What is QA in embedded software?

The Practical Industrial Experiences

Dr Jason Lee

Nov 4, 2016
Outline

- Background
  - Research Background & Industrial Background
- How QA works in Embedded Software
- Blackbox vs Whitebox
- Types of Testing
- Exploratory Testing Overview
- Emerging Trends of DevOps
- How QA fits into DevOps
- Career as QA
- Conclusion + Q & A
Background

- Senior Test Lead at Dolby
- Over 11+ years working experience in software testing
- Active researcher in software bug localization
- Adjunct Lecturer UWS and MQU
- ACM SEN Associate Editor

Abstract

As software continues to evolve in recent years, software testing is becoming more important and widely researched in the software engineering community. More challenges have been posed among the researchers and practitioners in the area of software bug localization to locate bugs automatically. This tutorial describes the state of the art of automated bug localization using dynamic analysis. It is vitally and relevant especially in the automated software engineering community. It will help to raise the understanding on the calibre of software bugs and how the automated bug localization approaches help reduce the prevalence of software bugs.
Research Background – (I of III)

• PhD in software testing

• Program spectra using dynamic analysis

• Aid programmers in debugging program – locating bugs
### Research Background – (II of III)

- Proposed spectra metrics in locating single bugs optimally

```c
mid() {
  int x, y, z, m;

  1: read("Enter 3 numbers:\", x, y, z);
  2: m = z;
  3: if (y<z)
  4:   if (x<y)
  5:     m = y;
  6:   else if (x<z)
  7:     m = y; // *** bug ***
  8:   else
  9:     if (x>y)
 10:    m = y;
 11:   else if (x>z)
 12:    m = x;
 13: print("Middle number is:\", m);
}
```

<table>
<thead>
<tr>
<th>Test Cases</th>
<th>3.3.5</th>
<th>1.2.3</th>
<th>3.2.1</th>
<th>5.5.5</th>
<th>5.3.4</th>
<th>2.1.3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>0.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pass/Fail Status</th>
<th>P</th>
<th>P</th>
<th>P</th>
<th>P</th>
<th>F</th>
</tr>
</thead>
</table>

**Figure 1: Example of Tarantula technique.**
Research Background – (III of III)

- Proposed using different test coverage information in locating bugs

- Proposed using different algorithms in giving weights to importance of test cases
Industrial Background (I of II)

- Embedded software – walkie talkie, modem, mobile phones, tablets
- Building automation tools, test strategy and planning
- Delivering embedded audio technologies in System On Chip (SOC)
Industrial Background (II of II)

• Working on Android AOSP and iOS app

• Teams focus on different market segments – PC, Mobile, Gaming
How QA Works in Embedded Software

- Drivers and stubs invoke APIs
- Customer use cases
- Machine language scripting
QA Processes in Embedded Software Industries

Validation & Verification

Find issues and anomalies

Logging issues and bugs

Prioritize with DEV and PO

Retest fixed bug(s)
Blackbox vs Whitebox

- Blackbox - Testing, either functional or non-functional, without reference to the internal structure of the component or system

Source: https://devopsbootcamp.osuosl.org/_images/black-box.png
Blackbox vs Whitebox

- Whitebox - Testing based on an analysis of the internal structure of the component or system

Types of Testing

Manual
• Basic UI
• Localization test
• Audio tools
• Listening
• Demo application

Automated
• API level testing
• Audio tools
Exploratory Testing Overview
Why Exploratory Testing?

IMPROVE QUALITY OF SOFTWARE
Process Flow of Exploratory Testing
Exploratory Session

• Where to Use?

• When to Use?

• How it Fits?
How Do We Explore?

Preparation  Execution  Reporting
Preparation

• Come up with a list of features to explore in the test session

• For every feature, select a subset of the use cases related to feature

• For every use case, come up with a set of use case scenarios

• Document them in an exploratory test session document
Exploratory (Mission-based) Session Template

Project name - Session number

Name/s: Name of tester/s

Aim: Outline features that will be explored in this session.

Abstract: High level description that expands on your aim.

Session time: Time per session

Test Environment: This section is used to outline all the necessary hardware and software components used for the execution of this test session e.g. Device, OS, Build and version number etc.

Use cases and use case scenarios: List use cases and their corresponding use case scenarios that you plan on executing in this session. In this section I also like to categorize them into their different features they’re exercising.

Test notes: (a tabular form of entering test notes throughout the test execution phase):

<table>
<thead>
<tr>
<th>Feature</th>
<th>Use Cases</th>
<th>Use Case Scenarios (selected from above)</th>
<th>Outcome (fill out during test execution)</th>
<th>JIRA Ticket/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature</td>
<td>Use Case #1 Use Case #2 ... Use Case #n</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feature</td>
<td>Use Case #1 Use Case #2 ... Use Case #n</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feature</td>
<td>Use Case #1 Use Case #2 ... Use Case #n</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Execution

- Execute in time boxed session

- Collaborate with other members (DEV, QA, PM, UI designer etc.)

- Capture as part of the Agile user stories
Example – Project K (I of III)

- Project K – Session 1
  - Aim: Explore the input and output file to the executables
  - Abstract: Explore around this requirement as this is an initial development build released by DEV
  - Session Time: 1 x 35 minutes session
  - Test Environment: x86 machine

- Use Case Scenario:

#1: I would like to input file to the executable and expect the output file
## Example – Project K (II of III)

**Test Notes Use Case Scenario:**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Use Case Scenarios</th>
<th>Outcome</th>
<th>JIRA Ticket(s)</th>
</tr>
</thead>
</table>
| Input to the executable          | 1. I want to input a file to the executable and expect output                      | 1. Output file output as expected and audible.  
2. Output file output only for the first input file but not for the others, no warnings produced.  
3. Not useful warning produced  
4. Not useful warning produced   | 1. N/A  
2. K-668  
3. K-692  
4. Same as #3                       |
|                                  | 2. I want to input multiple files to the executables and expect output             |                                                                                       |                                    |
|                                  | 3. I want to input missing input file to the executables                            |                                                                                       |                                    |
|                                  | 4. I want to input file without extension format                                   |                                                                                       |                                    |
Example – Project K (III of III)

• Future Test Items:
  • Retest the bug fixes for the above scenarios
  • Expand exploratory test session around this area

• Action Items:
  • Log Bugs found in session and link to tickets – Done
  • Add test cases to TestLink for the above cases – To Do
Reporting

• Debriefing phase

• Discussion, reporting the results and findings of the test session.

• This may include reporting:
  – use case scenarios that passed/failed
  – bugs found
  – alternative use case scenarios (future exploration ideas)
  – alternative testing approaches (alternative test setup).
Emerging Trends of DevOps
What is DevOps?

- Practice of operations and development engineers participating together in the entire service lifecycle, from design through the development process to production support.
Background of DevOps

• Aligned with Agile and lean approaches

• Emerged from effort businesses – respond to rapid changes

• Rapid movement and spreading throughout different organizations
Where DevOps Sits In?
DevOps – The Benefits
Practical DevOps Tools

- Git (GitHub)
- Dockers
- Jenkins
- Splunk
What is Continuous Integration?
What & How Continuous Integration Fits

• Development practice requires developers integrate code into a shared repository several times a day

• Each check in verified by an automated build
  - allow to detect problems early

• More than a process
Continuous Integration - Steps

- DEV checkout code
- Commit changes to repo
- Monitors the repo; checks out changes when they occur
- Builds system and runs tests
- Releases artefacts testing
- Informs team of build status
- Assigns build label to built code version
Jenkins – Build Information

Source: https://www.mirantis.com/blog/setting-external-openstack-testing-system-part-1/
Jenkins – Test Results Reports

TestNG Results

0 failures

Failed Tests
No Test method failed

Failed Configuration Methods
No Configuration method failed

Skipped Tests
No Test method was skipped

Skipped Configuration Methods
No Configuration method was skipped

All Tests (grouped by their packages)

<table>
<thead>
<tr>
<th>Package</th>
<th>Duration</th>
<th>Fail</th>
<th>(diff)</th>
<th>Skip</th>
<th>(diff)</th>
<th>Total</th>
<th>(diff)</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.miquro.bootstrapping.atm</td>
<td>149 msec</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

http://www.michael-noll.com/blog/2013/01/25/bootstrapping-a-java-project-with-gradle/
SDLCs

- Waterfall

- Agile

- Agile with Continuous Deploy
How QA Fits into DevOps

- Provide the ‘bridge’ to the Ops as customer advocates issues found
- Found issues and resolve effectively
- Velocity of feedback increases through iterative changes
- Automated tests reduces sizes of testing effort
Career as QA

• Learning curve in different domains

• Curious, inquisitive, think out of box

• Detail-oriented and problem solving skills

• Able to articulate the problems/issues found and possess strong communication skills
Recap

• Background
  – Research Background & Industrial Background

• How QA works in Embedded Software

• Blackbox vs Whitebox

• Types of Testing

• Exploratory Testing Overview

• Emerging Trends of DevOps

• How QA fits into DevOps

• Career as QA
Q&A

MiFavor 3.0
It has music settings for a variety of music styles