Evolutionary white-box software test with the EvoTest Framework, a progress report

Peter M. Kruse
April 1st, 2009
Table of Contents

• Motivation
• White Box Testing / Structural Testing
• Evolutionary Structural Testing
• EvoTest
• Case Studies
• Application
• Results
• Way forward
• Conclusion
Motivation

• Subsystems of today’s luxury cars have up to 15 million lines of code
• Test automation needed

• Can Evolutionary Testing be the solution?
  – Some prototypes exists
  – Still mainly research-based

• Why?
White Box Testing / Structural Testing

• **Aim:**
  – Maximal code coverage
  – Little effort as possible
  – Efficient selection of test cases

• **Usage:**
  – During unit-testing phase of a software project

• **Problem:**
  – Finding test cases which exercise all branches is a complex task

1) As required by ISO 26262, IEC 61508, DO-178B
Evolutionary Structural Testing
EvoTest

• EU funded research project (IST-33472)
• Aim:
  – Find solutions for software testing
  – Using evolutionary adaptive techniques
• EvoTest Framework:
  – Automated structural testing (White Box)
  – Functional testing (Black Box)
  – Signal Generator Component
  – …

http://www.evotest.eu/
Case Studies

- 4 case studies
- Embedded Software Module
- Automotive Industry
- Implemented in C Language

<table>
<thead>
<tr>
<th>Case Study</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Active Brake Assist software module (<em>dSpace TargetLink</em>)</td>
</tr>
<tr>
<td>B</td>
<td>Adaptive Headlight control software module (<em>dSpace TargetLink</em>)</td>
</tr>
<tr>
<td>C</td>
<td>Door-Lock control software module (<em>ETAS ASCET</em>)</td>
</tr>
<tr>
<td>D</td>
<td>Electric window control software module (<em>ETAS ASCET</em>)</td>
</tr>
</tbody>
</table>
Application

- Integration with Eclipse CDT

```c
/*
 * paper.c
 * Created on: 04.03.2009
 * Author: gross
 */

char global_short;

int (int local_int)
{
    int local_2 = global_short & local_int;
    if ((global_short == 2) && (local_int < 0))
        return 1;
    else if (global_short > 3)
        return 2;
    else
        return local_2;
}
```
Application

- Integration with Eclipse CDT
Application

- Result view
Application

• Result view

```c
/*
 * paper.c
 *
 * Created on: 04.03.2009
 * Author: gross
 */

char global_short;

int int local_int;

int local_2 = global_short && local_int;

if ((global_short == 2) && (local_int < 0))
    return 1;
else if (global_short > 3)
    return 2;
else
    return local_2;

// Test Case n°1
// global variables:
char global_short = 126;
// parameters
int local_int = 0;

// Test Case n°2
// global variables:
char global_short = -58;
// parameters
int local_int = 0;

// Test Case n°3
// global variables:
char global_short = 69;
// parameters
int local_int = 1;
// end```

Covered 6 out of 10 branches: 60%
## Results

<table>
<thead>
<tr>
<th>Case Study</th>
<th>Total Functions</th>
<th>Functions containing if or while statements</th>
<th>Test case generation successful</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>2</td>
<td>1 (1)</td>
</tr>
<tr>
<td>B</td>
<td>77</td>
<td>44</td>
<td>34 (48)</td>
</tr>
<tr>
<td>C</td>
<td>486</td>
<td>70</td>
<td>30 (50)</td>
</tr>
<tr>
<td>D</td>
<td>197</td>
<td>67</td>
<td>3 (27)</td>
</tr>
<tr>
<td>All</td>
<td>762</td>
<td>183</td>
<td>68 (126)</td>
</tr>
</tbody>
</table>

- **Coverage**
  - 37 % with early version
  - 69 % with later version
### Results

<table>
<thead>
<tr>
<th>Reason for failure to generate test cases</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pointer to simple type</td>
<td>62</td>
</tr>
<tr>
<td>Array operations / Pointer to arrays</td>
<td>43</td>
</tr>
<tr>
<td>Pointer to void</td>
<td>12</td>
</tr>
</tbody>
</table>
Coverage Rates

Distribution of branches in functions from Case Studies A & B

- Percentage of functions
- Number of branches

Distribution of branch coverage achieved by ETF Structural Test tool for Case Studies A and B

- Percentage of functions
- Branch Coverage (%)
Way forward

• **Parameter Reduction**
  - Large search space increases duration of test case generation
  - Unnecessary parameters might be of unsupported type
  ➢ Using data-flow analysis, increased coverage of software modules could be achieved
    As described by M. Harmen et al. “The Impact of input domain reduction on search-based data generation”, 2007

• **Support for pointers**
  - Major reason for failure of test-case generation
  - Pointers are too common to not support them
  ➢ Has been established in later version of tool
    Approach described by M. Prutkina & A. Windisch “Evolutionary structural testing of software with pointers”, 2008
Way forward

• **Volatile variables**
  – Variables can be memory-mapped to I/O registers
  – Variable values can change asynchronously to program flow
  ➢ Since the function under test is executed in isolation, changes to volatile variables need to be simulated

• **Multi-Function Instrumentation**
  – Test case generation only for single isolated functions
  – Called functions are not yet instrumented
  ➢ Optimizing the order of evaluation of single functions can lead to a reduction in generated test data for the module as a whole

• **GUI Improvements**
Conclusion

• The tool requires the source code under test to be preprocessed

• Despite the tool still being a prototype, function coverage of 69% was possible

• Significant work is still required before evolutionary structural testing is ready for industrial application

• Nevertheless this promising approach deserves further research
• Thank you

Peter M. Kruse
Berner & Mattner Systemtechnik
peter.kruse@berner-mattner.com