A Planning-based Approach to GUI Testing

Atif M. Memon
Martha E. Pollack
Mary Lou Soffa

Dept. of Computer Science
University of Pittsburgh
High-level Outline

- Design of PATHS (Planning Assisted Tester for graphical user interface Systems)
- Implementation Status
- Overview of Planning & Test Case Generation

The GUI Testing Framework

- GUI Specifications
- Planner based Test Case Generator
- Test Cases
Oracle Information Augmentation Tool

Test Suite (Test Cases + Oracle Information)

Test cases and Oracle Information from Previous Version

Regression Testing Algorithms

Update for Regression Testing

Coverage Evaluation Algorithms

Coverage Criteria

Coverage Report

Output

Output
Entire

Implementation Status

GUI Specifications

Automated Planning-based Test Case Generator

Test Cases

Automated Generation
Automated Oracle Creator → Test Oracles

Test Executor → GUI Under Test

Automated Execution
What is a GUI Test Case?

- Individual User Events
  - NOT ENOUGH!
  - Sequences of User Events lead to Different States
- Test Case: Sequence of User Events
Generating Test Case Sequences

- Randomly Choose Sequences
- Expert Chooses Sequences
- Generate Sequences from a Description of the GUI

Novel Idea

- Automatically Generate Test Cases for **EXPECTED TASKS**
AI Planning and GUI Testing

- Recent Advances in AI Planning
  - Propositional Planners
    - Very Fast
    - Based on
      - Flow-Graphs [Koehler et al. '97]
      - SAT Solving [Kautz & Selman '96]

- AI Planning, Especially Suitable for
  - Generating Test Cases
  - Automatically Generating Verification Information
  - Regression Testing

A Plan

Initial State

Action A

Action B

Action C

Goal State
A Plan for a GUI Task
Overview of Test Generation

<table>
<thead>
<tr>
<th>Phase</th>
<th>Step</th>
<th>Test Designer</th>
<th>Automatic Planning-based System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setup</td>
<td>1</td>
<td>Derive Planning Operators from GUI</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Code Preconditions and Effects of Operators</td>
<td></td>
</tr>
<tr>
<td>Test Case Generation</td>
<td>3</td>
<td>Specify a Task (Initial and Goal States)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Generate Test Cases</td>
<td></td>
</tr>
</tbody>
</table>

Straightforward Approach

- Define One Operator for each User Action

Operator :: \textsc{CUT}
Preconditions:
\textsc{isCurrent} (Menu2).

Effects:
\textsc{FORALL} Obj in Objects
\quad \textsc{Selected}(Obj)
\quad \textsc{ADD} \textsc{inClipboard}(Obj)
\quad \textsc{DEL} \textsc{onScreen}(Obj)
\quad \textsc{DEL} \textsc{Selected}(Obj)
\quad \textsc{ADD} \textsc{isCurrent} (Menu1)
\quad \textsc{DEL} \textsc{isCurrent} (Menu2).

\textit{First Order Predicate Logic}
Exploit the GUI's Structure

- Reduce the Number of Operators
  - System more Efficient
  - Easier for the Test Designer

Opening Modal Windows
Opening Menus

File Edit View Insert Format Tools Table Window Help

- New...
- Open...
- Close
- Save
- Save As...
- Save as HTML...
- Versions...
- Page Setup...
- Print Preview
- Print...

Send To

Interacting with the Underlying Software

Underlying Software
Create Hierarchical Operators

Two Types of Abstractions
- Combine Buttons Create System-Interaction Operators
- Decompose GUI Hierarchically Create Abstract Operators

Create System-Interaction Operators

Sys-Interaction Operator:
File_SendTo_MailRecipient
= <File + SendTo + MailRecipient >
Create Abstract Operators

Set Language
Thesaurus... Shift+F7
Hyphenation...

Straightforward Approach
Main GUI's Operator Set

Set Language
SelectFromList()
Default
OK
Cancel

Using Abstraction
Language Window's Operator Set

SelectFromList() Default
OK
Cancel

High Level Plan
SetLanguage() -> ...

Define Abstraction
Abstract Operator

Sub Plan
SelectFromList("English(US)") -> OK
Effects of Exploiting the GUI's Structure

- Reduction in Planning Operators
  - 325 operators  32 operators
  - Ratio 10:1 for MS WordPad
  - 20:1 for MS Word

- System Automatically Determines the System-interaction and Abstract Operators
Test Case

 INITIAL Primitive Operator SelectText ("This") Abstract Operator FormatFont ("This", 18pt) Primitive Operator SelectText ("text") Abstract Operator FormatFont ("text", Underline) GOAL

 Planner
 FormatFont ▶ 18 ▶ OK
 Planner
 FormatFont ▶ Underline ▶ OK
**Alternative Test Case**

```
Primitive Operator
SelectText ("This")
```

```
Abstract Operator
FormatFont ("This", 18pt)
```

```
Primitive Operator
SelectText ("text")
```

```
Abstract Operator
FormatFont ("text", Underline)
```

```
Primitive Operator
SelectFromList (18)
```

```
Abstract Operator
SelectText ("text")
```

```
Primitive Operator
SelectText ("This")
```

```
Format
Font
Underline
OK
```
Methods to Generate Alternative Test Cases

- Different Results from Planner
- Abstract Operator Decompositions
- Linearizations of the Partial-order Plan

Experimental Results

![Graph showing time (sec.) vs. test-case length](image)
<table>
<thead>
<tr>
<th>(Task) Plan No.</th>
<th>Plan Time (sec.)</th>
<th>Sub Plan Time (sec.)</th>
<th>Total Time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.16</td>
<td>0</td>
<td>3.16</td>
</tr>
<tr>
<td>2</td>
<td>3.17</td>
<td>0</td>
<td>3.17</td>
</tr>
<tr>
<td>3</td>
<td>3.2</td>
<td>0.01</td>
<td>3.21</td>
</tr>
<tr>
<td>4</td>
<td>3.38</td>
<td>0.01</td>
<td>3.39</td>
</tr>
<tr>
<td>5</td>
<td>3.44</td>
<td>0.02</td>
<td>3.46</td>
</tr>
<tr>
<td>6</td>
<td>4.09</td>
<td>0.04</td>
<td>4.13</td>
</tr>
<tr>
<td>7</td>
<td>8.88</td>
<td>0.02</td>
<td>8.9</td>
</tr>
<tr>
<td>8</td>
<td>40.47</td>
<td>0.04</td>
<td>40.51</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task No.</th>
<th>Single Level</th>
<th>Hierarchical</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Plan Length</td>
<td>Time (sec.)</td>
</tr>
<tr>
<td>1</td>
<td>18</td>
<td>8.93</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>47.62</td>
</tr>
<tr>
<td>3</td>
<td>24</td>
<td>189.87</td>
</tr>
<tr>
<td>4</td>
<td>26</td>
<td>3312.72</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Experimental Results (contd…)

Executing Test Cases and Verifying Results

Test Oracles & Regression Testing

- Exploiting Planning Algorithms to get Expected Behavior of GUI
- Using Hierarchical GUI Model for Efficient Regression Testing
Concluding Remarks

- Overview of PATHS & Current Status
- Planning & Test Case Generation
- Automatic Generation of Preconditions and Effects from GUI Specifications