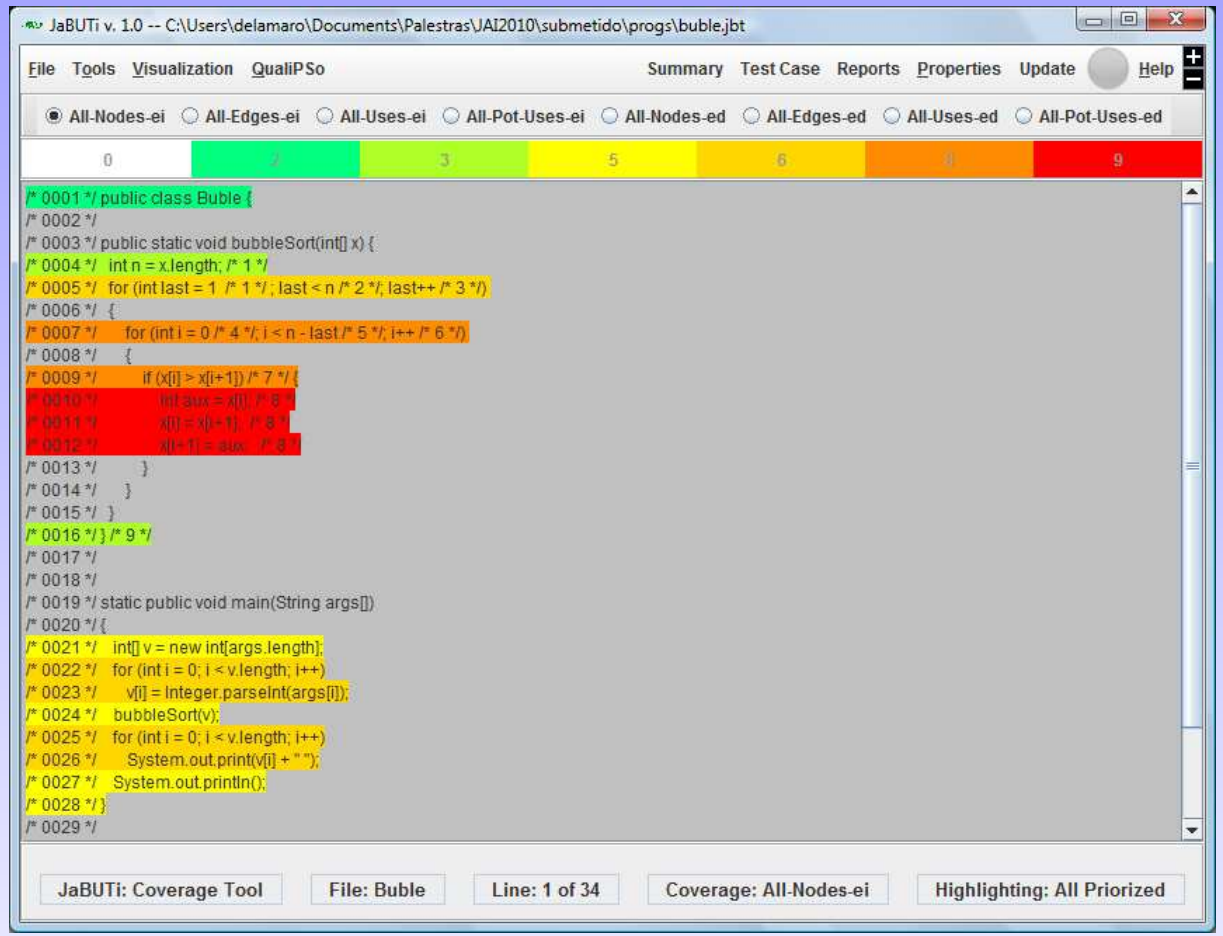
Observações

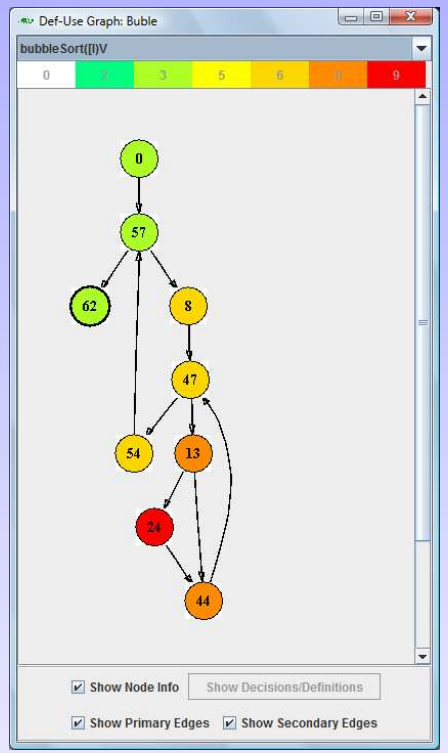
- Complementa os critérios de fluxo de controle
- Eficaz em revelar alguns tipos de defeitos, principalmente relacionados ao dados
- Na prática são critérios que não fizeram muito sucesso
  - Complexidade
  - Requisitos não executáveis
  - Ferramentas

Ferramenta – JaBUTi

- √ Java Bytecode Understanding and Testing
- √ Fluxo de controle e fluxo de dados
- √ Linguagem Java (bytecode da JVM)
- √ Critérios (in)dependente de exceções

JaBUTi

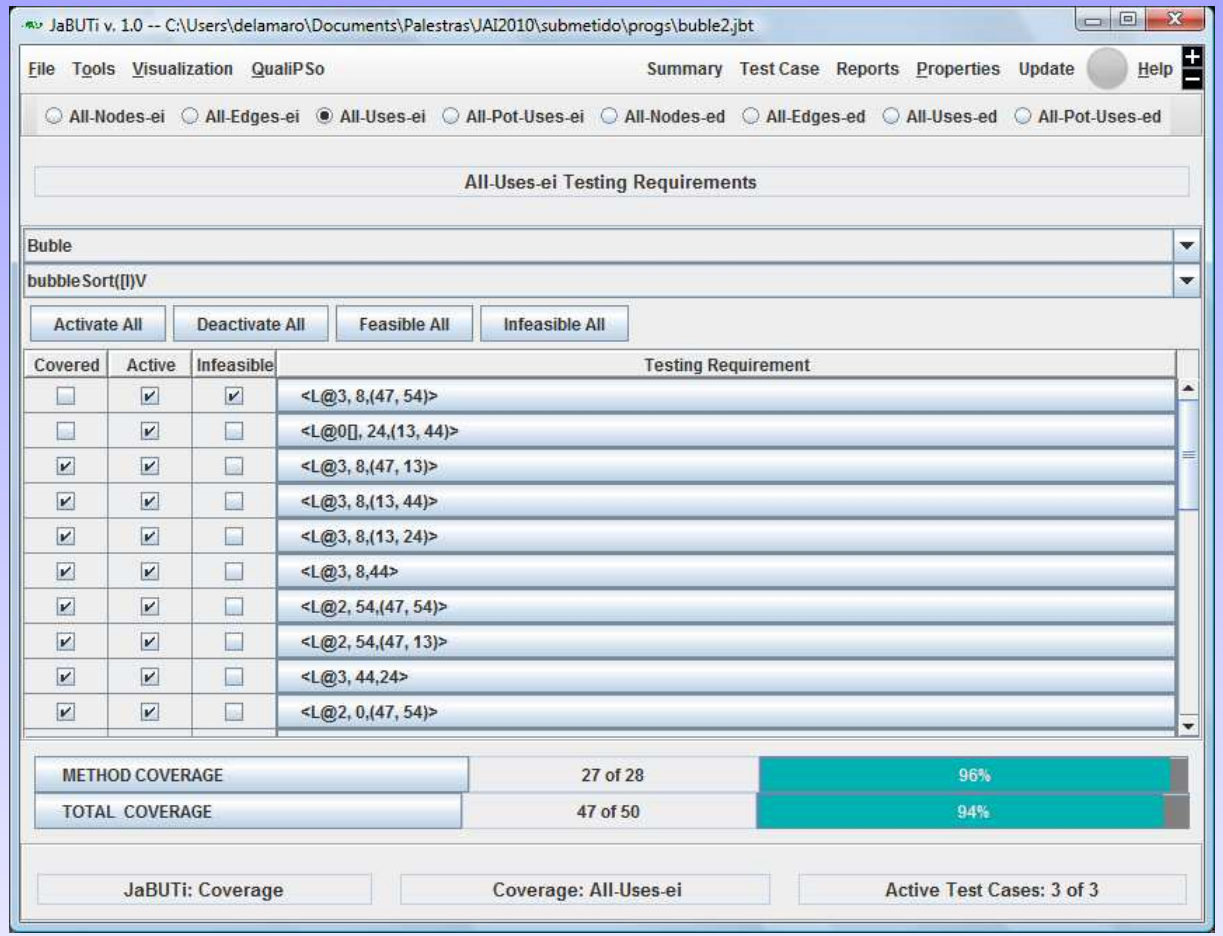

JaBUTi v. 1.0 -- C:\Users\delamaro\Documents\Palestras\JAI2010\submetido\progs\buble.jbt
File Tools Visualization QualiPSo Summary Test Case Reports Properties Update Help
All-Nodes-ei All-Edges-ei All-Uses-ei All-Pot-Uses-ei All-Nodes-ed All-Edges-ed All-Uses-ed All-Pot-Uses-ed
0 2 3 5 6 8 9
/* 0001 */ public class Buble {
/* 0002 */
/* 0003 */ public static void bubbleSort(int[] x) {
/* 0004 */ int n = x.length; /* 1 */
/* 0005 */ for (int last = 1 /* 1 */ ; last < n /* 2 */; last++ /* 3 */)
/* 0006 */ {
/* 0007 */ for (int i = 0 /* 4 */; i < n - last /* 5 */; i++ /* 6 */)
/* 0008 */ {
/* 0009 */ if (x[i] > x[i+1]) /* 7 */ {
/* 0010 */ int aux = x[i]; /* 8 */
/* 0011 */ x[i] = x[i+1]; /* 8 */
/* 0012 */ x[i+1] = aux; /* 8 */
/* 0013 */ }
/* 0014 */ }
/* 0015 */ }
/* 0016 */ } /* 9 */
/* 0017 */
/* 0018 */
/* 0019 */ static public void main(String args[])
/* 0020 */ {
/* 0021 */ int[] v = new int[args.length];
/* 0022 */ for (int i = 0; i < v.length; i++)
/* 0023 */ v[i] = Integer.parseInt(args[i]);
/* 0024 */ bubbleSort(v);
/* 0025 */ for (int i = 0; i < v.length; i++)
/* 0026 */ System.out.print(v[i] + " ");
/* 0027 */ System.out.println();
/* 0028 */ }
/* 0029 */
JaBUTi: Coverage Tool
File: Buble
Line: 1 of 34
Coverage: All-Nodes-ei
Highlighting: All Priorized


Def-Use Graph: Buble
bubbleSort([I)V
0 2 3 5 6 8 9
0
57
62
8
47
54
13
24
44
Show Node Info
Show Decisions/Definitions
Show Primary Edges
Show Secondary Edges

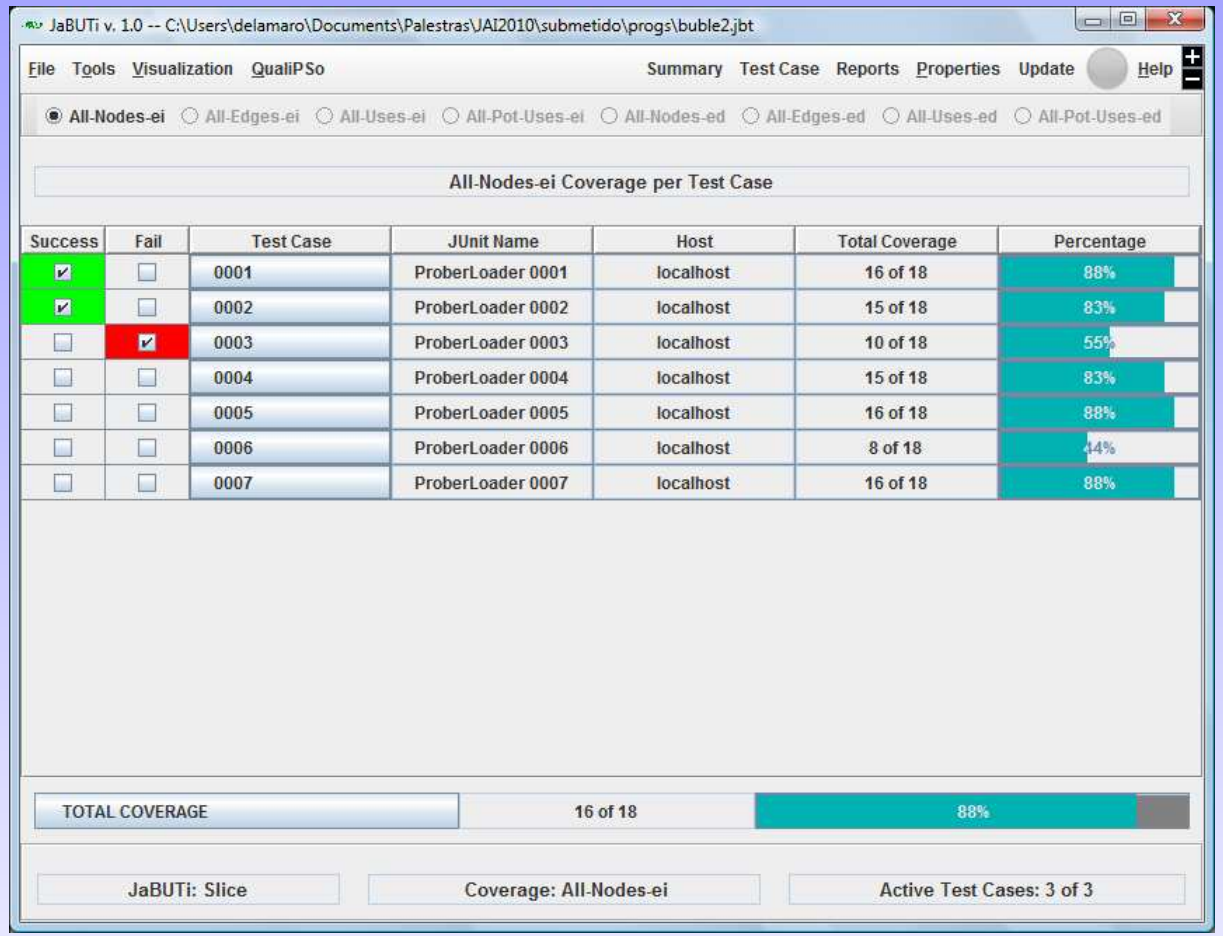
JaBUTi

- Execução manual ou integrada ao JUnit
- Priorização dos requisitos de teste
- Identificação de requisitos não executáveis
- Análise de cobertura por critério/classe/método
- Ferramenta de slicing
- Métricas estáticas OO
- `http://ccsl.icmc.usp.br`

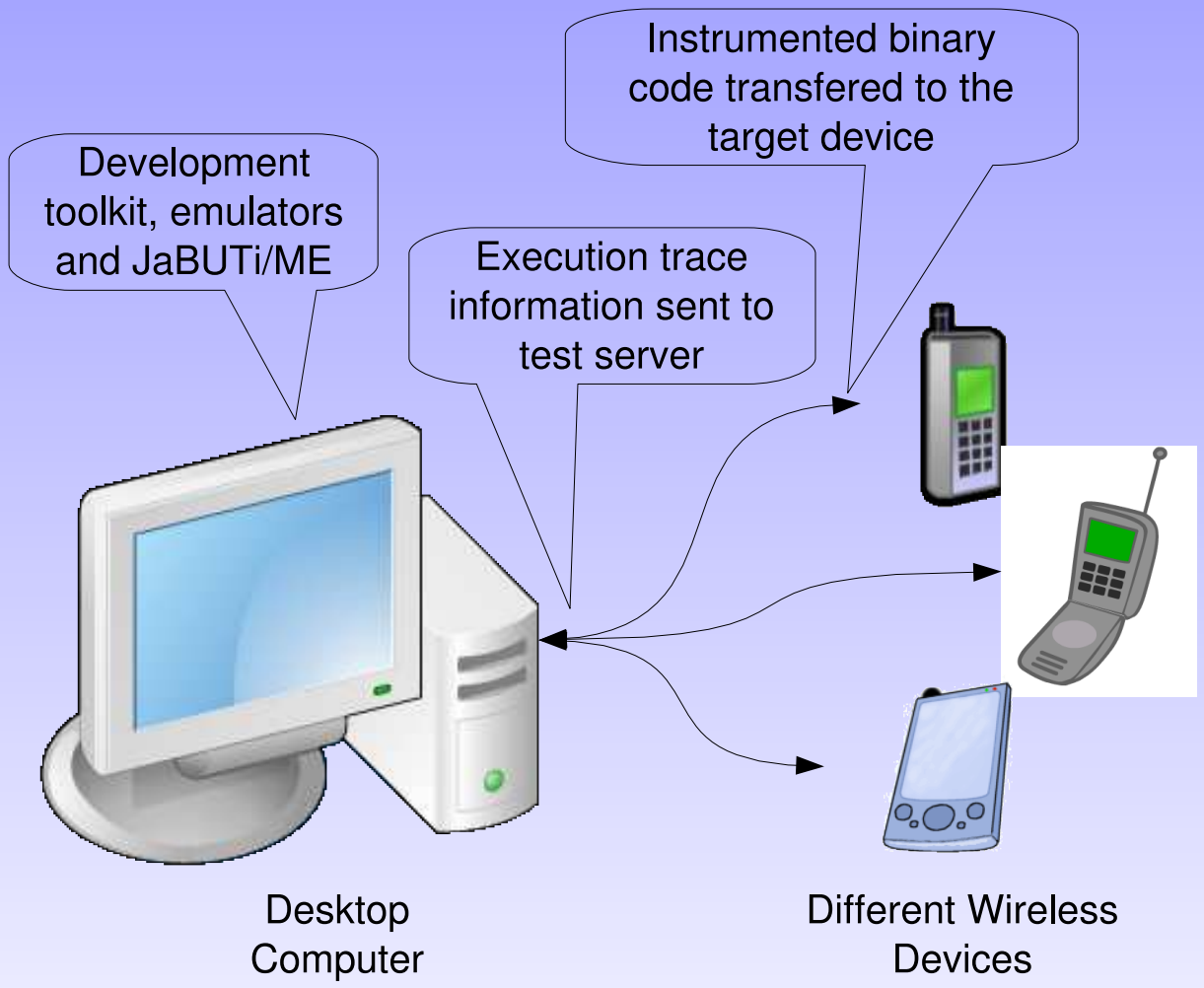
JaBUTi – análise de cobertura



JaBUTi – slicing



JaBUTi/ME


Instrumented binary code transfered to the target device
Development toolkit, emulators and JaBUTi/ME
Execution trace information sent to test server
Desktop Computer
Different Wireless Devices

Ferramenta – GCC

- Famoso compilador C
- Inclui opções que permitem gerar código instrumentado
- Execução gera arquivos com informação sobre cobertura de código (nós e desvios)

Ferramenta – gcov

```
        -:    0:Source:buble.c
        -:    0:Graph:buble.gcno
        -:    0:Data:buble.gcda
        -:    0:Runs:1
        -:    0:Programs:1
        -:    1:#include <stdio.h>
        -:    2:
        1:    3:void bubbleSort(int x[], int n) {
        3:    4:    for (int last = 1; last < n; last++) {
        5:    5:        for (int i = 0; i < n - last; i++) {
        3:    6:            if (x[i] > x[i+1]) {
    #####:    7:                int aux = x[i];
    #####:    8:                x[i] = x[i + 1];
    #####:    9:                x[i+1] = aux;
        -:   10:            }
        -:   11:        }
        -:   12:    }
        1:   13:}
```

Ferramenta – lcov