Fault Diagnosis of Embedded Software using Program Spectra

Peter Zoeteweij\textsuperscript{1}
Rui Abreu\textsuperscript{1}, Rob Golsteijn\textsuperscript{2}, Arjan van Gemund\textsuperscript{1}

\textsuperscript{1}Embedded Software Lab / SERG, TU Delft
\textsuperscript{2}NXP
TRADER project

• Improve the user-perceived reliability of high-volume consumer electronics devices
• Test case: television platform from NXP
• Partners:
  – Universities of Delft, Twente, Leiden,
  – Embedded Systems Institute, Design Technology Institute, IMEC Leuven
  – NXP, NXP Corporate Research, Philips TASS
Spectrum-based diagnosis

- Localize the faults that are the root cause of detected errors
- In software: computer aided, or automated debugging
  
- Improves the efficiency of the debugging stage: more bugs solved leads to more reliable systems
- Spectrum-based diagnosis lends itself well for integration with (automated) testing
Integration with testing

Test suite

- t1
- t2
- t3
- t4
- t5
Integration with testing

Test suite

- t2
- t3
- t4
- t5

Status

- t1  ✔
Integration with testing

Test suite

t3
t4
t5

Status

t1  ✔
t2  ✔
Integration with testing

Test suite

Status

- t1
- t2
- t3

Green squares represent successful tests, while red squares indicate failed tests.
Integration with testing

Test suite

Status

- t1: ✔
- t2: ✔
- t3: ✗
- t4: ✔
- t5
Integration with testing

Status
- t1 ✓
- t2 ✓
- t3 ✗
- t4 ✓
- t5 ✗
Integration with testing

System components are ranked according to likelihood of causing the detected errors.

Status
- t1 ✓
- t2 ✓
- t3 ✗
- t4 ✓
- t5 ✗
Spectrum-based diagnosis

• Debugging technique
• Easy to integrate with testing
• Low memory / CPU overhead: well suited for embedded systems

• Black box diagnosis:
  – Does not rely on modeling
  – No guarantee on accuracy

• Effective in practice
Outline

• Spectrum-based fault localization
• Case study
• Research
Terminology

- **failure**: delivered service ≠ correct service
  - (segmentation fault)
- **error**: system state that may cause a failure
  - (index out of bounds)
- **fault**: the cause of an error in the system
  - (bug: array index un-initialized)
Program spectra

- Execution profiles that indicate, or count which parts of a software system are used in a particular test case
- Introduced in [Reps97] for diagnosing Y2K problems
- Many different forms exist [Harrold98]:
  - Spectra of program locations
  - Spectra of branches / paths
  - Spectra of data dependencies
  - Spectra of method call sub-sequences
Block / function hit spectra

<table>
<thead>
<tr>
<th></th>
<th>Block hit spectrum</th>
<th>Function hit spectrum</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_i$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_n$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1: function $i$ called
0: function $i$ not called

1: block $i$ executed
0: block $i$ not executed

Block:
- C statement (compound stmt)
- cases of a switch statement
Fault diagnosis

1. Spectra for \( m \) test cases

\[
\begin{array}{cccc|c}
\text{blocks} & x_{11} & x_{12} & \ldots & x_{1n} \\
\text{cases} & x_{21} & x_{22} & \ldots & x_{2n} \\
& \ldots & \ldots & \ldots & \ldots \\
& x_{m1} & x_{m2} & \ldots & x_{mn} \\
\end{array}
\]

\[
\begin{array}{c}
e_1 \\
e_2 \\
\ldots \\
e_m
\end{array}
\]
## Fault diagnosis

1. Spectra for $m$ test cases

<table>
<thead>
<tr>
<th>$x_{11}$</th>
<th>$x_{12}$</th>
<th>...</th>
<th>$x_{1n}$</th>
<th>$e_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{21}$</td>
<td>$x_{22}$</td>
<td>...</td>
<td>$x_{2n}$</td>
<td>$e_2$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$x_{m1}$</td>
<td>$x_{m2}$</td>
<td>...</td>
<td>$x_{mn}$</td>
<td>$e_m$</td>
</tr>
</tbody>
</table>

Row $i$: the blocks that are executed in case $i$
Fault diagnosis

1. Spectra for $m$ test cases

\[
\begin{array}{cccc}
 x_{11} & x_{12} & \cdots & x_{1n} \\
 x_{21} & x_{22} & \cdots & x_{2n} \\
 \vdots & \vdots & \ddots & \vdots \\
 x_{m1} & x_{m2} & \cdots & x_{mn} \\
 \end{array}
\]

Column $j$ : the test cases in which block $j$ was executed
Fault diagnosis

1. Spectra for $m$ test cases
2. Error detection per test case

<table>
<thead>
<tr>
<th></th>
<th>$x_{11}$</th>
<th>$x_{12}$</th>
<th>...</th>
<th>$x_{1n}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{21}$</td>
<td>$x_{22}$</td>
<td>...</td>
<td>$x_{2n}$</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>$x_{m1}$</td>
<td>$x_{m2}$</td>
<td>...</td>
<td>$x_{mn}$</td>
<td></td>
</tr>
</tbody>
</table>

$e_i = 1$ : error in the $i$-th test
$e_i = 0$ : no error in the $i$-th test
Fault diagnosis

Compare every column vector with the error vector.

<table>
<thead>
<tr>
<th>$x_{11}$</th>
<th>$x_{12}$</th>
<th>...</th>
<th>$x_{1n}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{21}$</td>
<td>$x_{22}$</td>
<td>...</td>
<td>$x_{2n}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$x_{m1}$</td>
<td>$x_{m2}$</td>
<td>...</td>
<td>$x_{mn}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>error vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
</tr>
<tr>
<td>$e_2$</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>$e_m$</td>
</tr>
</tbody>
</table>

Similarity $s_j$
Fault diagnosis

Jaccard similarity coefficient:

\[ s_j = \frac{a_{11}}{a_{11} + a_{10} + a_{01}} \]
Fault diagnosis

Jaccard similarity coefficient:

\[ s_j = \frac{a_{11}}{a_{11} + a_{10} + a_{01}} \]
Fault diagnosis

Jaccard similarity coefficient:

\[ s_j = \frac{2}{2 + a_{10} + a_{01}} \]
Fault diagnosis

Jaccard similarity coefficient:

\[ s_j = \frac{2}{2 + 1 + a_{01}} \]
Fault diagnosis

Jaccard similarity coefficient:

\[
s_j = \frac{2}{2 + 1 + 1}
\]
Fault diagnosis

For every block: similarity with the error “block”

<table>
<thead>
<tr>
<th></th>
<th>$x_{11}$</th>
<th>$x_{12}$</th>
<th>…</th>
<th>$x_{1n}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{21}$</td>
<td>$x_{22}$</td>
<td>…</td>
<td></td>
<td>$x_{2n}$</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>$x_{m1}$</td>
<td>$x_{m2}$</td>
<td>…</td>
<td></td>
<td>$x_{mn}$</td>
</tr>
</tbody>
</table>

$s_1$  $s_2$  …   $s_n$

The block with the highest $s_i$ most likely contains the fault.
## Fault diagnosis

### $n$ blocks

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

### Error vector

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>
# Fault diagnosis

<table>
<thead>
<tr>
<th>( n ) blocks</th>
<th>error vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 1 1 1 0 0</td>
<td>0</td>
</tr>
<tr>
<td>0 0 0 1 0 1 1</td>
<td>1</td>
</tr>
<tr>
<td>1 1 1 1 0 0 0</td>
<td>1</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0</td>
<td>0</td>
</tr>
<tr>
<td>1 1 0 1 1 0 1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( m ) cases</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( \frac{2}{3} )</td>
<td>( \frac{1}{2} )</td>
</tr>
<tr>
<td>( \frac{1}{4} )</td>
<td>( \frac{3}{4} )</td>
</tr>
<tr>
<td>( \frac{1}{4} )</td>
<td>( \frac{1}{3} )</td>
</tr>
<tr>
<td>( \frac{2}{3} )</td>
<td></td>
</tr>
</tbody>
</table>
Fault diagnosis

<table>
<thead>
<tr>
<th>n blocks</th>
<th>error vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 1 1 1 1 0 0</td>
<td>0</td>
</tr>
<tr>
<td>0 0 0 1 0 1 1</td>
<td>1</td>
</tr>
<tr>
<td>1 1 1 1 0 0 0</td>
<td>1</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0</td>
<td>0</td>
</tr>
<tr>
<td>1 1 0 1 1 0 1</td>
<td>1</td>
</tr>
</tbody>
</table>

$\frac{2}{3}$ $\frac{1}{2}$ $\frac{1}{4}$ $\frac{3}{4}$ $\frac{1}{4}$ $\frac{1}{3}$ $\frac{2}{3}$
Spectrum-based diagnosis

- Large on-line transaction processing systems (search engines, web mail) [Chen02]
- Java software (method call sequences) [Dallmeier05]
- Visualizing test information to aid manual debugging [Jones02]
Embedded systems

• Low overhead
• Little infrastructure needed
• Consumer electronics
  – No time for exhaustive debugging
  – Helps to identify responsible teams / developers

• Diagnosis can drive a recovery mechanism, e.g., rebooting suspect processes
Case study – platform

- Control software of an analog TV
- Decoding RC input, displays the on-screen menu, teletext, optimizes parameters for audio / video processing based on signal analysis, etc.
- 450 K lines of C code
- 2 MB of RAM + 2 MB in development version
- CPU: MIPS running a small multi-tasking OS
- Work is organized in 315 logical threads
Case study

1. Load problem:
Diagnosis

- 150 hit spectra of 315 functions, corresponding to the logical threads (one per second):
  60 sec. TV, 30 sec. TXT, 60 sec. TV
- Marked the last 60 spectra as failed
- 2nd in ranking of 315 functions
Case study

2. Teletext lock-up:
   - Existing problem in another product line
   - Copied to our platform, triggered by a remote control key sequence
   - Inconsistency in two state variables, for which only specific combinations are allowed
Diagnosis

• 23 spectra (one per key pressed):
  – Verify that TV and teletext are working properly
  – Trigger the fault
  – Check that the teletext system is locked up

• Automatic instrumentation of the source code for recording block hit spectra using Front parser generator

• Error detection based on comparing state variables

• Shared 1st position in ranking of >60,000 blocks; 3 more key presses yield a perfect diagnosis
Conclusion

- Spectrum-based fault diagnosis can help to improve the efficiency of the debugging phase
- It can easily be integrated with testing
- It is a feasible technique in the area of embedded systems
- Proof of concept tested on industrial code (TV software) and representative errors
Outlook

• What is the effect of various external factors on the diagnostic accuracy?
  – Quality of the error detection information
  – Number of observations available
  – Strategies for selecting passed tests

• How to exploit knowledge about a system?
  – Known data / control dependencies
  – Known hierarchical relationships

• Compiler that generates code for
  – Recording spectra
  – Various error detection mechanisms
  – Diagnosis
References

- [Chen02] Chen et al. *Problem determination in large dynamic internet services*. DSN’02
- [Dallmeier05] Dallmeier, Lindig, Zeller, *Lightweight defect localization for Java*. ECOOP’05
- [Jones02] Jones, Harrold, Stasko, *Visualization of test information to assist fault localization*. ICSE’02
- [Reps97] Reps, Ball, Das, Larus, *The use of program profiling for software maintenance with applications to the year 2000 problem*. ESEC/FSE’97
Current research

- Effect of quality of error detection
- Effect of quantity of observations
- Siemens test set [Hutchins94]:
  - 7 programs, 20 – 124 blocks
  - 7 – 32 faulty versions per program: 132 faults in total
  - Up to 1000 – 5000 test cases per program: full code coverage
Error detection quality

Small fraction of fault activations detected is enough

![Graph showing diagnostic accuracy vs. % faults -> errors](image-url)
Number of runs

• On average, for the Siemens set:
  – Adding more failed tests is safe
  – 6 failed tests are enough
  – The number of passed tests has no influence

• However:
  – For individual runs the effect of adding passed tests differs
  – It stabilizes around 20 passed tests
Influence of #runs
Influence of #runs

- On average, for our benchmark:
  - Adding failed runs is safe
  - 6 failed runs is enough
  - The number of passed runs has no influence

- However
  - For individual runs, the effect of more passed runs differs
  - It stabilizes around 20