Unit Testing Tool Competition Round Four

Urko Rueda, René Just, Juan P. Galeotti, Tanja E. J. Vos

The 9th International Workshop on Search-Based Software Testing
1. About the Tool competition
2. The Tools
3. The Methodology
4. The Results
5. Lessons learned
## About the Tool competition

### Benchmarked Java unit testing at the class level

<table>
<thead>
<tr>
<th>Unit Testing Tool Competition</th>
<th>FITTEST <a href="http://crest.cs.ucl.ac.uk/fittest">crest.cs.ucl.ac.uk/fittest</a></th>
<th>Coverage metrics</th>
<th>Mutation metrics</th>
<th>CUTs / Projects / Tools</th>
<th>Tools SBST &amp; nonSBST</th>
</tr>
</thead>
<tbody>
<tr>
<td>2012 ICST’13</td>
<td>✓</td>
<td>Cobertura</td>
<td>Javalanche</td>
<td>77 / 5 / 2</td>
<td>Manual &amp; Randoop - baselines</td>
</tr>
<tr>
<td>2013 Round Two FITTEST’13</td>
<td>✓</td>
<td>JaCoCo</td>
<td>PI7est</td>
<td>63 / 9 / 4</td>
<td>1st + T3 &amp; Evosuite</td>
</tr>
<tr>
<td>2014 Round Three SBST’15</td>
<td>✗</td>
<td></td>
<td></td>
<td>63 / 9 / 8</td>
<td>2nd + Commercial &amp; GRT &amp; jTexPert &amp; Mosa(Evosuite)</td>
</tr>
<tr>
<td>2015 Round Four SBST’16</td>
<td>✗</td>
<td>Defects4J: github.com/rjust/defects4j + Real fault finding metric</td>
<td></td>
<td>68 / 5 / 4</td>
<td>Randoop - baseline &amp; T3 &amp; Evosuite &amp; jTexPert</td>
</tr>
</tbody>
</table>

---

4th Java unit testing competition 2
About the Tool competition

- Why?
  - Towards testing field maturity – this is just Java ...
  - Tools improvements, future developments insight

- What is new in the 4\textsuperscript{th} edition?
  - Benchmark infrastructure – split into
    - Test generation
    - Test execution & Test assessment (Defects4J)
  - Benchmark subjects (from Defects4J dataset)
  - Time budgets (1, 2, 4 & 8 minutes)
  - Flaky tests (non compliable, non reliable pass)
- SBST and non-SBST tools
- Command line tools
- Fully automated – no human intervention

<table>
<thead>
<tr>
<th>Tool</th>
<th>Technique</th>
<th>Static analysis</th>
<th>Edition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Randoop (baseline)</td>
<td>Random</td>
<td>✗</td>
<td>✓ ✓ ✓ ✓ ✓</td>
</tr>
<tr>
<td>T3</td>
<td></td>
<td>✗</td>
<td>✓ ✓ ✓ ✓</td>
</tr>
<tr>
<td>jTexPert</td>
<td>Random (guided)</td>
<td>✓</td>
<td>✗ ✗ ✓ ✓ ✓</td>
</tr>
<tr>
<td>Evosuite</td>
<td>Evolutionary algorithm</td>
<td>✓</td>
<td>✓ ✓ ✓ ✓</td>
</tr>
</tbody>
</table>
The Methodology

- Tool deployment
  - Installation – Linux environment
  - Wrapper implementation – runtool script
    - Std. IN/OUT communication protocol
    - 4th edition has a time budget
  - Tune-up cycle – setup, run, resolve issues
    - Benchmark infrastructure
      - Defects4J integration
      - Decoupling test generation from test execution/assessment
    - Tool – run over non contest benchmark samples
The Methodology

**Benchmark framework**

- “BENCHMARK”
- Src Path / Bin Path / ClassPath
- ClassPath for JUnit Compilation
  - preparation
  - “READY”

**Run tool for Tool T**

- “READY”
- Time-budget
- Name of CUT
- Generate file in ./temp/testcases
  - “READY”

**Loop**

- Compile + execute + measure test case

**Std. IN/OUT protocol**

4th Java Unit Testing Competition
The Methodology

- Benchmark infrastructure
  - Two HP Z820 workstations – each:
    - 2 CPU sockets for a total of 20 cores
    - 256Gb RAM
  - 32 virtual machines (16 per workstation)
    - Test generation
      - 1 core – control tool multi-threading capability
      - 8GB RAM
    - Test execution/assessment (tool independent)
      - 2 cores
      - 16Gb RAM – resolves out of memory issues
The Methodology

80 CUTs

benchmark tool replicated x32 VMs

T3
runtool

jTexpert
runtool

EvoSuite
runtool

Randoop
runtool

HP Z820 16 VMs
20core CPU
256Gb RAM

HP Z820 16 VMs
20core CPU
256Gb RAM

1core CPU
8Gb
RAM
2core CPU
16Gb RAM

1core CPU
8Gb
RAM
2core CPU
16Gb RAM

time budgets
time budgets

1 2 4 8m
time budgets
time budgets

1 2 4 8m
1 2 4 8m

1 2 4 8m
1 2 4 8m

1 2 4 8m
1 2 4 8m

1 2 4 8m
1 2 4 8m

RUNs 1, 2, 3
RUNs 4, 5, 6

generate test cases
generate test cases

collect metrics
collect metrics

aggregator

Calculate Score

4th Java unit testing competition
The Methodology

- Randoop
- T3
- EvoSuite
- jTexpert

Benchmark tool

- Test classes
  - @Test
  - @Test
  - @Test

Generate

- Time-budget (1, 2, 4, 8 min)
- CUT (fixed)

Run tool

- Compilable
  - Y
  - N

- Test classes
  - @Test
  - @Test
  - No flaky tests

- CUT (fixed)
  - 1 real fault

Run to detect and remove flaky tests

- Test classes
  - @Test
  - @Test

Defects4J

- CUT (mutated)

Run to collect metrics

Calculate score

4th Java unit testing competition
The Methodology

- Flaky tests
  - Passes during generation
  - But, might Fail during execution/assessment
  - False-positive warnings
    - Non reliable fault-detection
    - Non reliable mutation analysis
- Defects4J flaky tests sanity
  - Non compiling test classes
  - Failing tests over 5 executions (fixed CUT versions)
The Methodology

The Metrics – Test effectiveness

- Code coverage (fixed benchmark versions)
  - Defects4J <- Cobertura
  - Statement coverage
  - Condition coverage

- Mutation score
  - Defects4J <- Major framework (all mutation operators)

- Real fault detection (buggy benchmark versions)
  - 1 real fault per benchmark
  - 0 or 1 score, independent of how many tests reveal it
■ The Scoring formula

\[ \text{covScore}_{(T,L,C,r)} := w_i \cdot \text{cov}_i + w_b \cdot \text{cov}_b + w_m \cdot \text{cov}_m + \]

(real fault found? \( w_f : 0 \))

\( T = \text{Tool}; \ L = \text{Time budget}; \ C = \text{CUT}; \ r = \text{RUN (1..6)} \)

Coverages: \( \text{cov}_i = \text{statement}; \ \text{cov}_b = \text{condition} \)

\( \text{cov}_m = \text{mutants kill ratio} \)

Weights: \( w_i = 1; \ w_b = 2; \ w_m = 4; \ w_f = 4 \)
The Methodology

- The Scoring formula – time penalty

\[ tScore_{\langle T, L, C, r \rangle} := covScore_{\langle T, L, C, r \rangle} \cdot \min \left( 1, \frac{L}{\text{genTime}} \right) \]

- Test generation slot: \( L \leq 2 \cdot L \)
- No penalty if \( \text{genTime} \leq L \)
- Penalty for Extra time taken (\( \text{genTime} – L \))
  - Half \( \text{covScore} \) if the Tool must be killed (> 2 \( \cdot L \))
The Scoring formula – tests penalty

\[
\text{penalty}(T,L,C,r) := \begin{cases} 
2 & \text{if no compilable test classes} \\
\frac{\#u\text{Classes}}{\#\text{Classes}} + \frac{\#f\text{Tests}}{\#\text{Tests}} & \text{otherwise}
\end{cases}
\]

\#\text{Classes} = \text{generated test classes}; \ #u\text{Classes} = \text{uncompilable test classes;}
\#\text{Tests} = \text{test cases; } \#f\text{Tests} = \text{flaky test cases}.
The Scoring formula – Tool score

\[ \text{Score}(T,L,C,r) := \text{tScore}(T,L,C,r) - \text{penalty}(T,L,C,r) \]

\[ \text{Score}(T,L,C) := \text{avg}(\text{Score}(T,L,C,r) \text{ for all } r \text{ executions}) \]

\[ \text{score}_T := \sum_{L,C} \text{Score}_{\langle T,L,C \rangle} \]
The Methodology

- Conclusion validity
  - Reliability of treatment implementation
    - Tool deployment instructions EQUAL for all participants
  - Reliability of measures
    - Efficiency: wall clock time by Java `System.currentTimeMillis()`
    - Effectiveness: Defects4J
    - Tools non-deterministic nature: 6 runs (HW Capacity)
The Methodology

- **Internal validity**
  - CUTs from Defects4J (uniform and arbitrary selection from 5 open source projects)
    - Tools and benchmark infrastructure Tune-up samples
    - Contest benchmarks
  - Wrappers *runtool*: implemented by Tools side

- **Construct validity**
  - Scoring formula weights – quality indicators value
    - Empirical studies – correlation of proxy metrics for:
      Test effectiveness and Fault finding capability
### The Results

#### Table 3: Overall scores for all tools.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Budget</th>
<th>Score</th>
<th>Std.dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVO SUITE</td>
<td>*</td>
<td>1127</td>
<td>136.96</td>
</tr>
<tr>
<td>T3</td>
<td>*</td>
<td>978</td>
<td>86.17</td>
</tr>
<tr>
<td>JTEXPERT</td>
<td>*</td>
<td>931</td>
<td>137.03</td>
</tr>
<tr>
<td>RANDOOP</td>
<td>*</td>
<td>747</td>
<td>40.31</td>
</tr>
</tbody>
</table>

#### Table 4: Scores for all time budgets.

(OPTIMAL gives the maximum score and DEVELOPER the score achieved by the developer-written test suites).

<table>
<thead>
<tr>
<th>Tool</th>
<th>Budget</th>
<th>Score</th>
<th>Std.dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>T3</td>
<td>1min</td>
<td>220</td>
<td>27.836</td>
</tr>
<tr>
<td>EVO SUITE</td>
<td>1min</td>
<td>209</td>
<td>33.564</td>
</tr>
<tr>
<td>JTEXPERT</td>
<td>1min</td>
<td>179</td>
<td>38.811</td>
</tr>
<tr>
<td>RANDOOP</td>
<td>1min</td>
<td>155</td>
<td>11.500</td>
</tr>
<tr>
<td>EVO SUITE</td>
<td>2min</td>
<td>259</td>
<td>45.679</td>
</tr>
<tr>
<td>T3</td>
<td>2min</td>
<td>241</td>
<td>30.649</td>
</tr>
<tr>
<td>JTEXPERT</td>
<td>2min</td>
<td>231</td>
<td>41.199</td>
</tr>
<tr>
<td>RANDOOP</td>
<td>2min</td>
<td>179</td>
<td>12.553</td>
</tr>
<tr>
<td>EVO SUITE</td>
<td>4min</td>
<td>318</td>
<td>58.304</td>
</tr>
<tr>
<td>T3</td>
<td>4min</td>
<td>253</td>
<td>27.687</td>
</tr>
<tr>
<td>JTEXPERT</td>
<td>4min</td>
<td>251</td>
<td>48.000</td>
</tr>
<tr>
<td>RANDOOP</td>
<td>4min</td>
<td>197</td>
<td>16.254</td>
</tr>
<tr>
<td>EVO SUITE</td>
<td>8min</td>
<td>341</td>
<td>57.720</td>
</tr>
<tr>
<td>JTEXPERT</td>
<td>8min</td>
<td>270</td>
<td>47.830</td>
</tr>
<tr>
<td>T3</td>
<td>8min</td>
<td>263</td>
<td>27.687</td>
</tr>
<tr>
<td>RANDOOP</td>
<td>8min</td>
<td>216</td>
<td>20.708</td>
</tr>
<tr>
<td>OPTIMAL</td>
<td>–</td>
<td>748</td>
<td>–</td>
</tr>
<tr>
<td>DEVELOPER</td>
<td>–</td>
<td>611</td>
<td>–</td>
</tr>
</tbody>
</table>

4th Java unit testing competition 18

A single virtual machine would use 8 CPU months!

---

Contest run for ~1 week
Test generation, execution and assessment
x32 VMs

No time budgets
Lessons learned

- Testing Tools improvements
  - Automation, Test effectiveness, Comparability
- Benchmarking infrastructure improvements
  - Decoupling Test gen. from execution/assessment
  - Flaky tests identification and sanity
  - Fault finding capability measurement
  - Test effectiveness due to Test generation time
- What next?
  - Automated parallelization of the benchmark contest
  - More Tools, new languages? (i.e. C#?)
Universidad Politécnica de Valencia, ES
urueda@pros.upv.es, tvos@dsic.upv.es

Open Universiteit Heerlen, NL
tanja.vos@ou.nl

University of Massachusetts Amherst, MA, USA
rjust@cs.umass.edu

University of Buenos Aires, Argentina
jgaleotti@dc.uba.ar

web:  http://sbstcontest.dsic.upv.es/