An Eclipse-based Integrated and Automated Fault Localization System

Tristan Challener

April 26, 2015

Table of Contents

Motivation

- Debugging is complex and difficult

Motivation

- Debugging is complex and difficult
- Fault localization is the most expensive

Motivation

- Debugging is complex and difficult
- Fault localization is the most expensive
- Current techniques insufficient

Automatic Fault Localization

- Uses per-test coverage analysis

Automatic Fault Localization

- Uses per-test coverage analysis
- Ranks statements by suspiciousness

Automatic Fault Localization

- Uses per-test coverage analysis
- Ranks statements by suspiciousness
- Variety of risk evaluation functions

$$suspiciousness(e) = 1 - \frac{\frac{failed(e)}{totalfailed}}{\frac{passed(e)}{totalpassed} + \frac{failed(e)}{totalfailed}}$$

EXAM Score

- One method for comparing risk evaluation functions

EXAM Score

- One method for comparing risk evaluation functions
- Ranking of faulty statement in suspiciousness

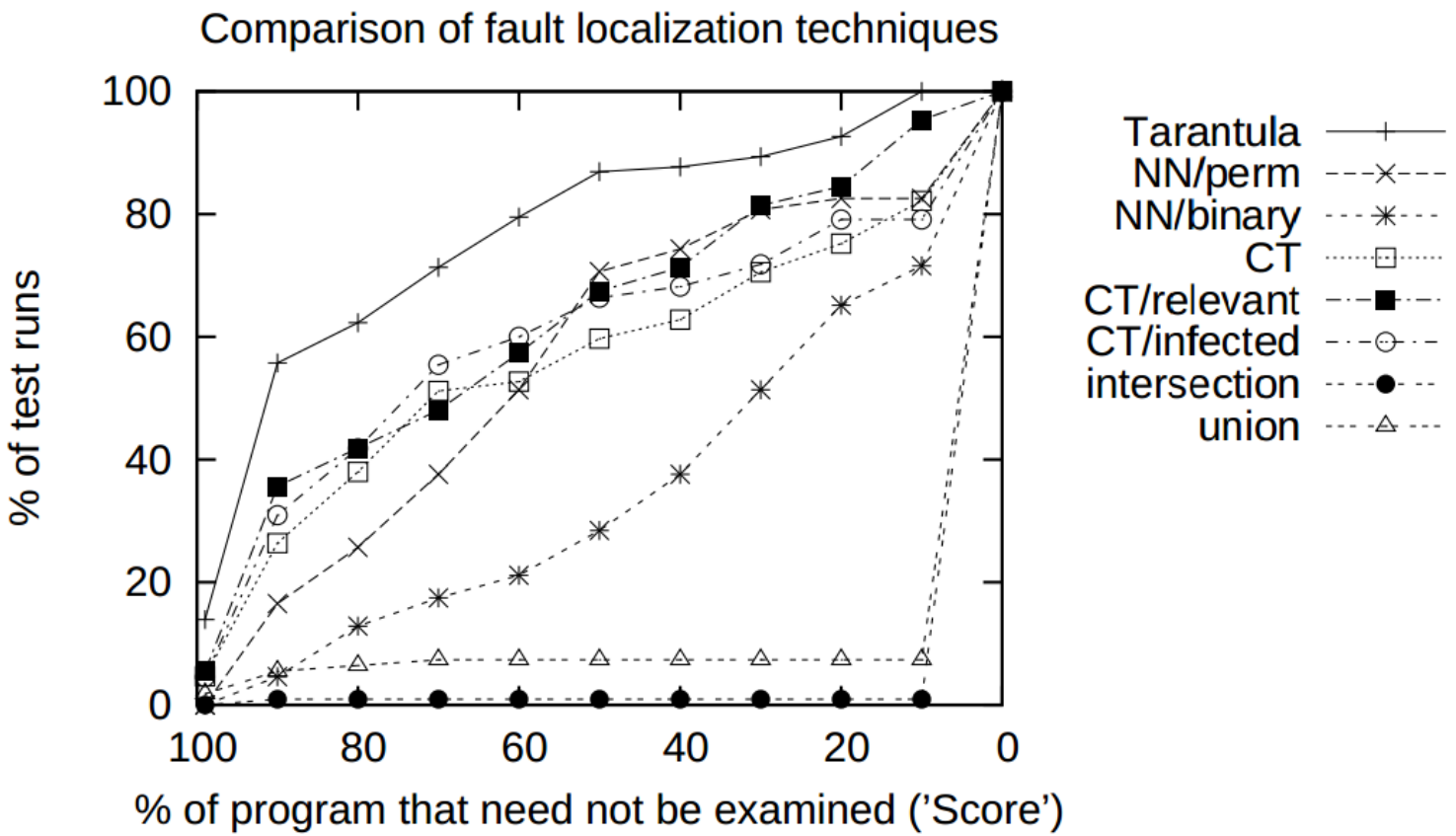
EXAM Score

- One method for comparing risk evaluation functions
- Ranking of faulty statement in suspiciousness
- Percentage of statements not needed to consider

EXAM Score

- One method for comparing risk evaluation functions
- Ranking of faulty statement in suspiciousness
- Percentage of statements not needed to consider
- Higher-is-better metric

EXAM Score


Comparison of fault localization techniques
% of test runs
100
80
60
40
20
0
100
80
60
40
20
0
% of program that need not be examined ('Score')
Tarantula
NN/perm
NN/binary
CT
CT/relevant
CT/infected
intersection
union

NCP Score

- Alternate method for comparing risk evaluation functions

NCP Score

- Alternate method for comparing risk evaluation functions
- Risk evaluation function becomes heuristic for GenProg

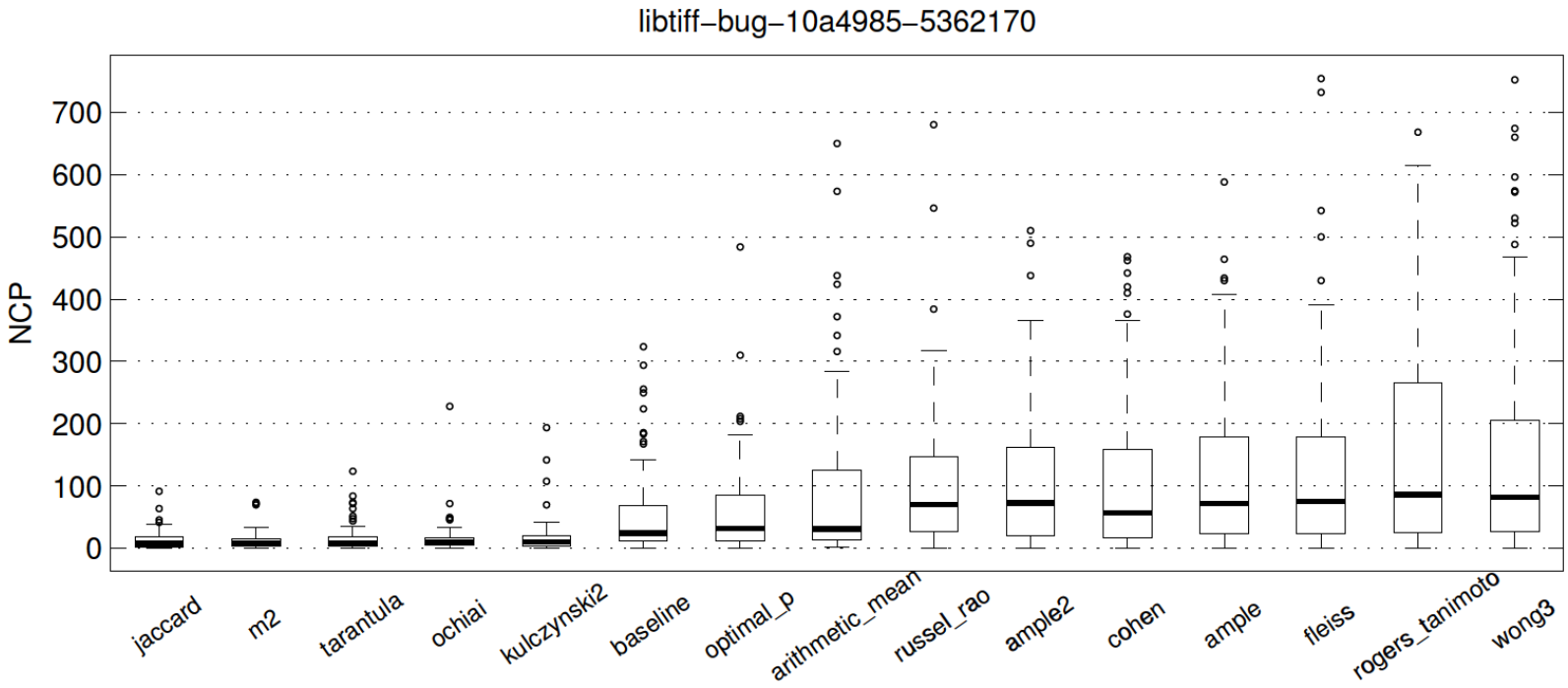
NCP Score

- Alternate method for comparing risk evaluation functions
- Risk evaluation function becomes heuristic for GenProg
- Number of patches before correct patch

NCP Score

- Alternate method for comparing risk evaluation functions
- Risk evaluation function becomes heuristic for GenProg
- Number of patches before correct patch
- Lower-is-better metric

NCP Score



Empirical Study

- Use EXAM score to compare risk evaluation functions

Empirical Study

- Use EXAM score to compare risk evaluation functions
- Select from existing studies

Empirical Study

- Use EXAM score to compare risk evaluation functions
- Select from existing studies
- GenProg(NCP) comparison

Empirical Study

- Use EXAM score to compare risk evaluation functions
- Select from existing studies
- GenProg(NCP) comparison
- Theoretical comparison

CodeCover Coverage

- Eclipse-compatible coverage analysis tool

CodeCover Coverage

- Eclipse-compatible coverage analysis tool
- Uses existing JUnit test suites

CodeCover Coverage

- Eclipse-compatible coverage analysis tool
- Uses existing JUnit test suites
- Generates coverage information for each test method

CodeCover Coverage

- Eclipse-compatible coverage analysis tool
- Uses existing JUnit test suites
- Generates coverage information for each test method
- Stores output in readable format (XML)

Using CodeCover

- Tested with a simple existing system with a JUnit test suite

Using CodeCover

- Tested with a simple existing system with a JUnit test suite
- Successfully produced per-test coverage

Using CodeCover

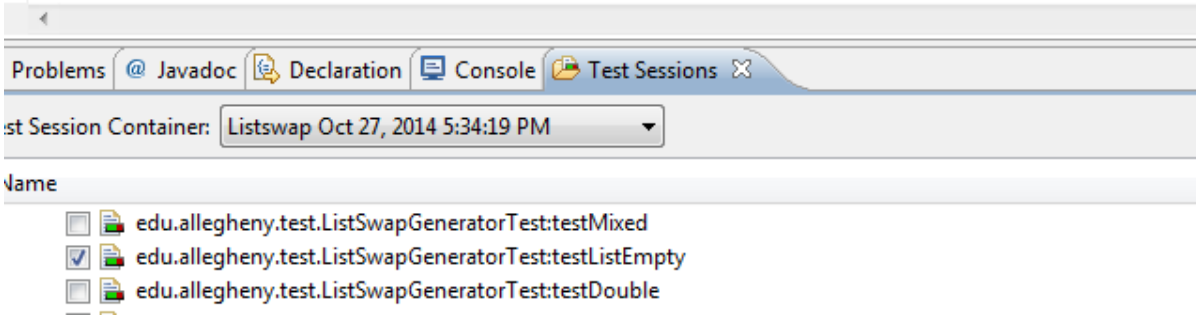
- Tested with a simple existing system with a JUnit test suite
- Successfully produced per-test coverage
- Produced XML representation of coverage data

Coverage Highlighting

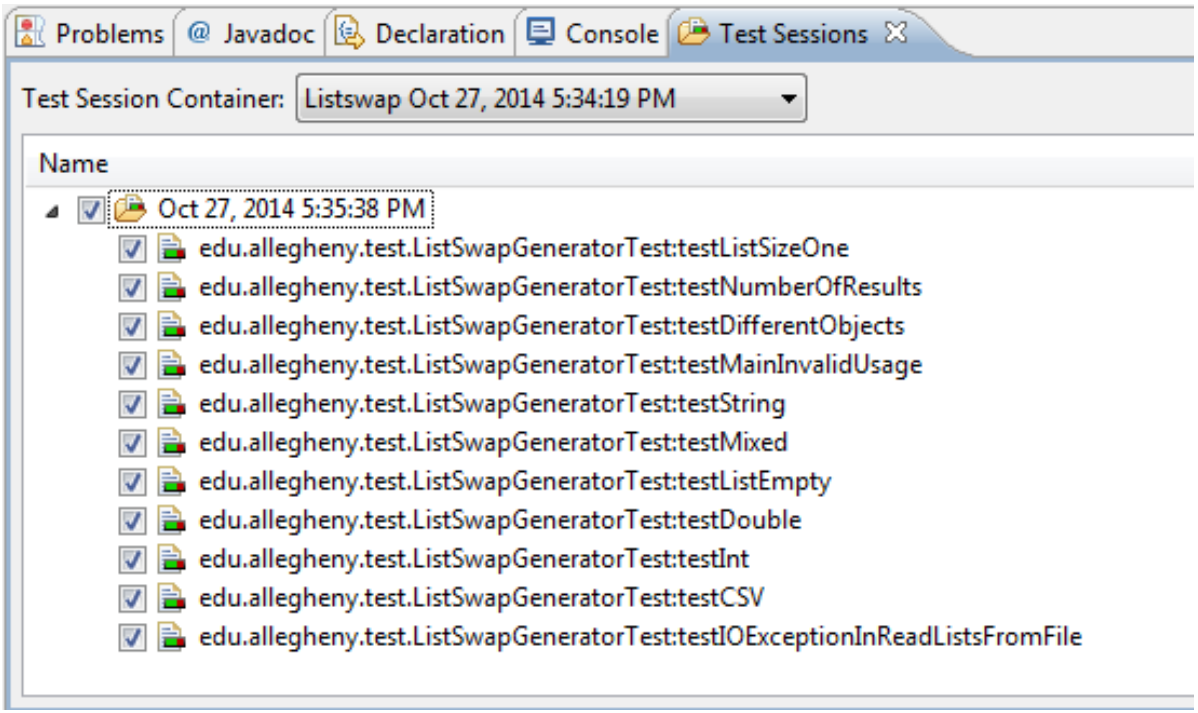
```
    public static String listsToString(ArrayList<ArrayList<Object>> list)
    {
        String ret = "";
        int listSize = list.size();

        for(int i = 0; i < listSize; i++)
        {
            ret += "L" + (i + 1) + ": \t" + list.get(i);
            if (i < listSize)
                ret += "\n";
        }

        return ret;
    }
}
```



Per-test Coverage



XML Representation

- Contains several pieces of information:

XML Representation

- Contains several pieces of information:
  - Complete source code
  - Statement definitions
  - List of statements covered by each test method for each file under test

XML Representation

- Contains several pieces of information:
  - Complete source code
  - Statement definitions
  - List of statements covered by each test method for each file under test

```
<BasicStmnt CovItemId="S4" CovItemPrefix="edu.allegheny.listswap.ListSwapGenerator
<LocList>
<Loc EndOffset="777" SrcFileId="1" StartOffset="735"/>
</LocList>
</BasicStmnt>
```

XML Representation

```
<TestCase Comment="" Date="14143657707 50"
    Name="edu.allegheny.test.ListSwapGeneratorTest:testString">
<CovList>
<CovPrefix CovItemPrefix="edu.allegheny.listswap.ListSwapGenerator.java">
<Cov CovItemId="B1" Value="5"/>
<Cov CovItemId="B2" Value="1"/>
<Cov CovItemId="L1-2" Value="1"/>
<Cov CovItemId="L2-0" Value="1"/>
<Cov CovItemId="L2-1" Value="1"/>
<Cov CovItemId="L2-2" Value="2"/>
<Cov CovItemId="L3-2" Value="1"/>
<Cov CovItemId="S1" Value="1"/>
<Cov CovItemId="S10" Value="1"/>
<Cov CovItemId="S11" Value="1"/>
<Cov CovItemId="S12" Value="1"/>
<Cov CovItemId="S13" Value="6"/>
<Cov CovItemId="S14" Value="6"/>
<Cov CovItemId="S15" Value="5"/>
<Cov CovItemId="S16" Value="1"/>
<Cov CovItemId="S2" Value="1"/>
<Cov CovItemId="S3" Value="4"/>
```

Parsing XML

- Document Object Model(DOM) parsing

Parsing XML

- Document Object Model(DOM) parsing
- Translate entire XML document into Java tree structure

Parsing XML

- Document Object Model(DOM) parsing
- Translate entire XML document into Java tree structure
- Allows ease of handling data once built

Parsing XML

```
ELEM: nodeName="BasicStmnt"
  ATTR: nodeName="CovItemId" nodeValue="S4"
   TEXT: nodeName="#text" nodeValue="S4"
  ATTR: nodeName="CovItemPrefix" nodeValue="edu.allegheny.listswap.ListSwapGenerator
   TEXT: nodeName="#text" nodeValue="edu.allegheny.listswap.ListSwapGenerator.java"
  ATTR: nodeName="Intrnl_Id" nodeValue="22"
   TEXT: nodeName="#text" nodeValue="22"
 TEXT: nodeName="#text" nodeValue=[WS]
 ELEM: nodeName="LocList"
  TEXT: nodeName="#text" nodeValue=[WS]
  ELEM: nodeName="Loc"
    ATTR: nodeName="EndOffset" nodeValue="777"
     TEXT: nodeName="#text" nodeValue="777"
    ATTR: nodeName="SrcFileId" nodeValue="1"
     TEXT: nodeName="#text" nodeValue="1"
    ATTR: nodeName="StartOffset" nodeValue="735"
     TEXT: nodeName="#text" nodeValue="735"
```

Parsing XML

```
 ATTR: nodeName="CovItemPrefix" nodeValue="edu.allegheny.listswap.ListSwapGenerator
  TEXT: nodeName="#text" nodeValue="edu.allegheny.listswap.ListSwapGenerator.java"
TEXT: nodeName="#text" nodeValue=[WS]
ELEM: nodeName="Cov"
  ATTR: nodeName="CovItemId" nodeValue="L1-0"
   TEXT: nodeName="#text" nodeValue="L1-0"
  ATTR: nodeName="Value" nodeValue="1"
   TEXT: nodeName="#text" nodeValue="1"
TEXT: nodeName="#text" nodeValue=[WS]
ELEM: nodeName="Cov"
  ATTR: nodeName="CovItemId" nodeValue="L3-0"
   TEXT: nodeName="#text" nodeValue="L3-0"
  ATTR: nodeName="Value" nodeValue="1"
   TEXT: nodeName="#text" nodeValue="1"
TEXT: nodeName="#text" nodeValue=[WS]
```

Conclusion

- Create an Eclipse plugin that uses existing tools

Conclusion

- Create an Eclipse plugin that uses existing tools
- Combine existing tools in a single fault localization system

Conclusion

- Create an Eclipse plugin that uses existing tools
- Combine existing tools in a single fault localization system
- Ambitious, but feasible

References

James A. Jones and Mary Jean Harrold.

Empirical evaluation of the tarantula automatic fault-localization technique.
In *Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering*, ASE '05, pages 273–282, New York, NY, USA, 2005. ACM.

Yuhua Qi, Xiaoguang Mao, Yan Lei, and Chengsong Wang.

Using automated program repair for evaluating the effectiveness of fault localization techniques.
In *Proceedings of the 2013 International Symposium on Software Testing and Analysis*, ISSTA 2013, pages 191–201, New York, NY, USA, 2013. ACM.

Xiaoyuan Xie, Tsong Yueh Chen, Fei-Ching Kuo, and Baowen Xu.

A theoretical analysis of the risk evaluation formulas for spectrum-based fault localization.
ACM Trans. Softw. Eng. Methodol., 22(4):31:1–31:40, October 2013.