Coverage Criteria for GUI Testing

Atif M. Memon*, Mary Lou Soffa, Martha E. Pollack
atif@cs.umd.edu
Dept. of Computer Science
University of Pittsburgh

*now at the University of Maryland

Research focus

50% of code
GUI
Interactions between the GUI and the underlying code
Underlying Code
GUI Test Case

- Sequence of Events
  - [IEEE TSE Feb '01]
- Not just individual events

Coverage Criteria

- Two purposes
  - Test data selection criteria
    - Rules used to select test cases
  - Test data adequacy criteria
    - Rules used to determine how much testing has been done
- Common Examples for Conventional Software
  - Statement coverage
  - Branch coverage
  - Path coverage
  \[
  \text{Structural Representation of the Code}
  \]
Coverage Criteria for GUIs

- Cannot use code-based coverage
  - Source code not always available
  - Event-based input
    - Different level of abstraction

- Our Contribution
  - Hierarchical structure of the GUI in terms of events
  - Coverage criteria based on events
Outline

- GUI Definition
- Representation of GUIs
- Coverage Criteria
- Case Study
- Conclusions

GUI Definition

- Hierarchical
- Graphical Front-end
- Accepts User-generated and System-generated events
- Fixed sets of events
- Deterministic Output
- State of the GUI is the set of Objects and their Properties
GUI Representation

• Motivation
  - GUI testing needs a “Unit of Testing”
    • Manageable
    • Test the unit comprehensively
    • Test interactions among units
  - GUIs are created using library elements
    • Need to test these elements before packaging them for reuse
      - Certain level of confidence that the element has been adequately tested
    • User of these elements should be able to test the element in its context of use

Model GUI Hierarchically

• Hierarchy
  - GUIs are decomposed into a hierarchy of components
  - Hierarchical decomposition makes testing intuitive and efficient
  - Several hierarchical views of GUIs
  - We examine Modal Dialogs to create the hierarchical model
Modal Windows in GUIs

Main

invokes

Modal Windows in GUIs

Main

Print
Definition: Integration tree is a triple \( \langle N, R, B \rangle \)

- \( N \) is the set of components in the GUI
- \( R \subseteq N \) is a designated component called the **Main** component
- \( B \) is the set of directed edges showing the invokes relation between components, i.e., \( (C_x, C_y) \in B \) if and only if \( C_x \) invokes \( C_y \).
**Definition:** Event $e_x$ follows $e_y$ iff $e_x$ can be performed immediately after $e_y$.

**Definition:** Event-flow graph is a 4-tuple $\langle V, E, B, I \rangle$
- $V$ is the set of vertices, representing events,
- $E$ is the set of directed edges, showing the follows relationship,
- $B$ is the set of events first available (shown in red),
- $I$ is the set of events that invoke other components (dotted lines).
Classifying Events

Classification

- A new classification of events aids in creating the hierarchical model of the GUI
  - Opening modal windows
    - Restricted-focus events
  - Closing modal windows
    - Termination events
  - Opening modeless windows
    - Unrestricted-focus events
  - Opening menus
    - Menu-open events
  - Interacting with underlying software
    - System-interaction events

Coverage Criteria

- Intuitively
  - Each component is a unit of testing
  - Test events within each component
    - Intra-component coverage criteria
  - Test events across components
    - Inter-component coverage criteria
Coverage Criteria

• **Intra-component Coverage**
  - Event coverage
    • Individual events
    • Each node in the event-flow graph
  - Event-interaction coverage
    • Each pair of events
    • Each edge in the event-flow graph
  - Length-n event sequence coverage
    • Sequences of events
    • Bounded by length
      - Length-1 event sequences
      - Length-2, length-6 event sequences
    • Paths in the event-flow graph

• **Inter-component Coverage**
  - Invocation coverage
    • Invoke each component
    • Each restricted-focus event
  - Invocation-termination coverage
    • Invoke each component and terminate it
    • Restricted-focus event followed by a termination event
  - Inter-component length-n coverage
    • Longer sequences from one component to another
    • Bounded by length
Case Study

• Purpose
  – To determine:
    • How many test cases do we need to test WordPad
    • Correlation between event and code-based coverage
    • How well did our planning-based approach [ICSE ’99] do

• Experimental design
  – GUI: our version of MS WordPad (36 modal windows, 362 events)
  – Hardware platform: 350 MHz Pentium based machine, 256 MB RAM

Test Cases for WordPad

<table>
<thead>
<tr>
<th>Component Name</th>
<th>Event-sequence Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Main</td>
<td>56</td>
</tr>
<tr>
<td>FileOpen</td>
<td>10</td>
</tr>
<tr>
<td>FileSave</td>
<td>12</td>
</tr>
<tr>
<td>Print</td>
<td>14</td>
</tr>
<tr>
<td>Print+Properties</td>
<td>11</td>
</tr>
<tr>
<td>Print+Properties</td>
<td>9</td>
</tr>
<tr>
<td>Main+FileOpen</td>
<td>12</td>
</tr>
<tr>
<td>Main+FileSave</td>
<td>10</td>
</tr>
<tr>
<td>Main+PageSetup</td>
<td>11</td>
</tr>
<tr>
<td>Main+FormatFont</td>
<td>9</td>
</tr>
<tr>
<td>Main+Print+Properties</td>
<td>12</td>
</tr>
</tbody>
</table>
Correlation between Event-based & Code-based Coverage

- Code Instrumentation
- Generated all event sequences up to length 3. Total test cases: 21,659
- Executed all 21,659 cases and obtained execution traces
- Statement coverage

Results
Evaluating the Planning Approach

- Used our earlier-developed planning-based approach
- 500 test cases of different lengths

<table>
<thead>
<tr>
<th>Component Name</th>
<th>Event-sequence Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1st</td>
</tr>
<tr>
<td>Main</td>
<td>88</td>
</tr>
<tr>
<td>FileOpen</td>
<td>90</td>
</tr>
<tr>
<td>FileSave</td>
<td>90</td>
</tr>
<tr>
<td>Print</td>
<td>92</td>
</tr>
<tr>
<td>Properties</td>
<td>92</td>
</tr>
<tr>
<td>PageSetup</td>
<td>91</td>
</tr>
<tr>
<td>FormatFont</td>
<td>89</td>
</tr>
<tr>
<td>Print+Properties</td>
<td>100</td>
</tr>
<tr>
<td>Main+FileOpen</td>
<td>100</td>
</tr>
<tr>
<td>Main+FileSave</td>
<td>100</td>
</tr>
<tr>
<td>Main+PageSetup</td>
<td>100</td>
</tr>
<tr>
<td>Main+FormatFont</td>
<td>100</td>
</tr>
<tr>
<td>Main+Print+Properties</td>
<td>50</td>
</tr>
</tbody>
</table>
Future Work

• GUI’s Structure and its Testability
• Apply Criteria to
  - Object-oriented Software
  - Component-based Software
  - Reactive Software