Complementing GUI Testing Scripts with Smart Monkey Testing

Dr. Pekka Aho

15 September 2022
Automated GUI testing scripts

Traditionally GUI testing scripts are...
- designed manually
  - captured with C&R tools, or
  - manually written test scripts
- static until maintenance required
  - maintained manually
- for regression testing
  - checking the correctness of a selected set of functionality and properties

Test step 1
  \[\text{Check detail A}\]
  \[\text{Check detail B}\]
Test step 2
  \[\text{Check detail C}\]
Monkey testing

In monkey testing, the tests are...

- generated during run-time
  - no test scripts to maintain
- dynamic, randomness often included
  - multiple executions could yield different results
- closer to robustness testing than functional testing
  - trying to find incorrectness anywhere in the system under testing
Monkey testing complements scripts

GUI testing scripts and monkey testing...
- find different kind of failures
- cover different parts of code

<table>
<thead>
<tr>
<th>Tool</th>
<th>New bugs</th>
<th>Known bugs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selenium</td>
<td>2 high</td>
<td>1 high</td>
</tr>
<tr>
<td>TESTAR</td>
<td>2 high 4 low</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tool</th>
<th>% IC</th>
<th>% LC</th>
<th>% MC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Espresso</td>
<td>43,9</td>
<td>43,4</td>
<td>45,9</td>
</tr>
<tr>
<td>TESTAR</td>
<td>41,0</td>
<td>40,7</td>
<td>40,8</td>
</tr>
<tr>
<td>combined</td>
<td>52,3</td>
<td>52,1</td>
<td>52,3</td>
</tr>
</tbody>
</table>
Monkey testing complements scripts
Random monkey testing is effective

Case study: Cap Gemini/ProRail

- Academia/industry cooperation
  - Open source TESTAR tool with random action selection and default test oracles
- Web application: Java based (JEE6) with 12,263 LOC
- System for managing the assignment of train platforms
- Existing test approach: manual with 100 test cases
- After a change in the application: all 100 test cases were executed as regression tests

<table>
<thead>
<tr>
<th></th>
<th>TESTAR</th>
<th>Manual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preparation</td>
<td>44 hour</td>
<td>43 hour</td>
</tr>
<tr>
<td>Testing</td>
<td>(51 hour)</td>
<td>6 hour</td>
</tr>
<tr>
<td>Post testing</td>
<td>5 hour</td>
<td>2 hour</td>
</tr>
<tr>
<td>Critical faults</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Functional coverage</td>
<td>80%</td>
<td>73%</td>
</tr>
</tbody>
</table>
How to make the monkey smarter

The monkey testing tool can be enhanced with...

- a memory (model) to remember where it has been and what it has done
- pre-defined information about the SUT
  - how to reach in every part of the SUT (e.g., login)
  - SUT-specific test inputs or test oracles
- improved action selection algorithms
  - AI or ML to learn how to reach a certain goal
  - using the memory (model) to systematically explore
- self-coordinating parallel GUI exploration
Open source TESTAR tool

Smart monkey testing tool
- state model inference based on observed GUI behaviour
- various action selection algorithms (random, systematic, ML)
- supports SUT-specific information and configuration
- distributed GUI exploration with independent TESTAR agents and shared state model database

Open source
- https://github.com/TESTARtool/TESTAR_dev
- https://testar.org/

Many ongoing research projects with industrial collaboration
Any further questions?

More details from:

pekka.aho@ou.nl or https://testar.org/