Automated Software Program Bug Localization : A Survey

Debjit Pal (dpal2)
Rizwan Mohiuddin (rmohiud2)
CS 512 Survey Presentation
Spring 2013
Date : 7th April, 2013
Motivation

**Buggy Code Replace v3**

- void
- subline(lin, pat, sub)
- char *lin;
- char *pat;
- char *sub;
{
  int i, lastm, m;

  lastm = -1;
  i = 0;
  while ((lin[i] != ENDSTR))
  {
    m = amatch(lin, i, pat, 0);
    if ((m >= 0) /* && (lastm != m) */) {
      putsub(lin, i, m, sub);
      lastm = m;
    }
    if ((m == -1) || (m == i)) {
      fputc(lin[i], stdout);
      i = i + 1;
    } else
      i = m;
  }
}

**Original Code Replace v3**

- void
- subline(lin, pat, sub)
- char *lin;
- char *pat;
- char *sub;
{
  int i, lastm, m;

  lastm = -1;
  i = 0;
  while ((lin[i] != ENDSTR))
  {
    m = amatch(lin, i, pat, 0);
    if ((m >= 0) && (lastm != m)) {
      putsub(lin, i, m, sub);
      lastm = m;
    }
    if ((m == -1) || (m == i)) {
      fputc(lin[i], stdout);
      i = i + 1;
    } else
      i = m;
  }
}
Different Bug Localization Techniques

**Static Techniques:**
- BLAST, Microsoft's SLAM, CLANG

**Parametric Analysis:**
- CBI, SOBER

**Non-Parametric Analysis:**
- DES

**Path Profiling:**
- HOLMES

**Predicate Profiling:**
- CBI, SOBER
Prominent Statistical Bug Localization Techniques

- Cooperative Bug Isolation (CBI) Project (Liblit et. al)
- Tarantula (Jones, Harrold)
- SOBER (Liu, Han et. al)
- PATHGEN and PATHPOST (Jiang et. al)
- HOLMES (Chilimbi, Liblit)
- DES (Zhang et. al)
Cooperative Bug Isolation (CBI) : Predicate Profiles

- **P** : A predictor in the program
- **R(P)** : Feedback report on the correctness of a execution of a program
- Locates predictors that are associated with a Bug,
- Source to Source Transformation
- Sparse Random Sampling to collect predicate profiles
• **Instrumented Predicate Types:**

1. **Branches:** if-else statements
2. **Returns:** Scalar returning function. Return value is \(<0, \leq 0, =0, \neq 0, >0, \geq 0\)
3. **Scalar Pairs:** Values at assignment statement. Six predicates are tracked for new value of \(x\): \(<, \leq, =, \neq, >, \geq\)

• **Methodology:**

1. Identify potential candidates of predictors for bug B.
   a) Removal of Predicates that are bug irrelevant in terms of a score called \(Increase(P)\)
   b) Rank the surviving predicate (Ranking model considers specificity and sensitivity)
2. Emulate the condition as if the Bug has been removed.
3. Iterate until all the bugs are explained or the list of predictor
Tarantula

• **Four techniques**
  - Set Union
  - Set Intersection
  - Nearest Neighbor
  - Cause Transitions

• **Comparison of Above Techniques**
  - Efficiency
  - Effectiveness

• **Principle**
  Entries in a program that are primarily executed by failed test cases are more likely to be faulty than those that are primarily executed by passed test cases.
Tarantula Continued..

• **Hue Tool**
  - **RED** => Executed by failed tests; highly suspicious of being faulty.
  - **GREEN** => Executed by passed test cases; likely not faulty.

\[
hue(s) = \frac{\text{passed}(s)}{\text{total} \times \text{passed}} + \frac{\text{failed}(s)}{\text{total} \times \text{failed}}\]

• **Statement Suspiciousness (s)**
  - **0** => ranked lower, least likely to be faulty.
  - **1** => ranked higher most likely to be faulty.

<table>
<thead>
<tr>
<th>mid() {</th>
<th>Test Cases</th>
<th>Suspiciousness</th>
</tr>
</thead>
<tbody>
<tr>
<td>int x, y, z, m;</td>
<td>3.3.5</td>
<td>12.2</td>
</tr>
<tr>
<td>1: read(&quot;Enter 3 numbers:\n&quot;,x,y,z);</td>
<td>● ● ● ● ● ●</td>
<td>0.5</td>
</tr>
<tr>
<td>2: m = z;</td>
<td>● ● ● ● ● ●</td>
<td>0.5</td>
</tr>
<tr>
<td>3: if (y&lt;z)</td>
<td>● ● ● ● ● ●</td>
<td>0.5</td>
</tr>
<tr>
<td>4: if (x&lt;y)</td>
<td>● ● ● ● ● ●</td>
<td>0.63</td>
</tr>
<tr>
<td>5: m = y;</td>
<td>● ● ● ● ● ●</td>
<td>0.0</td>
</tr>
<tr>
<td>6: else if (x&lt;z)</td>
<td>● ● ● ● ● ●</td>
<td>0.71</td>
</tr>
<tr>
<td>7: m = y; // *** bug ***</td>
<td>● ● ● ● ● ●</td>
<td>0.83</td>
</tr>
</tbody>
</table>
Tarantula Continued..

**Effectiveness**

**Efficiency**

![Comparison of fault localization techniques](chart.png)

**Table 3: Average time expressed in seconds.**

<table>
<thead>
<tr>
<th>Program</th>
<th>Tarantula (computation only)</th>
<th>Tarantula (including I/O)</th>
<th>Cause Transitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>print_tokens</td>
<td>0.0040</td>
<td>68.96</td>
<td>2590.1</td>
</tr>
<tr>
<td>print_tokens2</td>
<td>0.0037</td>
<td>50.50</td>
<td>6556.5</td>
</tr>
<tr>
<td>replace</td>
<td>0.0063</td>
<td>75.90</td>
<td>3588.9</td>
</tr>
<tr>
<td>schedule</td>
<td>0.0032</td>
<td>30.07</td>
<td>1909.3</td>
</tr>
<tr>
<td>schedule2</td>
<td>0.0030</td>
<td>30.02</td>
<td>7741.2</td>
</tr>
<tr>
<td>tcas</td>
<td>0.0025</td>
<td>12.37</td>
<td>184.8</td>
</tr>
<tr>
<td>tot_info</td>
<td>0.0031</td>
<td>8.51</td>
<td>521.4</td>
</tr>
</tbody>
</table>
SOBER : Parametric Analysis

• Models truth of predicate evaluation in **Correct** and **Incorrect** Runs.

• Evaluation Bias $\pi(P) = \frac{n_t}{n_t + n_f}$

• EB from multiple executions form Random Sample.

• $L(P) = \text{Sim}(f(X|\theta_p), f(X|\theta_f))$

• **Score Function** : $-\log(L(P))$
SOBER : Continued..

• **Difficulties of Ranking Procedure :**
  - Lack of prior knowledge of \( f_P(X|\theta) \)
  - Closed forms of \( f_P(X|\theta_p) \) and \( f_P(X|\theta_f) \)

• **Alternative way of Ranking :**
  - Assume \( f_P(X|\theta_p) = f_P(X|\theta_f) \)
  - Evaluate \( Y \), a new statistic based on observed EB on failure runs
  - Null Hypothesis : \( Y \) conforms to Normal Distribution

• **Differences with CBI**

<table>
<thead>
<tr>
<th>CBI</th>
<th>SOBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>P is predictor if its true evaluation causes program to crash</td>
<td>P is predictor if EB in failed and passed run differ significantly</td>
</tr>
<tr>
<td>Explores a single evaluation of P</td>
<td>Explores multiple evaluation of P in a execution of a program</td>
</tr>
</tbody>
</table>
Context-Aware Debugging

• **Association of Bug Locations and Control Flow Paths.**

• **Three Step Procedure:**
  - Instrumentation of source code through CBI framework
  - Feature Selection (failure related predicates as bug predictors) and Clustering (to group correlated predicates)
    • **Feature Selection**: *Using Support Vector Machine as Classifier*
    • **Clustering**: *Using k-means algorithm*
      – Traversal of Control Flow Graphs guided by branch prediction info from previous step.
HOLMES Framework : Program Path Profiling

Variants of Algorithms:

1. Non-Adaptive Debugging
2. Adaptive Debugging
   - Monitor Un-instrumented Program
   - Use Static Analysis and Bug Report to identify the portions of code prone to bug
   - Calculate Statistical Model using Non-Adaptive method
   - If model identifies strong bugs, report else continue

Implementation:

1. Bootstraping
2. Iterative Profiling
Fault Localization with Non-Parametric Analysis

• This paper proposes a non-parametric approach to measuring the similarity of the feature spectra of successful and failed runs.

• The proposed model is based on predicate-based fault localization. A general hypothesis testing model called Mann-Whitney test is used.

• The Mann-Whitney test is a non-parametric hypothesis testing test which is used to determine the degree of difference between spectra of program elements for successful and failed runs.

• This degree of difference is used to measure ranking score, with predicates having high-ranking score being more suspicious for errors.
Non-Parametric Analysis Contd..

• The idea is that if a predicate is relevant to a fault, the difference in its probability density functions of the evaluation biases of that predicate on whole sets of possible successful and failed runs should be large. The larger this difference, the more relevant that predicate is in relation to the fault.

• Mann-Whitney:
  – The main idea of this test is to transform the two sample sets of evaluation biases for a predicate from a number of successful and failed runs, to two rank-value sets and then measure the distance between two rank-value sets.

• Performance

Overall performance comparisons of Top 5 T-score results
A Quest: Parametric or Non-Parametric Techniques

- Can feature spectra be considered having a normal distribution?
- Can feature spectra of fault relevant predicates be considered having normal distribution?
- Can the effectiveness of Non-Parametric techniques be decoupled from program spectra distribution shape?
Evaluation Bias of a Predicate in Grep:

\[(int \ast)(match + (best\_len -1)) = (int )scan\_endl\]
Evaluation Bias of a Bug Predictor in replace v1: $m \geq 0$
Crosstab

• The coverage information of each executable statement and the execution result (success/failure) is presented with respect to each test case.

• A crosstab is constructed for each executable statement and a statistic is computed to determine the suspiciousness of the corresponding statement.

• Executable statements are then ranked in the order of decreasing suspiciousness, with the highly suspicious statements more susceptible to errors.
Crosstab Continued..

- Suspiciousness of each executable statement \( \omega \) in terms of its likelihood of containing bugs is calculated using a cross-classification table.

<table>
<thead>
<tr>
<th>( N )</th>
<th>total number of test cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N_F )</td>
<td>total number of failed test cases</td>
</tr>
<tr>
<td>( N_S )</td>
<td>total number of successful test cases</td>
</tr>
<tr>
<td>( N_C(\omega) )</td>
<td>number of test cases covering ( \omega )</td>
</tr>
<tr>
<td>( N_{CF}(\omega) )</td>
<td>number of failed test cases covering ( \omega )</td>
</tr>
<tr>
<td>( N_{CS}(\omega) )</td>
<td>number of successful test cases covering ( \omega )</td>
</tr>
<tr>
<td>( N_{U}(\omega) )</td>
<td>number of test cases not covering ( \omega )</td>
</tr>
<tr>
<td>( N_{UF}(\omega) )</td>
<td>number of failed test cases not covering ( \omega )</td>
</tr>
<tr>
<td>( N_{US}(\omega) )</td>
<td>number of successful test cases not covering ( \omega )</td>
</tr>
</tbody>
</table>

- Null hypothesis \( H_0 \): Program execution result is independent of the coverage of statement \( \omega \).
- A chi-square test is used to determine if the hypothesis should be rejected. The chi-square statistic is given by:

\[
\chi^2(\omega) = \frac{(N_{CS}(\omega) - E_{CS}(\omega))^2}{E_{CS}(\omega)} + \frac{(N_{CS}(\omega) - E_{CS}(\omega))^2}{E_{CS}(\omega)} + \frac{(N_{US}(\omega) - E_{US}(\omega))^2}{E_{US}(\omega)} + \frac{(N_{US}(\omega) - E_{US}(\omega))^2}{E_{US}(\omega)}
\]

where \( E_{CS}(\omega) = \frac{N_{C}(\omega) \times N_S}{N} \), \( E_{CS}(\omega) = \frac{N_{C}(\omega) \times N_S}{N} \), \( E_{US}(\omega) = \frac{N_{U}(\omega) \times N_S}{N} \), \( E_{US}(\omega) = \frac{N_{U}(\omega) \times N_S}{N} \)

- Corresponding Chi-square critical value \( \chi^2{\sigma} \) is found from the Chi-square distribution table, given the level of significance \( \sigma \).
- The null hypothesis is rejected when \( \chi^2(\omega) > \chi^2{\sigma} \), meaning that the execution result was dependent on the coverage of \( \omega \).
Crosstab Continued..

- Degree of association between execution result and statement coverage is given by contingency coefficient.

\[
M(\omega) = \frac{\chi^2(\omega)/N}{\sqrt{(\text{row}-1)(\text{col}-1)}}
\]

- This value lies between [0,1]; 0 => complete independence; 1 => complete association.

- A statistic is defined:

\[
\varphi(\omega) = \frac{P_F(\omega)}{P_S(\omega)} = \frac{N_{CF}(\omega)/N_F}{N_{CS}(\omega)/N_S}
\]

- The program statements can be classified into 5 classes depending on \(\varphi(\omega)\) and \(\chi^2(\omega)\) values:
  - Statements with \(\varphi > 1\) and \(\chi^2 > \chi^2_\alpha\) have a high degree of association between their coverage and the failed execution result
  - Statements with \(\varphi > 1\) and \(\chi^2 \leq \chi^2_\alpha\) have low degree of association between their coverage and the failed execution result
  - Statements with \(\varphi < 1\) and \(\chi^2 > \chi^2_\alpha\) have a high degree of association between their coverage and the successful execution result
  - Statements with \(\varphi < 1\) and \(\chi^2 \leq \chi^2_\alpha\) have a low degree of association between their coverage and the successful execution result
  - Statements with \(\varphi = 1\) whose coverage is independent of the execution result

The order of likelihood of classes most likely to have bugs in the decreasing order is \(1>2>5>4>3\).
Fault Localization through Evaluation Sequences

- This paper investigates if the effect of a lower-tier concept – evaluation sequences – of predicates, can be a significant factor affecting the effectiveness of predicate-based statistical fault localization.

- A study was conducted to show how distribution of evaluation biases at evaluation sequence level could be used to pinpoint a fault relevant predicate.

- Evaluation sequences of original and faulty versions were tested and their counts for each test case recorded.
Experiment to check if short circuit rules are helpful in fault localizations:

- Comparison between distributions of evaluation biases for evaluation sequences in passed & failed test cases.

Using predicate-based statistical fault-localization technique was referred to as the base technique (E.G: SOBER), and the use of evaluation sequences in predicate execution counts was known as the fine-grained version of the base technique (E.G: DES).

Comparison between 2 methods, i.e., ability to rank fault relevant predicates, was done by using T-score as a measure.

The T-score results were calculated for each program in the form of plotted graphs of “% of code examined” vs. “% of faults located”. DES_SOBER and DES_CBI both achieve better average fault-localization results per % of code examined than base SOBER and CBI for each program, both in terms of effectiveness and efficiency.
Our Pathway
Non-Parametric Analysis

• Non-parametric characterization of feature spectra namely Evaluation Bias
• Use Non-parametric test statistic value as indicator of Bug predictor
• Establish proposed approach on large program other than Siemens Suite, like Grep, Gzip, EXIF, CCRYPT, MOSS, Rhythmbox
Survey Report and Contacts

• **Survey is available at the following link:**

• **Contact:**
  Debjit Pal
dpal2@Illinois.edu
  247-Coordinated Science Laboratory
  Urbana, Illinois
Thank You !!