Automated Diagnosis of Software Configuration Errors

Sai Zhang, Michael D. Ernst
University of Washington
A typical software workflow

Inputs + Software → Outputs
Modern software is often configurable
Possible root causes of wrong output

Configuration errors

Configuration options

Wrong inputs

Inputs

Bugs

Configurable Software

Outputs

Studied by many existing automated debugging techniques

Studied by this paper!
Why configuration errors?

- Fixable by changing *configuration options*
- Actionable by system administrators or end-users

- 17% of the total technical support cost [Kapoor ’03, Yin ’11]

- Configuration options *vs.* Inputs
  - **Options**: customize program behaviors by altering the control flow
  - **Input values**: produce output for a specific task
Outline

- Example
- The ConfDiagnoser Technique
- Evaluation
- Related Work
- Contributions
An example configuration error

• A “bug report” against the Randoop test generation tool
  
  … Randoop fails to generate tests for NanoXML using the following command: `java randoop.main.Main NanoXML` ...

  …, but Randoop works perfectly well on its own examples, such as BinaryTree, TreeMap, etc.
Difficulty in diagnosing the Randoop error

• A silent failure
  – No crashing points
  – No stacktrace
  – No error message
• Inputs are already minimized

Delta debugging [Zeller’02], dynamic slicing [Zhang’06], capture/replay [Whitaker’04], stack trace analysis [Rakbin’11], tainting [Attariyan’12] ...
Root cause of the Randoop configuration error

57 Randoop options in total

... maxsize = 100 ...

Randoop code:

Sequence seq = createNewSeq();
if (seq.size() > maxsize) {
    return null;
}
...

Resolve the reported "bug":

java randoop.main.Main --maxsize=1000 NanoXML
ConfDiagnoser’s diagnosis report

- A ranked list of suspicious configuration options
- The top-ranked option for the Randoop error:

  Suspicious configuration option: maxsize

  It affects the behavior of predicate:
  “newSequence.size() > GenInputsAbstract.maxsize”
  (line 312, class: randoop.ForwardGenerator)

  This predicate evaluates to true:
  3.3% of the time in normal runs
  32.5% of the time in the undesired run
Outline

• Example
• The ConfDiagnoser Technique
• Evaluation
• Related Work
• Contributions
Outline

- Example
- The ConfDiagnoser Technique
- Evaluation
- Related Work
- Contributions
ConfDiagnoser’s assumptions

Configuration errors

Configuration options

Wrong inputs

Inputs

Correct Execution Traces

Configurable Software

Bugs

Outputs
ConfDiagnoser’s assumptions

Configuration errors

Configuration options

ConfDiagnoser

Correct Execution Traces

Inputs

Configurable Software

Outputs

Report

1.
2.
3.
ConfDiagnoser’s advantages

• Fully-automatically diagnoses configuration errors
• Diagnoses both crashing and non-crashing errors
• Requires no OS-level support
ConfDiagnoser’s insight

- Control flow propagates most configuration options’ effects

- Correct execution traces serve as approximate oracles
  - The control flow difference provides debugging clues

```java
//a configuration option
int maxsize = readFromCommandLine();
...
Sequence seq = createNewSeq();
if (seq.size() > maxsize) {
   return null;
}
```

This predicate evaluates to true:

- 3.3% of the time in correct runs
- 32.5% of the time in the bad runs
The ConfDiagnoser technique

Program

Configuration options

Configuration Propagation Analysis

affected predicates

//a configuration option
int maxsize = readFromCommandLine();
Sequence seq = createNewSequence();
...
if (seq.size() > maxsize) {
    return null;
}
affected predicate
The ConfDiagnoser technique

Program

Configuration options

Configuration Propagation Analysis

affected predicates

instrument

How often an affected predicate is evaluated
How often an affected predicate evaluates to true
The ConfDiagnoser technique

Program

Configuration options

Configuration Propagation Analysis

affected predicates

instrument

reproduce the error

A bad execution trace

Compare & Select

Correct Execution Traces

1. Convert a trace into a vector
2. Compute the cosine similarity between 2 vectors

a set of correct and similar execution traces
The ConfDiagnoser technique

1. Compare each predicate’s behavior between the bad and correct traces.
2. A metric for predicate’s behavior:

\[
\frac{1}{\text{exec frequency}} + \frac{1}{\text{true ratio}}
\]
The ConfDiagnoser technique

Program

Configuration options

- Configuration
- Propagation
- Analysis

A bad execution trace

Correct and similar trace

Behaviorally-deviated predicates

Compare & Select

Correct Execution Traces

Report
1. 
2. 
3. ...
Outline

• Example
• The ConfDiagnoser Technique
• Evaluation
• Related Work
• Contributions
Research questions

• How effective is ConfDiagnoser in diagnosing errors?
  – Diagnosis accuracy
  – Time cost
  – Comparison with three existing techniques
    • One configuration error diagnosis technique
    • Two general automated debugging techniques
**14 configuration errors from 5 subjects**

<table>
<thead>
<tr>
<th>Subject</th>
<th>LOC</th>
<th>#Options</th>
<th>#Non-crashing Errors</th>
<th>#Crashing Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Randoop</td>
<td>18587</td>
<td>57</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Weka</td>
<td>3810</td>
<td>14</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Synoptic</td>
<td>19153</td>
<td>37</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Soot</td>
<td>159271</td>
<td>49</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>JChord</td>
<td>23391</td>
<td>79</td>
<td>1</td>
<td>9</td>
</tr>
</tbody>
</table>

- Correct executions for each program
  - 6 – 16 examples from its user manual

*Collected from FAQ, forum posts, mailing list questions …
[Collect from Rabkin ASE’11]*
ConfDiagnoser’s accuracy and efficiency

- Measure accuracy by the absolute root cause ranking
- Time cost: 4 mins / error (on average)
Comparison with ConfAnalyzer [Rabkin ’11]

- The most recent configuration error diagnosis technique
  - Use dynamic tainting
  - Only supports crashing errors

Average rank
- ConfDiagnoser: 5th
- ConfAnalyzer: 12th

ConfDiagnoser produces:
- Better results on 8 errors
- Same results on 3 errors
- Worse results on 3 errors
Comparison with Tarantula [Jones ’03]

- Tarantula-based configuration debugging
  - Use statement coverage to localize suspicious statements
  - Use thin slicing to identify the affecting configuration options

Tarantula’s statement-level granularity is too fine-grained
- Many statements get the same suspiciousness value
- Statement coverage does not indicate predicate evaluation results

Average rank
- ConfDiagnoser: 5th
- Tarantula: 15th
Comparison with Invariant Analysis [McCamant ‘04]

- Invariant Analysis-based configuration debugging
  - Use method invariant difference to localize suspicious methods
  - Use thin slicing to identify the affecting configuration options

Invariant analysis’ method-level granularity is too coarse-grained
- Some control flow changes inside a method are not be reflected by invariants

Average rank
- ConfDiagnoser: 5th
- Invariant Analysis: 18th
Experimental conclusion

• ConfDiagnoser is **accurate** and **efficient**

• ConfDiagnoser **outperforms** existing techniques
  – One configuration error diagnosis technique
  – Two general automated debugging techniques
Outline

• Assumption, Goal, and Insight
• The ConfDiagnoser Technique
• Evaluation
• Related Work
• Contributions
Related work on configuration error diagnosis

- Tainting-based techniques
  - Dynamic tainting [Attariyan’08]
  - Static tainting [Rabkin’11]
    *Focuses exclusively on crashing errors*

- Search-based techniques
  - Delta debugging [Zeller’02], Chronus [Whitaker’04]
    *Requires a correct state for comparison, or OS-level support*

- Domain-specific techniques
  - PeerPressure [Wang’04]
  - RangeFixer [Xiong’12]
    *Targets a specific kind of configuration errors, and does not support a general language like Java*
Outline

- Assumption, Goal, and Insight
- The ConfDiagnoser Technique
- Evaluation
- Related Work
- Contributions
Contributions

• A technique to diagnose configuration errors
  *Compare relevant predicate behaviors between executions*
  – Fully automated
  – Can diagnose both crashing and non-crashing errors
  – Requires no OS-level support

• Experiments that demonstrate its usefulness
  – Accurate and fast
  – Outperforms three existing techniques

• The ConfDiagnoser tool implementation
  [http://config-errors.googlecode.com](http://config-errors.googlecode.com)
[Backup Slides]
Representation of configuration options inside ConfDiagnoser

• A configuration option is represented as a class field

• An example configuration option in Randoop:
  - `randoop.main.GenInputsAbstract.maxsize`

  
  ![Diagram]

  
  - Class name
  - Field name

• Made a 24-LOC syntactic change to 5 subject programs
  - Transform configuration option into class field