Session 2: Reverse Engineering

Orla Greevy, University of Berne, Switzerland

2nd International Workshop on Program Comprehension through Dynamic Analysis (PCODA) 2006
(colocated with 13th WCRE 2006)
Benevento, Italy
October 23rd 2006
Summary

1. Aiding the Comprehension of Testsuites
   Bas Cornelissen, Arie van Deursen and Leon Moonen
   Delft University of Technology, The Netherlands

2. A Lightweight Approach to Determining the Adequacy of Tests as Documentation
   Joris van Geet - University of Antwerp, Belgium
   Andy Zaidman - Delft University of Technology, The Netherlands
Aiding the Comprehension of Testsuites

Bas Cornelissen,
Arie van Deursen and
Leon Moonen*

Delft University of Technology (and CWI*),
The Netherlands
Goal: Visual Comprehension of Testsuites using Scenario Diagrams

Run test cases

extract execution traces

Visualize as Scenario diagrams
Challenge

Dynamic Analysis ➔
Large amount of Data
Large Scenario diagrams are hard to read
A Really big example!

[JinSight, De Pauw 1993]
Reduction: Hide Irrelevant details

Test Phases

```plaintext
{
  // Test Initialization
  ...
  // Test Execution
  ...
  // Result Validation
  ...
}
```
Apply **Limit Stack Depth Abstraction**

Limit the stack depth
Other Abstractions

- Hide constructors
- Remove irrelevant constructors

(objects not used in interactions in the execution of a test).
Easy to Read Scenario Diagram with relevant details only
Validation: Case Studies Planned

Jpacman

(Java) opensource,

testsuite > 50 test cases

CronMod

an industrial system (Java) with simple
and complex unit tests.
Summary and Comments

“Run test cases, filter out setup and result validation, apply semi-automatic abstractions (minimum stack depth) to reduce data and then visualize with scenario diagram”

Questions from Orla:

- Consider categorizing tests - define metrics?

- Consider filtering techniques (e.g. repeating sequences due to loops and recursion?)

- What are the next steps?
A Lightweight Approach to Determining the Adequacy of Tests as Documentation

Joris van Geet, University of Antwerp, Belgium

Andy Zaidman, Delft University of Technology, The Netherlands
The Research Question

Tests is to provide “living” documentation for a software system (Agile and XP).

… are the tests adequate?
Adequacy Hypothesis

“Unit tests are possibly not adequate enough for documentation purposes when they cover a number of units.”

one test for every unit (method).
The Approach: **Coverage + Isolation Factor**

1. Run test cases
2. Calculate Coverage
3. Calculate Isolation Factor (is each unit tested isolation?)
4. Trace with filter
5. Emma Tool
Analysis of Coverage Results

<table>
<thead>
<tr>
<th>Ant Version</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.6.3</td>
<td>61%</td>
</tr>
<tr>
<td>1.6.4</td>
<td>63%</td>
</tr>
<tr>
<td>1.6.5</td>
<td>65%</td>
</tr>
</tbody>
</table>

(~ 2/3 Methods are covered)

“not all parts of the system require thorough documentation”
Look at the **Isolation Factor** in more detail…

- Execution trace
- **Isolation Factor**
- Methods Called by Test
  - For all tests
- Tests Calling a Method
  - For all methods
- Dependency Relationships as sets
Average number of Tests for a Method

<table>
<thead>
<tr>
<th>Version</th>
<th>mean</th>
<th>σ</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.6.3</td>
<td>68.02</td>
<td>190.35</td>
</tr>
<tr>
<td>1.6.4</td>
<td>64.48</td>
<td>183.49</td>
</tr>
<tr>
<td>1.6.5</td>
<td>64.14</td>
<td>182.40</td>
</tr>
</tbody>
</table>

A method is tested by ~ 64 tests
### Average number of Methods invoked by a test

<table>
<thead>
<tr>
<th>version</th>
<th>mean</th>
<th>$\sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.6.3</td>
<td>230.45</td>
<td>146.10</td>
</tr>
<tr>
<td>1.6.4</td>
<td>215.68</td>
<td>135.68</td>
</tr>
<tr>
<td>1.6.5</td>
<td>215.41</td>
<td>137.01</td>
</tr>
</tbody>
</table>

**values are highly variable**

A test invokes ~ 215 methods
Box Plot: **Number of Methods tested by a Test**

Half of Tests test > 212 methods

Half of Tests test < 212 methods

Similar results as obtained by averages
Box plot distribution shows **Number of Tests** that test a method

- Few outliers
- Half the methods tested by no more than 7 tests
Anecdotal Evidence - Browse code

Tests share common setup (filter?)
Identify “often called” methods (characterization?)

Conclusion:
An integration test strategy has been adopted
How do testsuites evolve?

Evolution of tests

The amount of unique methods tested increases over time

A test tests more methods over time

New functionality is being tested
Summary of Paper

“Run test cases, calculate coverage and isolation factor and determine if tests are adequate.

Coverage > 60% AND Isolation Factor High”

Questions from Orla:

- Consider categorizing tests - define measurements?
- Consider evolution of tests - history measurements?
- Consider characterizing methods?
- Consider more case studies?
- What are the next steps?
Session 2: Summary

Bas, Arie and Leon: Visual Comprehension of Tests
- Test Phase identification
- Abstractions

Joris and Andy: Tests as adequate Documentation?
- Coverage
- Isolation Factor

What are the overlaps and synergies?