Laboratoire d'Informatique de Grenoble

Applying a pairwise coverage criterion to scenario-based testing

Lydie du Bousquet, Michael Delahaye, Catherine Oriat

Example: Bounded-Stack

```
public class BoundedStack {
    private int[] elems;
    private int numberOfElements;
    private int max;

    public  BoundedStack()  {…}

    public void push(int k) {…}
    public void pop() {…}
    public int top() {…}
   public boolean isEmpty() {…}
  }
```

Vocabulary

- A **test suite**
  - Set of test cases
  - Size: number of test cases
- A **test case**
  - Sequence of method calls
  - Size: number method calls
- Example:
  - T1: BoundedStack(); pop(); top();
  - T2: BoundedStack(); IsEmpty(); push(6);

Scenario-based testing

- To test the class,
  - Init the object
  - Apply different instantiated calls
- Scenario: $C; M^{3..3}$
  - C = { “int res; stack s = new stack(); int i = -1;” }
  - M = { “s.push(i++);”;
    “s.push(-1);”;
    “s.pop();”;
    “s.top();” }
- Complete unfolding => Test suite of 4^3 test cases

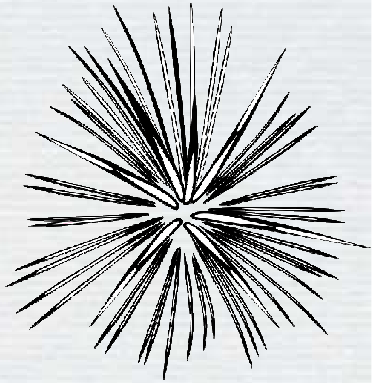
Executable test cases

```
public class Testsuite_BS1 extends TestCase {
  public void testSequence_1() {
    int res; stack s = new stack(); int i = -1;
    s.push(i++); s.push(i++); s.push(i++); s.push(i++); }
...
  public void testSequence_7() {
    int res; stack s = new stack(); int i = -1;
    s.push(-1); s.pop(); res = s.top(); s.push(i++); }
...
  public void testSequence_15() {
    int res; stack s = new stack(); int i = -1;
    res = s.top(); s.pop(); s.push(-1); res = s.top(); }
...
```

Oracle is not the subject of the article.
It can be implemented with assertions embedded in the code

Complete unfolding: combinatorial explosion

- [Arcuri] Size of the test cases is important to expose failure
- *$C; M^{3..3}$ -> $C; M^{10..10}$ (for instance)*
  - Combinatorial explosion!
  - So many test cases might not be relevant (execution cost)



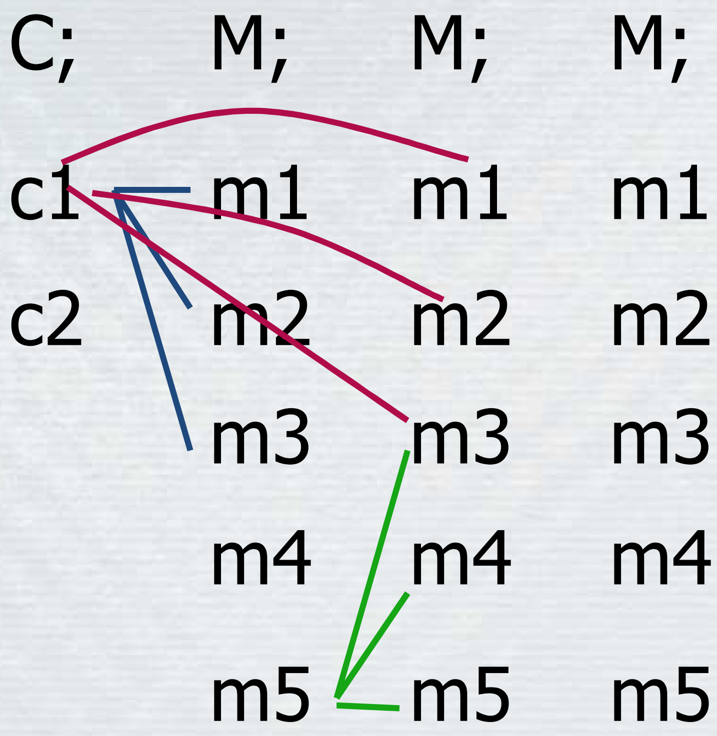
- Need to select a **subset** of test cases
- Different strategies for selection
  - <u>Randomly</u>: *But how many ?*
  - W.r.t some coverage criteria: why not <u>pairwise</u> ?
    - **Simple** to apply
    - A priori **relevant** in the sense that the order of calls has an importance

      push(1); pop(); different from pop(); push(1);

Pairwise coverage applied to method calls

C;	M;	M;	M;
c1	m1	m1	m1
c2	m2	m2	m2
	m3	m3	m3
	m4	m4	m4
	m5	m5	m5

Pairwise coverage applied to method calls



Is this coverage relevant?

- Experimentation
- Hypothesis: Random better than pairwise
- Subjects: 15 classes under tests
  - Containers and other types of classes with internal classes
- Test suites generated from scenarios: $C; M^{i..i}$
  - 252 test configurations = { SUT, C, M, i }
  - Pairwise selection with ACT => 100 test suites by configurations
  - Random selection => 100 test suites by configurations, same size

Test suite size

$\|C\|$	$\|M\|$	Number of method calls													
		3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	7	58	65	70	75	80	82	88	90	91	96	96	102	102	105
1	8	64	96	96	101	101	107	109	116	118	123	123	124	128	133
1	9	95	102	112	120	125	129	135	137	144	151	154	160	162	167
1	11	154	162	169	178	185	193	205	215	223	251	251	251	251	251
1	14	232	252	254	254	254	254	254	254	254	254	254	254	254	254
1	16	256	256	256	256	256	256	256	256	256	256	256	256	256	256
3	10	124	142	142	150	155	160	168	176	183	183	190	195	201	201

Fig. 6. Size of the test suites for each test configuration (i.e., number of test cases in a test suite)

Mutation analysis

- Mutant = Program under test + a single fault
  - Fault introduced w.r.t. mutation operator (e.g. + is transformed into -)
  - Mutant killed if Mutant and Original programs give different results
  - Mutation score: number of mutant killed by a test suite
- Trivial mutants are removed
  - Mutants killed by a test case composed of a single method call
  - Not relevant w.r.t. Pairwise hypothesis
- 1720 Non trivial mutants for the 15 classes under test
- Experimentation: comparing mutation score

Program Under Test					Test suite(s)	
Name	LOC[1]	# methods	# mutants	# non trivial mutants	$\lvert C\rvert$	$\lvert M\rvert$
ArrayStack [40]	100	8	85	54	1	7
AvlTree [40]	281	18	116	114	1	7
BankcardKernel [36]	538	13	536	424	1	16
BinTree [40]	124	5	148	110	1	9
BinarySearchTree [40]	219	15	201	166		
data set 1					1	16
data set 2					1	8
BinomialHeap [40]	434	6	97	73	1	8
BinomialQueue [40]	222	14	121	94	1	7
BoundedStack [29], [35]	75	10	204	166	1	9
Buffer [2]	44	4	206	156	1	7
Inventory [29], [35]	82	10	109	40		
data set 1					1	16
data set 2					1	11
Node [29], [35]	136	9	35	15	1	14
Queue [29], [35]	73	5	115	71	1	7
RedBlackTree [40]	254	16	129	71	1	8
VendingMachine [39], [29], [35]	85	6	113	104		
data set 1					1	7
data set 2					1	16
Another VendingMachine[2]	61	6	96	62	3	10
Total	2,728	145	2,311	1,720		

1. LOC is computed with *LOC Calculator* tool, http://code.google.com/p/loc-calculator/; white lines are not counted.
2. The second vending machine can be downloaded att http://en.literateprograms.org/Vending Machine (java).

Mutation score in average

	3	4	5	6	7	8	9	10	11	12	13	14	15	16
							ArrayStack							
R	15.12 (2.10)	17.35 (0.93)	17.91 (0.32)	18.12 (0.43)	18.00 (0.00)	18.01 (0.10)	18.05 (0.22)	18.04 (0.20)	18.02 (0.14)	18.04 (0.20)	18.05 (0.22)	18.00 (0.00)	18.05 (0.22)	18.45 (2.96)
P	16.09 (1.35)	17.77 (0.53)	17.95 (0.26)	18.00 (0.00)	18.00 (0.00)	18.00 (0.00)	18.06 (0.24)	18.03 (0.17)	18.27 (2.10)	18.22 (2.09)	18.46 (2.94)	18.26 (2.10)	18.26 (2.10)	18.68 (3.58)
							AVLTree							
R	27.93 (5.62)	35.90 (5.99)	41.48 (4.32)	43.40 (2.72)	44.78 (1.10)	45.40 (0.92)	45.66 (0.76)	45.82 (0.58)	45.94 (0.34)	45.96 (0.28)	46.34 (0.61)	46.00 (0.00)	46.00 (0.00)	46.00 (0.00)
P	28.61 (4.66)	36.79 (5.49)	42.36 (3.63)	44.40 (1.63)	44.87 (1.34)	45.31 (0.96)	45.74 (0.67)	45.90 (0.44)	45.94 (0.34)	46.00 (0.00)	45.98 (0.20)	45.98 (0.20)	46.00 (0.00)	46.00 (0.00)
							BankCardKernel							
R	245.29 (18.10)	268.42 (24.96)	288.91 (25.52)	306.76 (18.36)	316.12 (14.25)	324.86 (10.24)	329.68 (10.36)	332.31 (12.15)	336.71 (9.61)	340.74 (9.12)	344.37 (11.95)	346.43 (10.31)	351.63 (12.31)	353.83 (11.32)
P	247.61 (17.68)	275.20 (23.81)	295.34 (18.96)	308.76 (12.75)	316.69 (10.65)	323.79 (11.27)	328.31 (11.73)	333.20 (10.29)	337.40 (9.58)	341.01 (9.13)	345.79 (9.79)	348.89 (10.10)	351.76 (10.93)	354.92 (11.78)
							BinarySearchTree							
R	71.98 (8.23)	83.76 (4.79)	87.63 (3.33)	91.78 (4.09)	94.43 (4.37)	96.10 (4.62)	97.80 (4.83)	100.25 (4.70)	100.77 (4.84)	102.54 (4.37)	103.13 (4.39)	103.15 (4.01)	104.52 (3.49)	104.76 (3.41)
P	75.57 (6.78)	84.68 (3.24)	88.73 (3.36)	91.62 (3.49)	93.84 (4.32)	95.67 (4.20)	97.76 (4.69)	99.76 (4.88)	101.13 (4.63)	102.34 (4.52)	103.19 (4.30)	103.96 (4.06)	104.44 (3.72)	105.00 (3.23)
							BinarySearchTree8							
R	38.59 (14.03)	66.66 (10.72)	78.54 (6.69)	84.41 (4.34)	87.68 (4.75)	91.32 (3.49)	92.74 (3.18)	93.50 (1.89)	94.29 (2.32)	95.01 (2.82)	95.48 (2.40)	96.35 (3.29)	96.50 (3.24)	96.46 (3.18)
P	40.37 (11.99)	69.31 (8.97)	79.60 (6.07)	85.65 (4.51)	88.81 (3.86)	91.42 (2.99)	93.13 (2.46)	93.88 (2.39)	94.77 (2.29)	94.94 (1.98)	95.54 (2.65)	96.29 (3.12)	96.60 (3.41)	97.06 (3.35)
							BinomialHeap							
R	41.25 (6.76)	51.10 (3.44)	54.39 (2.12)	58.24 (3.22)	60.83 (2.80)	62.47 (1.49)	62.80 (0.70)	62.99 (0.44)	63.07 (0.36)	63.13 (0.37)	63.13 (0.37)	63.15 (0.36)	63.20 (0.40)	63.29 (0.46)
P	43.38 (6.12)	51.31 (3.02)	54.53 (2.18)	57.80 (3.17)	61.13 (2.83)	62.30 (1.87)	62.96 (0.56)	62.98 (0.53)	63.05 (0.22)	63.07 (0.35)	63.11 (0.31)	63.12 (0.33)	63.20 (0.40)	63.25 (0.43)
							BinomialQueue							
R	51.64 (5.91)	61.61 (6.07)	68.58 (4.49)	73.05 (2.89)	75.32 (1.69)	76.31 (1.56)	77.27 (0.98)	77.75 (0.98)	78.03 (1.06)	78.43 (0.87)	78.73 (0.81)	78.90 (0.64)	79.03 (0.52)	79.06 (0.58)
P	53.76 (4.51)	63.52 (5.66)	70.12 (3.82)	73.35 (2.57)	75.57 (1.51)	76.39 (1.25)	77.46 (1.04)	77.91 (0.91)	78.36 (0.74)	78.66 (0.62)	78.94 (0.58)	79.05 (0.52)	79.10 (0.44)	79.20 (0.43)
							BinTree							
R	76.67 (9.38)	92.22 (5.25)	98.86 (2.89)	101.56 (0.96)	102.09 (0.57)	102.31 (0.49)	101.45 (0.50)	101.65 (0.50)	101.78 (0.42)	101.97 (0.22)	101.95 (0.22)	101.97 (0.22)	101.99 (0.10)	102.00 (0.00)
P	76.85 (10.01)	93.16 (5.18)	99.40 (2.38)	101.64 (0.56)	102.09 (0.45)	102.41 (0.49)	102.60 (0.49)	101.78 (0.41)	101.84 (0.37)	101.92 (0.27)	101.97 (0.17)	101.99 (0.10)	102.00 (0.00)	102.00 (0.00)
							BoundedStack							
R	89.54 (8.04)	105.17 (5.08)	112.66 (3.70)	116.82 (5.12)	119.04 (4.76)	123.17 (3.96)	124.56 (3.11)	125.40 (2.25)	125.96 (0.83)	126.02 (0.14)	126.04 (0.20)	126.02 (0.14)	126.09 (0.32)	126.05 (0.22)
P	90.14 (5.30)	106.39 (4.45)	112.77 (3.52)	116.26 (4.06)	120.37 (4.67)	122.69 (4.31)	124.65 (3.13)	125.54 (1.79)	125.94 (0.91)	126.01 (0.10)	126.00 (0.00)	126.02 (0.14)	126.03 (0.17)	126.07 (0.26)
							Buffer							
R	131.42 (0.84)	132.00 (0.00)	132.00 (0.00)	132.03 (0.17)	132.00 (0.00)	132.03 (0.17)	132.07 (0.26)	132.01 (0.10)	132.05 (0.22)	132.02 (0.14)	132.03 (0.17)	132.03 (0.17)	132.00 (0.00)	132.02 (0.14)
P	131.72 (0.55)	132.00 (0.00)	132.00 (0.00)	132.04 (0.20)	132.03 (0.17)	132.00 (0.00)	132.05 (0.22)	132.02 (0.14)	132.00 (0.00)	132.03 (0.17)	132.03 (0.17)	132.04 (0.20)	132.06 (0.24)	132.03 (0.17)
							Inventory - Data Set 1							
R	100.13 (1.29)	100.91 (0.40)	101.00 (0.00)	101.03 (0.17)	101.00 (0.00)	101.00 (0.00)	101.00 (0.00)	101.04 (0.20)	101.00 (0.00)	101.00 (0.00)	101.00 (0.00)	101.00 (0.00)	101.00 (0.00)	101.00 (0.00)
P	101.00 (0.00)	101.00 (0.00)	101.00 (0.00)	101.00 (0.00)	101.00 (0.00)	101.00 (0.00)	101.01 (0.10)	101.00 (0.00)	101.04 (0.20)	101.01 (0.10)	101.00 (0.00)	101.01 (0.10)	101.00 (0.00)	101.00 (0.00)
							Inventory - Data Set 2							
R	86.21 (1.33)	87.99 (1.07)	89.15 (0.87)	89.82 (0.73)	90.47 (0.81)	91.10 (0.87)	91.75 (0.74)	92.28 (0.75)	92.57 (0.56)	92.84 (0.39)	92.83 (0.43)	92.95 (0.36)	92.96 (0.28)	93.02 (0.25)
P	86.51 (0.88)	88.16 (0.87)	89.10 (0.76)	89.91 (0.69)	90.50 (0.81)	91.17 (0.92)	91.84 (0.85)	92.31 (0.76)	92.53 (0.69)	92.83 (0.49)	92.94 (0.34)	92.99 (0.26)	93.04 (0.37)	93.07 (0.35)
							Node							
R	6.31 (0.88)	7.20 (0.84)	7.73 (0.51)	7.93 (0.41)	7.96 (0.20)	7.99 (0.10)	8.00 (0.00)	8.04 (0.20)	7.99 (0.10)	8.00 (0.00)	8.00 (0.00)	8.02 (0.14)	8.00 (0.00)	8.00 (0.00)
P	6.60 (0.81)	7.28 (0.78)	7.58 (0.62)	7.87 (0.34)	7.94 (0.24)	7.99 (0.10)	8.02 (0.14)	8.00 (0.00)	8.02 (0.14)	8.00 (0.00)	8.00 (0.00)	8.00 (0.00)	8.00 (0.00)	8.00 (0.00)
							Queue							
R	42.21 (3.98)	50.46 (5.79)	57.48 (4.68)	60.34 (1.83)	60.96 (0.40)	61.00 (0.00)	61.00 (0.00)	61.00 (0.00)	61.00 (0.00)	61.00 (0.00)	61.01 (0.10)	61.00 (0.00)	61.00 (0.00)	61.00 (0.00)
P	42.53 (3.84)	50.86 (5.44)	57.86 (4.18)	60.61 (1.11)	61.00 (0.00)	61.00 (0.00)	61.00 (0.00)	61.00 (0.00)	61.00 (0.00)	61.00 (0.00)	61.00 (0.00)	61.00 (0.00)	61.00 (0.00)	61.00 (0.00)
							RedBlackTree							
R	18.15 (0.82)	19.90 (0.70)	20.62 (0.60)	20.89 (0.31)	21.00 (0.00)	21.00 (0.00)	21.00 (0.00)	21.00 (0.00)	21.15 (0.36)	21.00 (0.00)	21.00 (0.00)	21.01 (0.10)	21.11 (0.35)	21.08 (0.34)
P	18.41 (0.74)	19.91 (0.71)	20.67 (0.47)	20.95 (0.22)	21.00 (0.00)	21.00 (0.00)	21.00 (0.00)	21.00 (0.00)	21.00 (0.00)	21.01 (0.10)	21.00 (0.00)	21.01 (0.10)	21.04 (0.20)	21.10 (0.30)
							VendingMachine 2							
R	40.35 (1.34)	40.99 (0.10)	41.00 (0.00)	41.00 (0.00)	41.00 (0.00)	41.06 (0.24)	41.00 (0.00)	41.00 (0.00)	41.00 (0.00)	41.00 (0.00)	41.00 (0.00)	41.00 (0.00)	41.00 (0.00)	41.00 (0.00)
P	40.67 (0.57)	40.98 (0.14)	41.00 (0.00)	41.00 (0.00)	41.00 (0.00)	41.00 (0.00)	41.01 (0.10)	41.00 (0.00)	41.00 (0.00)	41.01 (0.10)	41.00 (0.00)	41.00 (0.00)	41.00 (0.00)	41.00 (0.00)
							VendingMachine - Data Set 1							
R	76.97 (6.70)	84.19 (4.65)	86.56 (2.17)	86.89 (1.10)	87.00 (0.00)	87.00 (0.00)	87.05 (0.22)	87.11 (0.31)	87.14 (0.38)	87.37 (0.51)	87.51 (0.63)	87.66 (0.67)	87.85 (0.69)	88.02 (0.74)
P	78.18 (5.19)	84.56 (4.55)	86.89 (1.09)	87.00 (0.00)	87.02 (0.14)	87.04 (0.20)	87.10 (0.30)	87.09 (0.29)	87.25 (0.48)	87.32 (0.53)	87.56 (0.59)	87.83 (0.69)	87.99 (0.66)	88.12 (0.71)
							VendingMachine - Data Set 2							
R	74.59 (2.07)	77.73 (1.54)	79.78 (1.42)	81.17 (1.06)	81.77 (0.68)	82.15 (0.64)	82.33 (0.57)	82.35 (0.50)	82.49 (0.50)	82.61 (0.51)	82.83 (0.38)	82.93 (0.33)	82.94 (0.28)	82.95 (0.26)
P	75.79 (1.21)	78.28 (1.36)	80.47 (1.31)	81.62 (0.96)	81.94 (0.53)	82.20 (0.53)	82.35 (0.52)	82.44 (0.52)	82.58 (0.52)	82.71 (0.45)	82.75 (0.43)	82.85 (0.38)	83.07 (0.45)	82.99 (0.22)

Experiental results

- Contingency table
  - Pairwise test suites: PT
  - Random test suites: RT

	3	4	5	6	7	8	9	10	11	12	13	14	15	16	Total
$PT > RT$	18	16	13	11	11	7	13	9	11	9	8	9	9	11	155
$PT < RT$	0	1	2	6	3	8	3	6	5	7	6	7	4	1	59
$PT = RT$	0	1	3	1	4	3	2	3	2	2	4	2	5	6	38

Fig. 8. Contingency table of the average mutation score

- Wilcoxon signed-rank test
  - p-value of 8:22810
  - Hypothesis can be rejected with more than 95% confidence
  - (even with more than 99%)

Threats of validity

- Program under test (number and type)
- Choice of data
- Type of faults (mutation)

Conclusion & perspectives

- Pairwise coverage better than random selection
- Longer is better (see Arcuri)
- Size of pairwise test suite relevant

- New experiments with more complex scenarios