Do Automated Program Repair Techniques Repair Hard and Important Bugs?

Manish Motwani
Sandhya Sankararayanan  René Just  Yuriy Brun

University of Massachusetts Amherst
Automatic Program Repair: An Active Research Area

Is the bug important to fix?

Is the patched program correct?

Automated program repair publications per year [1]

Automatic Program Repair: An Active Research Area

Is the bug important to fix?

Is the patched program correct?

Automated program repair publications per year [1]

Automatic Program Repair: An Active Research Area

Is the bug important to fix?

Is the patched program correct?

Automated program repair publications per year [1]

Automatic Program Repair: An Active Research Area

Is the bug important to fix?

Is the bug hard to fix?

Is the patched program correct?

Automated program repair publications per year [1]

Motivation

Prior evaluations of automated repair have focused on:

- Fraction of defects repaired [1,2]
- Computational resources required to repair defects [3,4]
- Correctness and quality of generated patches [5,6,7]
- Patch maintainability [8]
- Repair acceptability [9,10]

## Motivation

<table>
<thead>
<tr>
<th>YetAnotherFix</th>
<th>ThisNeverEndsFix</th>
</tr>
</thead>
<tbody>
<tr>
<td>fixes 60% of the defects</td>
<td>fixes 30% of the defects</td>
</tr>
<tr>
<td>Defect-1 patched</td>
<td>Defect-1 not patched</td>
</tr>
<tr>
<td>Defect-2 patched</td>
<td>Defect-2 not patched</td>
</tr>
<tr>
<td>Defect-3 not patched</td>
<td>Defect-3 patched</td>
</tr>
<tr>
<td>Defect-4 patched</td>
<td>Defect-4 not patched</td>
</tr>
<tr>
<td>Defect-5 patched</td>
<td>Defect-5 not patched</td>
</tr>
<tr>
<td>Defect-6 not patched</td>
<td>Defect-6 not patched</td>
</tr>
<tr>
<td>Defect-7 patched</td>
<td>Defect-7 not patched</td>
</tr>
<tr>
<td>Defect-8 patched</td>
<td>Defect-8 not patched</td>
</tr>
<tr>
<td>Defect-9 not patched</td>
<td>Defect-9 patched</td>
</tr>
<tr>
<td>Defect-10 not patched</td>
<td>Defect-10 patched</td>
</tr>
</tbody>
</table>

Which automated program repair technique is better?
Motivation

<table>
<thead>
<tr>
<th>Defect</th>
<th>YetAnotherFix</th>
<th>ThisNeverEndsFix</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>patched</td>
<td>not patched</td>
</tr>
<tr>
<td>Defect-3</td>
<td><strong>not patched</strong></td>
<td>patched</td>
</tr>
<tr>
<td>Defect-4</td>
<td>patched</td>
<td>not patched</td>
</tr>
<tr>
<td>Defect-5</td>
<td>patched</td>
<td>not patched</td>
</tr>
<tr>
<td>Defect-6</td>
<td><strong>not patched</strong></td>
<td>not patched</td>
</tr>
<tr>
<td>Defect-7</td>
<td>patched</td>
<td>not patched</td>
</tr>
<tr>
<td>Defect-8</td>
<td>patched</td>
<td>not patched</td>
</tr>
<tr>
<td>Defect-9</td>
<td><strong>not patched</strong></td>
<td>patched</td>
</tr>
<tr>
<td>Defect-10</td>
<td><strong>not patched</strong></td>
<td>patched</td>
</tr>
</tbody>
</table>

Hard to fix defects

Which automated program repair technique is better?

How about now?
Which is harder to fix?

Invalid error message

Bug #53898 PHAR reports invalid error message, when the directory does not exist
Submitted: 2011-02-01 09:55 UTC  Modified: 2011-02-01 13:01 UTC

From: nightstorm at tien dot pl
Status: Closed
PHP Version: 5.3.5
Private report: No

[2011-02-01 09:55 UTC] nightstorm at tien dot pl

Description:

When we try to create a PHAR archive, and the directory, where we want to save it does not exist, PHAR complains about invalid file extension, instead of informing that there is a problem with accessing the requested location. This is a bit confusing.
Which is harder to fix?

Invalid error message

Invalid memory access
(Application crash)
Which is harder to fix? Which is more important to fix?

Invalid error message

Invalid memory access
(Application crash)
Which is harder to fix? Which is more important to fix?

Invalid error message
Invalid memory access
(Application crash)

Easy and less important
Hard and more important

How do we measure hardness and importance of a defect?
Goals of this study

A methodology for measuring a defect’s hardness and importance.

An evaluation of whether automated program repair techniques repair hard and important defects.
Measuring hardness and importance of a defect
Measuring hardness and importance of a defect

bug report

Developer-written patch

- Priority
- Versions
- Time to fix
Measuring hardness and importance of a defect

Developer-written patch

Test-suite

bug report

Priority

Versions

Time to fix

adds new variable

adds new loop

file count

adds new conditional

deletes conditional

line count

Failing tests

Relevant Tests
Measuring hardness and importance of a defect

bug report

Developer-written patch

Test-suite

Priority

Versions

Time to fix

Other parameters may also exist.
Measuring hardness and importance of a defect

Analyzed 8 popular bug-tracking systems
- IBM Rational ClearQuest
- HP Quality Center
- Mantis Bug Tracker
- Redmine
- JIRA
- Bugzilla
- FogBugz
- Trac

Analyzed 3 popular open-source code repositories
- GitHub
- SourceForge
- Google Code

Analyzed 2 defect benchmarks
- Defects4J
- ManyBugs
Measuring hardness and importance of a defect

5 defect characteristics defined in terms of 11 abstract parameters

<table>
<thead>
<tr>
<th>Defect Importance</th>
<th>Defect Complexity</th>
<th>Test Effectiveness</th>
<th>Defect Independence</th>
<th>Developer-written patch characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority</td>
<td>File count</td>
<td>Failing test count</td>
<td>Dependents count</td>
<td>Patch modification type</td>
</tr>
<tr>
<td>Time to Fix</td>
<td>Line count</td>
<td>Relevant test count</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Versions</td>
<td>Reproducibility</td>
<td>Test suite coverage</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Diagram showing the relationships between defect characteristics and developer-written patch characteristics.](chart.png)
Evaluating repair techniques along new dimensions

- 2 defect benchmarks: Defects4J and ManyBugs
- Semi-automatically annotated 409 defects with:
  - 5 defects characteristics defined using 11 abstract parameters.
Evaluating repair techniques along new dimensions

- 2 defect benchmarks: Defects4J and ManyBugs
- Semi-automatically annotated 409 defects with:
  - 5 defects characteristics defined using 11 abstract parameters.
  - Existing repairability and repair quality results of 7 automated repair techniques.
Evaluating repair techniques along new dimensions

- 2 defect benchmarks: Defects4J and ManyBugs
- Semi-automatically annotated 409 defects with:
  - 5 defects characteristics defined using 11 abstract parameters.
  - Existing repairability and repair quality results of 7 automated repair techniques.
- Identify if repairability of a repair technique correlates (Somer’s Delta $\in [-1, 1]$) with each abstract parameter.
Do repair techniques repair important defects?

Java repair techniques are more likely to repair defects that are important for developers.
Do repair techniques repair hard defects?

C repair techniques are less likely to repair defects that required developers to write more code.
Do repair techniques repair defects with **effective test suites**?

Java repair techniques are less likely to repair defects with effective test suites.
What patch modification types are challenging for automated repair?

9 Patch modification types [1]

- Adds one or more if statements
- Adds one or more loops
- Adds one or more new variables
- Changes one or more conditionals
- Adds one or more method calls
- Changes one or more method signatures
- Changes one or more method arguments
- Adds one or more new methods
- Changes one or more data structures or types

Defects that required developers to add loops or a new method call, or change a method signature are challenging for automated repair techniques to patch.

What about correct patches?

Only Prophet (15) and SPR (13) generate sufficient number of correct patches.
What about correct patches?

Prophet is less likely to produce patches for more complex defects, and even less likely to produce correct patches for the same defects.
What about correct patches?

Prophet is less likely to produce patches for more complex defects, and even less likely to produce correct patches for the same defects.
Contributions

Methodology to measure importance and hardness of a defect.

5 defect characteristics defined in terms of 11 abstract parameters.
Contributions

Methodology to measure importance and hardness of a defect.

Methodology to evaluate automated program repair techniques along new dimensions.

How do we define hardness and importance of a defect?

5 defect characteristics defined in terms of 11 abstract parameters:

- Priority
- Time to Fix
- Versions
- File count
- Line count
- Reproducibility
- Failing test count
- Relevant test count
- Test suite coverage
- Dependents count
- Patch modification type

Defect Importance

Defect Complexity

Test Effectiveness

Defect Independence

Developer-written patch characteristics

How do we evaluate automated program repair techniques along new dimensions?

- 2 defect benchmarks: Defects4J and ManyBugs
- Annotated 409 defects with:
  - 5 defects characteristics defined using 11 abstract parameters.
  - Existing repairability and repair quality results of 7 automated repair techniques.
- Identify if repairability of a repair technique correlates (Somer's Delta ∈ [−1, 1]) with each abstract parameter.
Contributions

Methodology to measure importance and hardness of a defect.

Methodology to evaluate automated program repair techniques along new dimensions.

Evaluation of 7 automated program repair techniques on 409 real-world defects.

How do we define hardness and importance of a defect?

5 defect characteristics defined in terms of 11 abstract parameters.

- Priority
- Time to Fix
- Versions
- File count
- Line count
- Reproducibility
- Failing test count
- Relevant test count
- Test suite coverage
- Dependents count
- Patch modification type

Defect Importance
Defect Complexity
Test Effectiveness
Defect Independence
Developer-written patch characteristics

▶ 2 defect benchmarks: Defects4J and ManyBugs
▶ Annotated 409 defects with:
  - 5 defects characteristics defined using 11 abstract parameters.
  - Existing repairability and repair quality results of 7 automated repair techniques.
  - Identify if repairability of a repair technique correlates (Somer's Delta ∈ [−1, 1]) with each abstract parameter.

Repair techniques produce too few correct patches.
Our analysis uses all patches instead, as an estimate.
Recommendations

Repair research should evaluate if new techniques repair hard and important defects.

- Is the bug important to fix?
- Is the bug hard to fix?
- Is the patched program correct?

![Diagram showing a process flow from buggy program to patched program through APR (Automatic Program Repair) with test suite evaluation].

- Defects4J (224 defects)
- ManyBugs (185 defects)

- Importance Complexity Test Effectiveness Independence Patch Characteristics

- ▶ 2 defect benchmarks: Defects4J and ManyBugs
- ▶ Annotated 409 defects with:
  - ▶ 5 defects characteristics defined using 11 abstract parameters.
  - ▶ Existing repairability and repair quality results of 7 automated repair techniques.
  - ▶ Identify if repairability of a repair technique correlates (Somer's Delta ∈ [−1, 1]) with each abstract parameter.
  - ▶ Some defects were not linked to bug reports.
  - ▶ Repair techniques produce too few correct patches.
  - Our analysis uses all patches instead, as an estimate.

- ![Table showing repair results for YetAnotherFix and ThisNeverEndsFix].

Annotated datasets and scripts are available at [https://github.com/LASER-UMASS/AutomatedRepairApplicabilityData](https://github.com/LASER-UMASS/AutomatedRepairApplicabilityData)
Recommendations

Repair research should evaluate if new techniques repair hard and important defects.

- **Is the bug important to fix?**
- **Is the bug hard to fix?**
- **Is the patched program correct?**
- **Which automated program repair technique is better?**

Repair research should target defects that existing techniques have missed.

- **9 Patch modification types** [1]
  - adds one or more if statements
  - adds one or more loops
  - adds one or more new variables
  - changes one or more conditionals
  - adds one or more method calls
  - changes one or more method arguments
  - changes one or more data structures or types
  - adds one or more new methods
  - changes one or more method signatures

Defects that required developers to add loops or a new method call, or change a method signature are challenging for automated repair techniques to patch.

- **Defects** [1]
  - Defects4J
  - ManyBugs
  - (224 defects)
  - (185 defects)

- **Importance Complexity Test Effectiveness Independence Patch Characteristics**

Our analysis uses all patches instead, as an estimate.

Annotated datasets and scripts are available at [https://github.com/LASER-UMASS/AutomatedRepairApplicabilityData](https://github.com/LASER-UMASS/AutomatedRepairApplicabilityData)


Recommendations

Repair research should evaluate if new techniques repair hard and important defects.

Evaluation benchmarks need to account for diversity of defect complexity, importance, etc.

What patch modification types are associated with an automated repair technique’s repairability?

- Importance
- Complexity
- Test Effectiveness
- Independence
- Patch Characteristics

Defects that required developers to add loops or a new method call, or change a method signature are challenging for automated repair techniques to patch.

Defects that required developers to add one or more new variables or changed one or more conditionals are challenging for automated repair techniques to patch.

Annotated datasets and scripts are available at

https://github.com/LASER-UMASS/AutomatedRepairApplicabilityData

http://people.cs.umass.edu/~mmotwani/
Evaluation Methodology

Somers’ Delta
What is the strength of association?

Mann-Whitney U Test
Are the two populations Patched Vs. Unpatched significantly different?

Independent Variable
- Abstract parameter

Dependent Variable
- Repairability

Correlation Coeff (r), 95% CI
- p-value