Testing GUI-based Software With Undetermined Input Spaces

Dissertation Defense
Bao N. Nguyen
October 2\textsuperscript{nd}, 2013

http://www.cs.umd.edu/~baonn/defense.pptx
GUI Input Space and GUI Testing

• GUI input space
  – Space of all possible event sequences

• GUI testing
  – Sampling on GUI input space

• Automation techniques
  – Capture/Replay
  – Script-based
  – Random-walk
Model-based Testing

• Formalize the input space as a graph (a.k.a. model)

• **Test generation = Searching**
  – **Systematically** identify all GUI elements to test
  – **Automatically** generate test cases by graph search

• **Examples:**
  – Finite State Machines [White, Belli]
  – UML diagrams [Paiva]
  – Event-flow Graph [Memon]
Event-flow Graph (EFG)

- A directed graph with
  - Nodes: all events on the GUI
  - Edges: {	extit{follows}} relationships
- The {	extit{GUI Ripping}} algorithm
  - One-pass depth-first GUI traversal
  - Obtain a GUI tree and analyze the \textit{follows} relationships
GUI Ripping Illustration
The Problem: Incomplete Input Spaces

- Contemporary GUIs are *Context-sensitive Reachability*
- GUI input spaces are no longer easy to determine
The Problem: Incomplete Input Spaces

- Contemporary GUIs are Context-sensitive Reachability
- GUI input spaces are no longer easy to determine
The Problem: Incomplete Input Spaces

- Contemporary GUIs are **Context-sensitive Reachability**
- GUI input spaces are no longer easy to determine
The Problem: Incomplete Input Spaces

• Contemporary GUIs are *Context-sensitive Reachability*

• GUI input spaces are no longer easy to determine
The Problem: Incomplete Input Spaces

- Contemporary GUIs are **Context-sensitive Reachability**
- GUI input spaces are no longer easy to determine
Thesis Statement

“GUI-based applications can be effectively and efficiently tested by systematically and dynamically leveraging application execution observations”
Approach:

Observe-Model-Exercise* (OME*)
GUI Testing Paradigm

[TSE 2012, ASEJ 2012]
What’s Next

• OME* overview
• Intellectual challenges and solutions
• Empirical studies
• Discussion and Future work
OME* Overview

Application Under Test → Observe

Observe → Runtime States, New GUI Elements

Runtime States, New GUI Elements → GUI Model & Test Cases

GUI Model & Test Cases → Model

Exercise → Model
Five Intellectual Challenges

**Ch1**: How to replicate event context-sensitive behaviors?

**Ch2**: How to extract new GUI elements?

**Ch3**: How to identify new GUI elements?

**Ch4**: How to add new elements to the model?

**Ch5**: How to generate additional test cases?
Goal: Automatically exercise all possible event pairs on the GUI (i.e., the pairwise test adequacy criteria)
Context-Aware GUI Model

Ch1: How to replicate context-sensitive behaviors?

• Context-Aware Mapping (CAM):
  \{model element, context event sequence\}

• Dynamically built at run-time by observing the GUI

• EFG$^+$ = EFG + CAM

Available events observed during execution
New Model Element Extraction

Ch2: How to extract new GUI elements?

- Dynamically capture and analyze GUI states after each GUI events

- Examples of “observing” technology:
  - Accessibility APIs
  - Object reflection
  - Dynamic HTML source analysis
Unique Widget Signature

Ch3: How to identify new widgets?

- **Widget signature**

\[
\begin{align*}
    w_{state} & \leftarrow \langle (p_1, v_1), \ldots, (p_n, v_n) \rangle \\
    \langle v_i, \ldots, v_k \rangle & \leftarrow \text{select}(\text{filter}_p, w_{state}) \\
    w_{sig} & \leftarrow \Gamma(C_{state}; \gamma_i(v_i), \ldots, \gamma_k(v_k))
\end{align*}
\]

- **Container signature**

\[
\begin{align*}
    C_{state} & \leftarrow \langle (p_1, v_1), \ldots, (p_n, v_n) \rangle \\
    \langle v_i, \ldots, v_k \rangle & \leftarrow \text{select}(\text{filter}_p, C_{state}) \\
    C_{sig} & \leftarrow \Phi(\phi_i(v_i), \ldots, \phi_k(v_k))
\end{align*}
\]
Incremental Model Enhancement

*Ch4: How to add new elements to the model?*

- Add new EFG nodes and edges
- Add new CAM entries
- Update existing CAM entries

*Execute test case* \(<e_1, e_4>\)

<table>
<thead>
<tr>
<th>Edge</th>
<th>Path to Edge</th>
</tr>
</thead>
<tbody>
<tr>
<td>(e4, e5)</td>
<td>&lt;e2, e3&gt;</td>
</tr>
<tr>
<td>(e2, e1)</td>
<td>NONE</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

CAM

EFG
Incremental Test Case Generation

Ch5: How to generate additional test cases?

- Identify uncovered model elements
- Search a path to model elements
  - First, look up in CAM
  - If failed, rely on EFG topology

<table>
<thead>
<tr>
<th>Edge</th>
<th>Path to Edge</th>
</tr>
</thead>
<tbody>
<tr>
<td>(e4, e5)</td>
<td>&lt;e2, e3&gt;</td>
</tr>
<tr>
<td>(e2, e1)</td>
<td>NONE</td>
</tr>
<tr>
<td>(e4, e8)</td>
<td>&lt;e1&gt;</td>
</tr>
<tr>
<td>(e4, e5)</td>
<td>&lt;e1&gt;</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

CAM

EFG
Summary: Five Intellectual Contributions

**C1:** A new context-ware GUI model

**C2:** A dynamic GUI element extraction technique

**C3:** A unique widget identification schema

**C4:** An incremental model enhancement algorithm

**C5:** An incremental test case generation algorithm
OME* Illustration
Empirical Study

Key research questions:
1. How does the input space expand with OME*?
2. What are the impacts of input space enhancement?
   - Test adequacy
   - Fault detection effectiveness

Independent variable: Testing techniques
1. **BL**: Baseline case for GUI Ripping-based technique
2. **NoMap**: OME* without Context-Aware Mapping
3. **RND**: Random-walk on the GUI (perform 5 times)
4. **OME***: Full OME* with Context-Aware Mapping
## Subject Applications

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Abbv.</th>
<th>Version</th>
<th>Age (years)</th>
<th>Size (KLOC)</th>
<th>Download (x1000)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ArgoUML</td>
<td>AU</td>
<td>0.33.1</td>
<td>13</td>
<td>70.0</td>
<td>N/A</td>
</tr>
<tr>
<td>2</td>
<td>Buddi</td>
<td>BD</td>
<td>3.4.0.8</td>
<td>7</td>
<td>9.6</td>
<td>897.5</td>
</tr>
<tr>
<td>3</td>
<td>CrosswordSage</td>
<td>CS</td>
<td>0.3.5</td>
<td>8</td>
<td>1.8</td>
<td>4.6</td>
</tr>
<tr>
<td>4</td>
<td>DrJava</td>
<td>DJ</td>
<td>r5004</td>
<td>10</td>
<td>65.0</td>
<td>1,227.4</td>
</tr>
<tr>
<td>5</td>
<td>JabRef</td>
<td>JR</td>
<td>2.7b</td>
<td>9</td>
<td>44.5</td>
<td>1,173.3</td>
</tr>
<tr>
<td>6</td>
<td>OmegaT</td>
<td>OT</td>
<td>2.1.3</td>
<td>10</td>
<td>19.8</td>
<td>254.6</td>
</tr>
<tr>
<td>7</td>
<td>PdfSam</td>
<td>PS</td>
<td>2.2.1</td>
<td>7</td>
<td>6.1</td>
<td>2,548.4</td>
</tr>
<tr>
<td>8</td>
<td>Rachota</td>
<td>RC</td>
<td>2.3</td>
<td>9</td>
<td>11.2</td>
<td>74.4</td>
</tr>
</tbody>
</table>
Running Experiments

- Test case execution
  - Total test runs: \(~400,000\) (1-5 mins/run)
  - Execution time: \(~1000\) machine-days
  - Data storage: 3TB

- Tool: GUITAR
  - 50 KLOCs, 6 GUI platforms, 8K downloads
  - [http://guitar.sourceforge.net](http://guitar.sourceforge.net)

- Computing resource: Skoll cluster
  - 120 nodes
  - [http://skoll.umiacs.umd.edu](http://skoll.umiacs.umd.edu)
Result: Input Space Expansion

Node expansion

Edge expansion
Result: Event Coverage

% Node Covered

% Edge Covered
Result: Fault Detection Effectiveness (Crash)

- Total faults: 56
- New faults: 34
- Missing faults: 1

ArgoUML – Fault Detection

BL

i1

i2

i3

i4

feasible
infeasible
error
Fault Example: Revert to Saved A Document

- **e1:** Insert a class
- **e2:** File
- **e3:** Revert to Saved
- **e4:** Yes
Result: Code Coverage

Statement coverage

Branch coverage

Does not improve much!
Thesis Statement

“GUI-based applications can be effectively and efficiently tested by systematically and dynamically leveraging application execution observations”
• Point out the input space determination problem for GUI testing
  – And propose a solution
• Propose a new direction for software testing research
• Expand the “horizon” of other EFG-based approaches
• Provide a tool for project management
• Open new opportunities for other research areas
  – Reverse engineering through testing
  – Specification mining
Direct Extensions

• Expand the subject application pool
• Explore different test adequacy criterion
• Study different input space exploration strategies
• Study the correlation between model creation and test case generation
• Expand OME* to non-GUI testing
  – Object-orient programs, Distributed systems, Component-based systems
Broader Impacts: In Research

• New research topics:
  – Predicting human performance in HCI [CHI’12 & ICSE’13]
  – Static analysis for GUI testing [ISSRE’12]
  – Test repairing [ISSRE'12]
  – Android & Mobile Testing [ASE'12]
  – Usage-based GUI test case generation [under submission]
  – Code coverage interaction in GUI testing [under submission]
  – Automated GUI debugging [under submission]

• Evaluation benchmarks:
  – COMET: http://comet.unl.edu/
  – Maryland Dozen: http://samwise.cs.umd.edu:8080
Broader Impacts: Outside Research

• Education
  - UMD:
    - CMSC 435: Software Engineering
    - CMSC 426: Programming Handheld Systems
    - CMSC 737: Fundamentals of Software Testing
  - UNL:
    - CSE 496/896: AI and Heuristic Techniques in SE
    - CSE 990: Software Quality Methods
  - UIUC:
    - CS 527: Topics in Software Engineering
  - UVa:
    - CS 4501: Special Topics in Computer Science
  - ICSE 2013 tutorials

• Industry
  - Open source communities
  - Google Analytics Team
  - Cisco, VMware
Question & Answer
Backup
Why wasn’t code coverage improved?

• In GUI, code is not created equal
  – Initialization code is always covered
  – Some code is hard to cover
    • Environment specific code
    • Error handling code
  – Some code is unable to cover (i.e. dead-code)
Event Interaction Matters: BD3

Buddi.java

```java
InetAddress proxyAddress = InetAddress
    .getByName(PrefsModel.getInstance().getProxyServer());
InetSocketAddress socketAddress =
    new InetSocketAddress(proxyAddress,
        PrefsModel.getInstance().getPort());
```

NetworkPreferences.java

```java
try {
    PrefsModel.getInstance().
        setPort(Integer.parseInt(port.getText().
            .replaceAll(\"\D\", \"\")));
} catch (NumberFormatException nfe){
    Logger.getLogger()
        .warning("Incorrect port number; setting to 80");
} `
State Matters: Fault AU3

Code to setup the faulty value

```java
if (retval == JFileChooser.APPROVE_OPTION) {
    File thefile = chooser.getSelectedFile();
    if (thefile != null) {
        String path = thefile.getPath();
        Configuration.setString(,
            SaveGraphicsManager.KEY_SAVE_GRAPHICS_PATH,
            path);

        thefile = new File(thefile.getParentFile(),
            sgm.fixExtension(thefile.getName()));
        String suffix = sgm.getFilterFromFileName(thefile.getName());
        getSuffix();
        return doSave(thefile, suffix, true);
    }
}
```

Code to Consume the faulty value

```java
FileOutputStream fo = new FileOutputStream(thefile);
```
```java
cmd.setStream(fo);
```
AU9: 3-way bug
Fault AU9: Event Interaction Matters (2)
Buddi Input Space Expansion
GUITAR in Different Platforms
Experiment Variables

• Dependent variables
  – Input space expansion
  – Event coverage
  – Code coverage
  – Faults detected