Plan Generation for GUI Testing

Atif M. Memon
Martha E. Pollack
Mary Lou Soffa

Dept. of Computer Science
University of Pittsburgh

Research Focus

Interactions between the GUI and the Underlying Code
Why Planning for GUI Testing

- GUIs are Event Driven
- Individual User Events
  - NOT ENOUGH!
  - Sequences of User Events lead to Different States
- Test Case: Sequence of User Events
- How to Generate Test Cases?
- Use Planning to Select Likely Test Cases

Selecting Test Sequences

- Infinitely Many
- Randomly Choose Sequences
- Expert Chooses Sequences
- Automatically Generate Events for COMMONLY USED TASKS
A Plan for a GUI Task

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>

### Straightforward Approach

- Define One Operator for each User Action

<table>
<thead>
<tr>
<th>Menu2</th>
<th>Menu1</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>Edit</td>
</tr>
<tr>
<td>Copy</td>
<td>View</td>
</tr>
<tr>
<td>Cut</td>
<td>Ins</td>
</tr>
<tr>
<td>Paste</td>
<td>Del</td>
</tr>
</tbody>
</table>

**Operator :: CUT**

**Preconditions:**

\[ \text{isCurrent}(\text{Menu2}). \]

**Effects:**

\[ \text{FORALL Obj in Objects} \]
\[ \text{Selected}(\text{Obj}) \]
\[ \text{ADD inClipboard}(\text{Obj}) \]
\[ \text{DEL onScreen}(\text{Obj}) \]
\[ \text{DEL Selected}(\text{Obj}) \]
\[ \text{ADD isCurrent}(\text{Menu1}) \]
\[ \text{DEL isCurrent}(\text{Menu2}). \]
Exploit the GUI's Structure

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

Opening Modal Windows
Opening Menus

Interacting with the 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>

[Diagram of GUI with Send To menu item and options]
Create Abstract Operators

Straightforward Approach
Main GUI's Operator Set

- Set Language
- SelectFromList()
- Default
- OK
- Cancel

Using Abstraction
Language Window's Operator Set

- Set Language
- SelectFromList()
**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**

---

**Initial State**

This is the text.

**Goal State**

This is the text.
Test Case

INITIAL

Primitive Operator
SelectText ("This")

Abstract Operator
FormatFont ("This", 18pt)

Primitive Operator
SelectText ("text")

Abstract Operator
FormatFont ("text", Underline)

GOAL

Primitive Operator
FormatFont 18 OK

Abstract Operator
Planner

Primitive Operator
FormatFont Underline OK

Abstract Operator
Planner
Different from HTN Planning

No Interactions
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
- Automatic Generation of Preconditions and Effects from GUI Specifications
- Generate Expected Output (Automated Verification)