An Observation-based Model for Fault Localization

R. Abreu  P. Zoeteweij  A.J.C. van Gemund

Delft University of Technology

WODA, July 2008
Motivation

Situation

- SW Debugging is overly expensive
- Many debugging tools/approaches
- Model-based (MBD)
- Dynamic/statistics-based (SFL)

Rationale

- MBD needs a model as input which is often not available
- SFL cannot distinguish components with the same execution pattern

In this presentation, a new novel approach...

- Dynamic information to extract a model ▶ inspired by SFL
- Candidates ranked using Bayes’ update ▶ inspired by MBD
Outline

Concepts and Definitions

Observation-based Model

Evaluation
  Synthetic
  Experimental

Conclusions & Future Work
A program under analysis comprises a set of $M$ components.

Statements in the context of this paper.

The program is executed using $N$ test cases (runs).

Component activity is recorded in terms of program spectra.

Program spectra is a set of counter or flags for each component.

In this presentation, statement-hit spectra is used.
Observation Matrix

- Row $O_{i*}$ indicates whether a component was involved in run $i$.
- Column $O_{*j}$ indicates in which runs component $j$ was involved.
- The error vector $e$ indicates whether a run has failed or passed.

$$
O = \begin{bmatrix}
O_{11} & O_{12} & \cdots & O_{1M} & e_1 \\
O_{21} & O_{22} & \cdots & O_{2M} & e_2 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
O_{N1} & O_{N2} & \cdots & O_{NM} & e_N \\
\end{bmatrix}
$$

- Input to the debugging method is **only** $O$. 
Outline

Concepts and Definitions

Observation-based Model

Evaluation

Synthetic
Experimental

Conclusions & Future Work
Model Generation

- Compile the observation matrix into propositional logic
  - More specifically, conjunctions of disjunctions

- Suppose the following source code and program spectra

```c
(y1,y2) 3inv(bool x) {
1. w = !x
2. y1 = !w;
3. y2 = w; //fault: ! missing
   return (y1,y2);
}
```

<table>
<thead>
<tr>
<th></th>
<th>c1</th>
<th>c2</th>
<th>c3</th>
<th>e</th>
<th>obs1</th>
<th>obs2</th>
<th>obs3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Yields the following propositional logic

\[
(\neg h_1 \lor \neg h_3) \land (\neg h_2 \lor \neg h_3)
\]
Solve the Model

- Compute the minimal hitting set
  - NP complete
  - TUDelft heuristic: STACCATO

- The solution for the example’s model
  \[(\neg h_1 \vee \neg h_3) \land (\neg h_2 \vee \neg h_3)\]

\[\Downarrow\]

\[(\neg h_3) \vee (\neg h_1 \land \neg h_2)\]

- Thus, either \(c_3\) is faulty or \(c_1\) and \(c_2\) are faulty
Observation-based Model

Ranking Diagnoses

- Set of diagnosis candidates can be large
- Bayes’ update to compute probabilities

\[
\Pr(d_k|\text{obs}) = \frac{\Pr(\text{obs}|d_k)}{\Pr(\text{obs})} \cdot \Pr(d_k)
\]

where

- \(\Pr(d_k) = p^{d_k} \cdot (1 - p)^{M-|d_k|}\) and, e.g., \(p = 0.01\)

\[
\Pr(\text{obs}|d_k) = \begin{cases} 
0 & \text{if } \text{SD} \land \text{obs} \land d_k \models \bot \\
1 & \text{if } d_k \rightarrow \text{obs} \land \text{SD} \\
\varepsilon & \text{if } d_k \rightarrow \{\text{obs}_1 \land \text{SD}, \ldots, \text{obs} \land \text{SD}, \ldots, \text{obs}_k \land \text{SD}\}
\end{cases}
\]

\[
\varepsilon = \begin{cases} 
g(d_k)^t & \text{if run passed} \\
1 - g(d_k)^t & \text{if run failed}
\end{cases}
\]
g estimates the probability that components in $d_k$ produce a correct output

\[
g(d_k) = \frac{\sum_{i=1..N} \left[ (\bigvee_{j \in d_k} o_{ij} = 1) \land e_i = 0 \right]}{\sum_{i=1..N, j \in d_k} \left[ \bigvee o_{ij} = 1 \right]}
\]

Back to our example...

<table>
<thead>
<tr>
<th>$d_k$</th>
<th>Pr($d_k$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>{3}</td>
<td>0.995</td>
</tr>
<tr>
<td>{1,2}</td>
<td>0.005</td>
</tr>
</tbody>
</table>

Meaning that, one would start by inspecting component 3
Outline

Concepts and Definitions

Observation-based Model

Evaluation
  Synthetic
  Experimental

Conclusions & Future Work
Experimental Setup

- Study the effects of the following on the diagnostic accuracy
  - Number of Failing Runs
  - Behavior for Small Number of Runs
  - Behavior for Large Number of Runs

- Observation matrices built based on
  - Probability a component is touched $r$
  - Probability a faulty component fails $g$
  - Fault cardinality $C$

- Evaluation Metric: Wasted Effort
(c) \( g = 0.1 \) and \( r = 0.6 \)

(d) \( g = 0.1 \) and \( r = 0.4 \)

(e) \( g = 0.9 \) and \( r = 0.6 \)

(f) \( g = 0.9 \) and \( r = 0.4 \)
**Optimal $N^*$ for perfect diagnosis ($r = 0.6$)**

<table>
<thead>
<tr>
<th>$g$</th>
<th>0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C$</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>$N^*$</td>
<td>13 31 90 120 250</td>
</tr>
<tr>
<td>$N_F$</td>
<td>5 19 71 111 245</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$g$</th>
<th>0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C$</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>$N^*$</td>
<td>200 300 500 1000 1700</td>
</tr>
<tr>
<td>$N_F$</td>
<td>12 36 84 219 459</td>
</tr>
</tbody>
</table>
Outline

Concepts and Definitions

Observation-based Model

Evaluation

Synthetic

Experimental

Conclusions & Future Work
Experimental Setup

Programs

- Siemens set of programs
  - 7 programs with several (single fault) faulty versions
  - $O(100)$ LOC
  - $O(1000)$ test cases
- GNU gcov to obtain the observation matrix

Evaluation Metric

- Percentage of code that needs to be inspected

$$Effort = \frac{\text{position of fault location}}{\text{LOC}}$$
Experimental Results

Cumulative Percentage of Located Faults
Outline

Concepts and Definitions

Observation-based Model

Evaluation
  Synthetic
  Experimental

Conclusions & Future Work
Conclusions

▶ Fault localization approach
  ▶ Uses abstraction of program traces to generate a (dynamic, sub-) model
  ▶ The set of traces for pass/fail executions is used to reason about the observed failures

▶ Set of candidates also contains multiple-fault explanations

▶ Theoretically, given sufficient test cases are available, this approach will reveal the true faulty state

▶ Results using the Siemens set have shown that our approach outperforms other state-of-the-art approaches
Future Work

- Study the diagnostic performance for multiple-fault programs
- Study the possibility of engaging several developers to find the faults
- Reducing the hitting set algorithm complexity
  - STACCATO is under development
- Apply to other (real) programs (e.g., space)
For more info:

- http://www.st.ewi.tudelft.nl/~abreu
- email: r.f.abreu@tudelft.nl