Automated Test Oracles for GUIs

Atif M. Memon
Martha E. Pollack
Mary Lou Soffa

Dept. of Computer Science
University of Pittsburgh

GUI Test Cases

• GUI Test Case
  - Sequence of events
A Test Case for WordPad

SelectText ("This") -> Format -> Font -> 18 -> OK -> SelectText ("text")

Format -> Font -> Underline -> OK

What Is Correct Behavior

SelectText ("This") -> Format -> Font -> 18 -> OK

SelectText ("text") -> Format -> Font -> Underline -> OK

This is the text.
Check State, not only Output!!

**Research Focus**

- **Goal**
  - To check the GUI's state after each event

- **Approaches**
  - Manual
  - Automated

- **Challenges**
  - Generating expected state
  - Extracting actual state
  - Comparing expected & actual states
Outline

- Overview of GUI Oracle
- Generating Expected State
  - Modeling the GUI's State
    - Objects
    - Properties
  - Modeling the Events
- Obtaining Actual GUI's State
- Comparing Actual & Expected States
- Case Study: MS WordPad
- Concluding Remarks

Overview of GUI Oracle

![Diagram showing the overview of GUI Oracle]

- Test Case
- Formal GUI Model
- Expected-state Generator
- Expected State
- Oracle
- Verdict
- Actual State
- Execution Monitor
- Run-time information from executing GUI
Modeling the GUI

A GUI consists of Objects

- Button
- Form
- Label
- Window State
- Width
- AutoScroll
- Align
- Caption
- Color
- Font
- Caption
- Enabled
- Visible
- Height
All Properties of Cancel

- Cancel: TRUE
- Caption: Cancel
- Font: TFont
- Height: 65
- Hint: None
- Left: 0
- ModalResult: mlNone
- Name: Button1
- ParentFont: False
- PopupMenu: True
- ShowHint: False
- TabOrder: 0
- TabStop: True
- Tag: 0
- Top: 0
- Visible: True
- Width: 150
Determining Properties

- Manual Examination of GUI
- Specifications (Reduced Set)
  - GUI being tested
- Toolkit/Language (Complete Set)
  - All available properties

Now we know how to represent the GUI’s state

Modeling Events

- Events are State Transducers

State: $S_i$

Event: $e$

Notation: $S_j = [S_i, e]$

State: $S_j$
Representing Events

- We define an event as:
  \[ \text{State}_j = [\text{State}_i, \text{event}] \]
- For example:
  \[ \text{State}_j = [\text{State}_i, \text{cut}] \]
- Need a compact representation

Operators

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)`

**Obtaining next state**
**Deriving Expected State**

- Given $S_0$, the initial state,
- A sequence of events $e_1 \rightarrow e_2 \rightarrow e_3 \rightarrow \ldots \rightarrow e_n$
- Obtain $S_1 = [S_0, e_1]$
- And $S_i = [S_{i-1}, e_i]$

![Diagram showing sequence of states $S_0, S_1, S_2, S_3, \ldots, S_n$ with transitions $e_1, e_2, e_3, \ldots, e_n$.]

**Obtaining Actual GUI’s State**

- **Execution Monitor**
  - Screen Scraping
  - Queries
  - Compatible with Expected State
  - Returns <Object, Property, Value>
    - <Button1, "Caption", "Cancel">
Automated Execution

Test Executor

GUI Under Test

Test Cases

Expected State

ACTUAL STATE:
(isCurrent ROOT)
(Contains ROOT D)
(Contains ROOT D)

Execution Monitor

Verifier

ACTUAL STATE:
(isCurrent ROOT)
(Contains ROOT D)
(Contains ROOT D)
Comparing Actual and Expected States

- Verifier
- Three Levels of Testing
  - Changed Property Set *(Operators)*
  - GUI Relevant Property Set *(Specifications)*
  - Complete Property Set *(Toolkit/Language)*
- Hybrid Approach
  - Use all 3

Case Study

- **Purpose:** Determine
  - Time to Derive Expected State
  - Time to Execute Monitor and Verifier
- **Experimental Design**
  - **GUI:** Our Version of MS WordPad (36 Modal Windows, 362 events)
  - **Test Cases:** Generated 290 Test Cases (6-56 events) using an AI Planner
  - **Hardware Platform:** 350 MHz Pentium based Machine, 256 MB RAM
  - **Properties:** Reduced Set
  - **Level of Testing:** GUI Relevant Property Set
Deriving Expected State

Generating Test Cases and Deriving Expected State

Total CPU time (test case and expected state) 75.84 sec.
### Execution

Executing Test Cases, Verifier and Execution Monitor

![Graph showing time vs. test-case length]

- **Test Case + Verifier + Execution Monitor**
- **Verifier + Execution Monitor**
- **Test Case**

- **Relevant-properties verification**
- **Total running time < 10 minutes**

### Related Work

- **GUIs**
  - PATHS [Memon et al., ICSE'99]
  - TDE [Ostrand et al., ISSTA'98]
  - VFSM/FSM [Shehady et al., FTCS'97]

- **Conventional Software - Oracles**
  - TAOS [Richardson et al., ISSTA'94]
  - TOBAC [Siepman et al., ISSTA'94]
  - Reference Testing [Su et al., Software'91; Vogel, ISSTA'93]
Concluding Remarks

Derived a formal model of a GUI from its specifications and showed its usefulness in developing oracles
- General
- Portable
- Operator reuse - libraries
- Natural extension of our test-case generation technique