An Initial Study on Ideal GUI Test Case Replayability

Arthur-Jozsef Molnar
Babes-Bolyai University
What’s the problem?

• GUI applications are ubiquitous
  – Lots of tools for design
  – Lots of platforms to choose from
  – Lots of widget toolkits
  – Great testing processes
What’s the problem?

• GUI applications are ubiquitous
  – Lots of tools for design
  – Lots of platforms to choose from
  – Lots of widget toolkits
  – Great testing processes
Existing solutions...

• Capture/Replay tools
  – Marathon
    • http://www.marathontesting.com/
  – Abbot
    • http://abbot.sourceforge.net/doc/overview.shtml
  – Pounder
    • http://pounder.sourceforge.net/

• Model based approaches
  – Automated testing using GUITAR[1]
    • Reverse engineering the target GUI
    • Creating Test Cases
    • Running Test Cases
    • Checking the outcome

GUIs change...
Reusing GUI test cases

- **Using genetic algorithms to repair non-executable tests**

- **Inserting and removing events**

- **Identifying functionally equivalent GUI elements and updating tests**
Reusing GUI test cases

• Proposed solutions ...
  – Using genetic algorithms to repair non-executable tests
  – Inserting and removing events
  – Identifying functionally equivalent GUI elements and updating tests

⇒ No turnkey solution ... yet!
How many test case can be repaired and ... how well?
Our study

1. **We use 28 versions of two well established open source applications**
   - FreeMind.
   - jEdit.

2. **We build information on functionally equivalent GUI widgets between consecutive versions.**
   - Our oracle information.

3. **We generate test cases of length 2, 3, 4 and 5.**
   - A test case is a sequence of GUI interactions

4. **We partition test cases into one of four categories.**
   - We simulate test case execution due to computation constraints

5. **We assess the results**
   - Cross-sectional study
   - Longitudinal study
1. The applications - FreeMind

- 13 versions studied
- November 2000 – September 2007

- Taken from project CVS
- 3500 – 65K lines of code
- 13.5 million downloads
1. The applications - jEdit

- 17 versions studied
- January 2000 – May 2010

- Versions 2.3pre2 – 4.3.2
- 23K – 106K lines of code
- 6.5 million downloads
2. Functionally equivalent widgets

Examples of functionally equivalent widgets

2. Functionally equivalent widgets

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accelerator</td>
<td>The shortcut key combination used to trigger the binded action</td>
</tr>
<tr>
<td>Class</td>
<td>The class implementing the widget</td>
</tr>
<tr>
<td>Icon</td>
<td>The path to the widget’s icon</td>
</tr>
<tr>
<td>Text</td>
<td>The text associated with the widget</td>
</tr>
<tr>
<td>Title</td>
<td>The widget’s title</td>
</tr>
<tr>
<td>X</td>
<td>The X coordinate of the widget’s upper corner in its containing window</td>
</tr>
<tr>
<td>Y</td>
<td>The Y coordinate of the widget’s upper corner in its containing window</td>
</tr>
<tr>
<td>Width</td>
<td>The widget’s width</td>
</tr>
<tr>
<td>Height</td>
<td>The widget’s height</td>
</tr>
<tr>
<td>Index</td>
<td>The widget’s index in the parent container</td>
</tr>
</tbody>
</table>
2. Functionally equivalent widgets

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent</td>
<td>The <em>Modes</em> menu</td>
</tr>
<tr>
<td>Children</td>
<td>-</td>
</tr>
<tr>
<td>Class</td>
<td><code>javax.swing.JMenuItem</code></td>
</tr>
<tr>
<td>Accelerator</td>
<td>Alt+3</td>
</tr>
<tr>
<td>Title</td>
<td>-</td>
</tr>
<tr>
<td>Icon</td>
<td>-</td>
</tr>
<tr>
<td>X</td>
<td>319</td>
</tr>
<tr>
<td>Y</td>
<td>45</td>
</tr>
<tr>
<td>Width</td>
<td>129</td>
</tr>
<tr>
<td>Height</td>
<td>21</td>
</tr>
<tr>
<td>Index</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent</td>
<td>The <code>javax.swing.JToolBar</code> element</td>
</tr>
<tr>
<td>Children</td>
<td>-</td>
</tr>
<tr>
<td>Class</td>
<td><code>org.gjt.sp.jedit.gui.EnhancedButton</code></td>
</tr>
<tr>
<td>Accelerator</td>
<td>-</td>
</tr>
<tr>
<td>Title</td>
<td>-</td>
</tr>
<tr>
<td>Icon</td>
<td><code>org/gjt/sp/jedit/icons/Cut24.gif</code></td>
</tr>
<tr>
<td>X</td>
<td>196</td>
</tr>
<tr>
<td>Y</td>
<td>47</td>
</tr>
<tr>
<td>Width</td>
<td>30</td>
</tr>
<tr>
<td>Height</td>
<td>30</td>
</tr>
<tr>
<td>Index</td>
<td>7</td>
</tr>
</tbody>
</table>
3. Generated test cases

- All length-2 test cases (event interaction)
- 10,000 length-3, length-4 and length-5 test cases (each)
- Grand total:
  - 404,826 FreeMind test cases
  - 564,869 jEdit test cases

- Test cases generated and replay simulated for all subsequent versions 3 times => ~ 24 million test case execution simulations
4. Test case replayability

- **Replayable using widget Id**
  - Widget Id computed by some of its properties
  - Used by GUITAR
  - Easy to implement but error prone
  - *Sadly, the state of the art 😞*

- **Replayable after repair**
  - Perfectly replayable

- **Repairable**
  - Not the same intermediary steps

- **Unrepairable**
  - Some test steps cannot be replicated
  - Might still be usable
  - No correspondence with initial test case
4. Study approaches

• **Cross-sectional**
  – We use consecutive application versions
  – Test replay is simulated and test cases classified
  – Can reveal version pairs that have been heavily changed

• **Longitudinal**
  – We take test cases built for the first studied version
  – Simulate their execution on all subsequent versions
  – Classify them
  – *A true measure of test case replayability 😊*
4. Cross-section results - FreeMind

Replayable using widget Id
Replayable after repair
Repairable
Unrepairable
4. Cross-section results - jEdit

![Cross-section results chart for jEdit]

*An Initial Study on Ideal GUI Test Case Replayability*
4. Longitudinal results - FreeMind
4. Longitudinal results - jEdit

An Initial Study on Ideal GUI Test Case Replayability
4. Conclusions

• What affects test replayability?
  – Major GUI changes
  – Test case age
  – Test case length

• Most test cases can be at least repaired
4. Future steps

• A more comprehensive study
  – More versions
  – More applications
    • .NET, Qt, SWT, mobile ...

• Run test cases, not just simulate them
Thank you!