Coverage-Based Reduction of Test Execution Time: Lessons from a Very Large Industrial Project
Content

• Academic-industry collaboration details
• Test environment
• Challenges and gaps between research and practice
• Our results from coverage analysis
Collaboration Details

• Started in 2012
• Recurring student activities (> 10 theses, internships)
• PhD project: Testing in Very Large Software Projects
  – PhD student at Heidelberg University and SAP
• Success factors:
  – Good combination: Practical relevant & nontrivial research
  – Real, large scale software product as a use case
• Challenges:
  – Transfer research to production
  – Find interested persons in charge
Test Environment

• SAP HANA
  – In-memory database management system
  – Core product platform of SAP
  – Several million LOC C/C++, scales up to >600 cores

• Testing
  – More than 1000 test suites with more than 100 000 tests
  – Coverage is line based per test suite
  – Test framework in python
    • Test sends SQL to HANA and checks results
GAPS BETWEEN RESEARCH AND PRACTICE
Project goals and discovered gaps

• We want to
  – Reduce test runtime
  – Increase specificity of coverage based test characterization

• We encountered several issues with existing work
## Evaluation with Small Projects

- Practitioners do not trust small evaluations

<table>
<thead>
<tr>
<th>Work</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alspaugh et al. 2007</td>
<td>5 classes to 22 classes</td>
</tr>
<tr>
<td>Zhang et al. 2009</td>
<td>53 testcases to 209 testcases</td>
</tr>
<tr>
<td>Li et al. 2009</td>
<td>374 LOC to 11 kLOC</td>
</tr>
<tr>
<td>You et al. 2011</td>
<td>500 LOC to 10 kLOC</td>
</tr>
<tr>
<td>Zhang et al. 2013</td>
<td>2 kLOC to 80 kLOC</td>
</tr>
<tr>
<td>Do et al. 2008</td>
<td>7 kLOC to 80 kLOC</td>
</tr>
<tr>
<td>Elbaum et al. 2002</td>
<td>8 kLOC to 300 kLOC</td>
</tr>
<tr>
<td><strong>Our work</strong></td>
<td><strong>&gt; 3.50 MLOC</strong></td>
</tr>
</tbody>
</table>

Related work comparing overlap-aware vs. non-overlap-aware solvers for TCS or TCP

1 See paper for details
Flaky Tests

- Execute test 1: OK
- Execute test 1: OK
- Execute test 1: OK
- Execute test 1: Failed
- Execute test 1: OK

Investigate?

- Test infrastructure?
- Hardware Problems?
- Memory leak?
- Test dependencies?
- Real bug? (e.g. concurrency)
- Performance?
- and more ...

Ignore?
Flaky Tests

- Execute test 1: OK
- Execute test 1: OK
- Execute test 1: OK
- Execute test 1: Failed
- Execute test 1: OK

Real world is not perfect and return of investment avoids perfection

Flaky test detection and handling is time consuming

Test infrastructure?
Hardware Problems?
Memory leak?
Test dependencies?
Real bug? (e.g. concurrency)
Performance?
and more …
Shared coverage

Database Code

Covered by nearly all tests

Large part of coverage is not specific
Random Coverage

- Coverage A: 651 074 lines hit
- Coverage B: 651 845 lines hit
- Coverage C: 651 862 lines hit
- Coverage D: 652 015 lines hit
Random Coverage

- Coverage A: 651,074 lines hit
- Coverage B: 651,845 lines hit
- Coverage C: 651,862 lines hit
- Coverage D: 652,015 lines hit

In Fact:
A and B from same Test1
C and D from same Test2
Test2 contains Test1 + more

Impossible to find exactly identical or included tests
Size of Coverage Data

Size is nontrivial and increasing
OUR RESULTS ON COVERAGE ANALYSIS
Overlap-Aware Coverage Algorithms

• Test Case Selection
  – Time budget 1h: Which tests to run?
    • Objective: coverage – Maximum budgeted cov. problem
  – Which tests to run for full coverage?
    • Objective: cardinality – Set cover problem
    • Objective: runtime – Weighted set cover problem

• Test Case Prioritization
  – Which tests to run first? Objective: coverage (per time)

Unsafe algorithms, we could miss functionality
Overlap-Aware Coverage Algorithms

• **Test Case Selection**
  - Time budget 1h: Which tests to run?
    - Objective: coverage – Maximum budgeted cov. problem
  - Which tests to run for full coverage?
    - Objective: cardinality – Set cover problem
    - Objective: runtime – Weighted set cover problem

• **Test Case Prioritization**
  - Which tests to run first? Objective: coverage (per time)

Unsafe algorithms, we could miss functionality
Overlap-Aware vs. Simple Greedy

Coverage

Test 1
Test 2
Test 3
Overlap-Aware vs. Simple Greedy

Coverage
- Test 1
- Test 2
- Test 3

Simple greedy
- Test 1
- Test 2
- Test 3
Overlap-Aware vs. Simple Greedy

- **Coverage**
  - Simple greedy: Test 1, Test 2, Test 3
  - Overlap-aware greedy: Test 1, Test 2 (omitted), Test 3

- **Simple greedy**
  - Test 1
  - Test 2
  - Test 3

- **Overlap-aware greedy**
  - Test 1
  - Test 2 (omitted)
  - Test 3
Comparison Overlap-Aware

Figure 2. Exemplary comparison between different algorithms for maximum budgeted coverage problem. Higher is better.

- **Overlap-aware greedy** reaches more coverage faster

- Runtime for single run: <10s
- Also works for test clusters with buckets
Parallel Variant for Test Clusters

Budget: 1 x 3 hours

Test Server A

- Test 1
- Test 2
- Test 3
- Test 4

Test Server 2

- Test 5
- Test 6
- Test 7

Test Server 3

- Test 1
- Test 2
- Test 3
- Test 4
Parallel Variant for Test Clusters

Test Server A

Budget: 1 x 3 hours

Test 1 → Test 2 → Test 3 → Test 4

Server 1
Budget: 1 hour

Test 1 → Test 2 → Test 3

Server 2
Budget: 1 hour

Test 1 → Test 2

Server 3
Budget: 1 hour
Overlap-Aware for Test Clusters

Overlap-Aware Greedy for Test Clusters with 1, 4, 8, 16 or 32 Servers

Coverage decrease < 0,01% -> works for test clusters
Coverage Redundancy

```c
int example_function(int a, int b) {
    int c = a + b;
    int d = a - b;
    return c*d;
}
```
Coverage Redundancy

```c
int example_function(int a, int b) {
    int c = a + b;
    int d = a - b;
    return c*d;
}
```

<table>
<thead>
<tr>
<th></th>
<th>Test1</th>
<th>Test2</th>
<th>Test3</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>S2</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>S3</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>S4</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>S5</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>
Coverage Redundancy

```c
int example_function(int a, int b) {
    int c = a + b;
    int d = a - b;
    return c*d;
}
```

<table>
<thead>
<tr>
<th></th>
<th>Test1</th>
<th>Test2</th>
<th>Test3</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>S2</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>S3</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>S4</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>S5</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>

---
Coverage Redundancy

Large part of coverage data is redundant
Shared Coverage Problem

- Ask SAP engineers where they expect coverage for Test1
Shared Coverage Problem

- Ask SAP engineers where they expect coverage for Test1

- Measure Test1

Coverage does not characterize Test1
Filtering Shared Coverage Data

Considered two approaches:

a) **Baseline approach**
   Define baseline test and remove baseline coverage from all other tests

b) **Testcount approach**
   Remove all lines covered by more than e.g. 238 tests (of e.g. 1200 in total)
Distribution plot. E.g. 80% of all lines hit are covered by only 238 or less test suites and 31% of all lines are covered by only 1 test
Filtering Shared Coverage Evaluation

Measurement

Coverage for Test1

After Approach

Filtered Coverage for Test1
Filtering Shared Coverage Evaluation

- List of top 5 directories ordered by lines hit:
  F, C, B, D, A
  D, F, A, B, C
Filtering Shared Coverage Evaluation

• List of top 5 directories ordered by lines hit:
  
  F, C, B, D, A

  D, F, A, B, C

• Ask SAP engineers if this fits their expectations:

  No

  Yes
Filtering Shared Coverage Evaluation

<table>
<thead>
<tr>
<th>Original Coverage</th>
<th>Correct=x</th>
<th>Incorrect</th>
<th>Unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test1</td>
<td>x</td>
<td>x</td>
<td>unknown</td>
</tr>
<tr>
<td>...</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test10</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3/10</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Filtering Shared Coverage Evaluation

Specificity improved significantly
**Summary**

### Gaps
- **Flaky Tests**
  - Execute test 1: OK
  - Execute test 1: OK
  - Execute test 1: Failed
  - Execute test 1: OK

### Random Coverage
- Coverage A: 651,074 lines hit
- Coverage B: 651,845 lines hit
- Coverage C: 651,862 lines hit
- Coverage D: 652,015 lines hit

### Evaluation with Small Projects
- Practitioners do not trust small evaluations

### Coverage Redundancy

```c
int example_function(int a, int b) {
    int c = a + b;
    int d = a - b;
    return c * d;
}
```

<table>
<thead>
<tr>
<th>Coverage run</th>
<th>Lines hit</th>
<th>Lines groups</th>
<th>Redundancy</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015-11-15</td>
<td>2901575</td>
<td>79741</td>
<td>97.25</td>
</tr>
<tr>
<td>2016-05-19</td>
<td>3172337</td>
<td>93162</td>
<td>97.06</td>
</tr>
<tr>
<td>2016-08-04</td>
<td>3371109</td>
<td>97368</td>
<td>97.11</td>
</tr>
<tr>
<td>2016-10-25</td>
<td>3510727</td>
<td>104764</td>
<td>97.02</td>
</tr>
<tr>
<td>2016-11-01</td>
<td>3421780</td>
<td>104837</td>
<td>96.94</td>
</tr>
<tr>
<td>2016-11-15</td>
<td>3436853</td>
<td>106030</td>
<td>96.91</td>
</tr>
</tbody>
</table>

### Filtering Shared Coverage Evaluation

**Comparison Overlap-Aware**

- Coverage A: 651,074 lines hit
- Coverage B: 651,845 lines hit
- Coverage C: 651,862 lines hit
- Coverage D: 652,015 lines hit
Backup Slides
## Filtering Shared Coverage Evaluation

<table>
<thead>
<tr>
<th>File</th>
<th># lines hit</th>
<th>Directory</th>
<th># lines hit</th>
</tr>
</thead>
<tbody>
<tr>
<td>DirA\File1</td>
<td>2</td>
<td>DirA</td>
<td>2</td>
</tr>
<tr>
<td>DirB\File2</td>
<td>3</td>
<td>DirB</td>
<td>17</td>
</tr>
<tr>
<td>DirB\File3</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DirB\File4</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DirB\DirM\File5</td>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Coverage result for Test1

<table>
<thead>
<tr>
<th>Directory</th>
<th># lines hit</th>
</tr>
</thead>
<tbody>
<tr>
<td>DirB</td>
<td>17</td>
</tr>
<tr>
<td>DirA</td>
<td>2</td>
</tr>
</tbody>
</table>

Coverage result for Test1 per directory

List of directories ordered by #lines hit:
DirB, DirA

Ask SAP engineers if DirA or DirB is expected for Test1

Top directory is wrong, coverage is not specific
Overlap-Aware for Test Clusters

Overlap-aware greedy for test clusters with parallelization factor from 1 to 50