Quality Assurance of a Mobile Network Measurement Testbed Through Systematic Software Testing

Utkarsh Goel, James Espeland, Upulee Kanewala, and Mike P. Wittie

SEDE 2016
Motivation

- Application developers care about user experience.
- A good user experience attracts users and increases profits.
- Decreases the amount of technical support.
Ways to Improve User Experience

- **Usefulness** of the application - offers a service that the user wants.

- **Good looking** user interface - design is **attractive**.

- **Easy to use** for the end-user - navigation is **not** complicated.

- **Reliability** of the application - works correctly **all the time**.
  - Hard to ensure
  - Time consuming
  - Requires lots of money

**DEVELOPERS NEED HIGH INCENTIVES FOR TESTING RELIABILITY**
What happens when code is run without testing?
Code Reliability is Important

- Developers need to **thoroughly test** their applications.
- Testing makes applications **more reliable**.
- Testing **minimizes unexpected** results.
- A reliable application increases the user-experience.
MITATE
(Mobile Internet Testbed for Application Traffic Experimentation)

● Allows mobile application prototyping in mobile networks.

● Developers can simulate Web, video, gaming application traffic.

● MITATE measures application performance in different networks.

● Allows developers to customize traffic to ensure good user-experience in a given network.

How MITATE works

1. Exp. configuration
2. Query for new experiments
3. Experiment traffic
4. Traffic metrics
5. Exp. result data
MITATE Source Code

- MITATE consists of > 9600 lines of code.

- Four components
  - Mobile application - written in Java (Android)
  - Measurement Servers - written in Java
  - Command line API to interact with MITATE’s backend servers - Shell
  - Web server - written in PHP

- Two important parts of MITATE on which to focus our testing efforts
  - Command line interface.
  - PHP code that validates the XML-based experiment configuration and uploads it to the database.
Command Line API - Source Code

- Consists of ~ 230 lines of code.
  - 27 “if statements”
  - Two “for loops”
  - One “while loop”
  - Five “function calls”
Black-box Testing of Shell Script

mitate.sh <option>
- login
- init <output_filename>
- validate <XML_Filename>
- getExpCost <XML_Filename>
- checkAvailableCredits
- upload <XML_Filename>
- makePublic <experiment_ID>
- getExpStatus <experiment_ID>
- query <user_experiment_list.txt> <output_filename>
- delete <experiment_ID>
- logout

<table>
<thead>
<tr>
<th>Number of Arguments</th>
<th>Partition Number</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>P1</td>
<td>No arguments.</td>
</tr>
<tr>
<td>1</td>
<td>P2</td>
<td>Argument is valid.</td>
</tr>
<tr>
<td></td>
<td>P3</td>
<td>Argument is invalid.</td>
</tr>
<tr>
<td>2</td>
<td>P4</td>
<td>Both arguments are valid.</td>
</tr>
<tr>
<td></td>
<td>P5</td>
<td>First argument is invalid.</td>
</tr>
<tr>
<td></td>
<td>P6</td>
<td>First argument is valid, the second is invalid.</td>
</tr>
<tr>
<td></td>
<td>P7</td>
<td>First argument is valid, the second is empty.</td>
</tr>
<tr>
<td>3</td>
<td>P8</td>
<td>All arguments are valid.</td>
</tr>
<tr>
<td></td>
<td>P9</td>
<td>First and second arguments are valid and the third argument is invalid.</td>
</tr>
<tr>
<td></td>
<td>P10</td>
<td>First and third arguments are valid and the second argument is invalid.</td>
</tr>
<tr>
<td>&gt;3</td>
<td>P11</td>
<td>First, second, and third arguments are valid.</td>
</tr>
</tbody>
</table>
Black-box Testing of Shell Script - Results

- Output for partitions P1 through P10 matched the expected output.

- An extraneous argument as presented by partition P11 resulted in no error indication to the user.

- We fixed the code to check for extraneous arguments in the Shell script.

- We repeated the P11 test which resulted in the expected output.

<table>
<thead>
<tr>
<th>Number of Arguments</th>
<th>Partition Number</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>P1</td>
<td>No arguments.</td>
</tr>
<tr>
<td>1</td>
<td>P2</td>
<td>Argument is valid.</td>
</tr>
<tr>
<td></td>
<td>P3</td>
<td>Argument is invalid.</td>
</tr>
<tr>
<td>2</td>
<td>P4</td>
<td>Both arguments are valid.</td>
</tr>
<tr>
<td></td>
<td>P5</td>
<td>First argument is invalid.</td>
</tr>
<tr>
<td></td>
<td>P6</td>
<td>First argument is valid, the second is invalid.</td>
</tr>
<tr>
<td></td>
<td>P7</td>
<td>First argument is valid, the second is empty.</td>
</tr>
<tr>
<td>3</td>
<td>P8</td>
<td>All arguments are valid.</td>
</tr>
<tr>
<td></td>
<td>P9</td>
<td>First and second arguments are valid and the third argument is invalid.</td>
</tr>
<tr>
<td></td>
<td>P10</td>
<td>First and third arguments are valid and the second argument is invalid.</td>
</tr>
<tr>
<td>&gt;3</td>
<td>P11</td>
<td>First, second, and third arguments are valid.</td>
</tr>
</tbody>
</table>
Mutation Testing of Shell Script

- Generated 15 mutants in the Shell using mutant-clustering as follows:
  - Un-nested conditional blocks
  - Nested conditional blocks
  - Multi-nested conditional blocks

- Killed 14 of the 15 mutants
  - Mutation Score of 93.3%

- Surviving mutant was a functionally equivalent mutation.
PHP – Source Code

- Consists of ~ 250 lines of code.
  - 39 “if statements”
  - Five “for loops”
  - Five “while loops”
  - Three “function calls”
Black-box Testing of PHP Code

<table>
<thead>
<tr>
<th>Partition Number</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Empty XML file.</td>
</tr>
<tr>
<td>P2</td>
<td>Valid XML file.</td>
</tr>
<tr>
<td>P3</td>
<td>XML file with invalid tags.</td>
</tr>
<tr>
<td>P4</td>
<td>XML file where integer values are replaced with character values.</td>
</tr>
<tr>
<td>P5</td>
<td>XML file where character values are replaced with integer values.</td>
</tr>
<tr>
<td>P6</td>
<td>XML file with extra tags.</td>
</tr>
</tbody>
</table>

- Results:
  - PHP code did **not throw human-readable errors** for some test cases.
  - We **modified PHP code** to throw human-readable error messages.
  - Repeated testing with new PHP code and **results were correct**.
Mutation Testing of PHP Code

- Generated **44 mutants covering all** of the conditionals.
- **Killed 39 of the mutants** verified by inspecting the Shell output and the database.
  - Mutation Score of 88.6%
- Two surviving mutants were due to **misspelled variable names**.
- Three surviving mutants were due to **code logic**.
- We **corrected** the misspelled variables and **refactored** some of the code logic.
- Retested: **Mutation Score of 100%**

**Before:**

```php
if (a) {
    // Do this
}
if (!a) {
    // Do that
}
```

**After:**

```php
if (a) {
    // Do this
}
else {
    // Do that
}
```
Some Takeaways

- Mutation testing is usually used to validate the test suite, but we also uncovered minor bugs during mutation testing.
- Use of multiple software testing techniques results in more robust testing.
- MITATE is more reliable now.
Thank you

Data and code available at:
https://github.com/ugoel/CSCI591_SoftwareTesting
Background Slides