Using a Goal-driven Approach to Generate Test Cases for GUIs

Atif M. Memon
Martha E. Pollack
Mary Lou Soffa

Dept. of Computer Science
University of Pittsburgh
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
- How to Generate Test Cases? Infinitely many Sequences

Generating Test Sequences

- Randomly Choose Sequences
- Expert Chooses Sequences
- Automatically Generate Events for COMMONLY USED TASKS
Recent Advances in AI Planning

- Propositional Planners
  - Very Fast
  - Based on
    - Flow-Graphs [Koehler et al. ‘97]
    - SAT Solving [Kautz & Selman ‘96]
Outline

- Using Planning for Test Case Generation
  - Overall Approach
  - Exploiting GUI Structure
  - Generating Alternative Test Cases
- Experimental Results
- Related Research
- Concluding Remarks
## 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></td>
<td>Derive Planning Operators from GUI</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></td>
<td>Generate Test Cases</td>
</tr>
</tbody>
</table>

### A Plan for a GUI Task

**Initial State**

![Initial State](image1)

**Goal State**

![Goal State](image2)

- **SelectText(”This”)**
  - This is the text.

- **SetFontSize(18)**
  - Font size is 18.

- **SelectText(”text”)**
  - This is the text.

- **MouseClicked(U)**
  - Mouse click on the text.
Straightforward Approach

- Define **One Operator for each User Action**

**Operator :: CUT**

**Preconditions:**
- `isCurrent(Menu2)`

**Effects:**
- `FORALL Obj in Objects`
  - `Selected(Obj)`
  - `ADD inClipboard(Obj)`
  - `DEL onScreen(Obj)`
  - `DEL Selected(Obj)`
  - `ADD isCurrent(Menu1)`
  - `DEL isCurrent(Menu2)`.

First Order Predicate Logic

Exploit the GUI’s Structure

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

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

Expansion Operator:

File_SendTo_MailRecipient

= <File + SendTo + MailRecipient>

Create Abstract Operators

Using Abstraction

Window's Language

Main GUI's Operator Set

SelectFromList()

Set Language

Default

OK	Cancel

Straightforward Approach

Main GUI's Operator Set

SelectFromList()

Set Language

Default

OK	Cancel
Create Abstract Operators

Language Window's Operator Set

- SelectFromList()
- Default
- OK
- Cancel

High Level Plan

- ... → SetLanguage() → ... → Planner

Sub Plan

- SelectFromList ("English(US)") → OK

Define Abstraction

SetLanguage()

Abstract Operator

Effects of Exploiting the GUI's Structure

- Reduction in Planning Operators
  - 325 operators → 32 operators
  - Ratio 10:1 for MS WordPad
- System Automatically Determines the Expansion and Abstract Operators
Test Case

Initial State

Goal State

This is the text.
Planner
FormatFont → 18 → OK

Planner
FormatFont → Underline → OK

Expand Macro
Format → Font

Expand Macro
Format → Font

SelectText ("This")
Format → Font → 18 → OK → SelectText ("text")

Format → Font → Underline → OK
Alternative

Methods to Generate Alternative Test Cases

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

• Purpose
  - To Determine whether Planning is a Feasible Approach for GUI Test Case Generation
    • Execution Time
    • Human Effort

• Experimental Design
  - GUI: MS WordPad
  - Planner: IPP [Koehler et al. ’97]
  - Hardware Platform: 300 MHz Pentium based Machine, 200 MB RAM, Linux OS
  - 8 Tasks, Multiple Test Cases for each Task

Experimental Results

<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>
Related Work

• GUI Testing
  - Genetic Algorithm Technique [Kasik and George]
  - Visual TDE for GUIs [Foster, Goradia, Ostrand, and Szermer]

• Planning for Testing
  - [Adele Howe, Anneliese Von Mayrhauser, Richard Mraz in ASE ’97]

Concluding Remarks

• Automatic Planning is a Feasible Approach for GUI Test Case Generation

• Test Generation for a Larger GUI
• Automatic Generation of Preconditions and Effects from GUI Specifications
• Generate Expected Output