1

Regression Testing of GUIs

Atif M. Memon
atif@cs.umd.edu

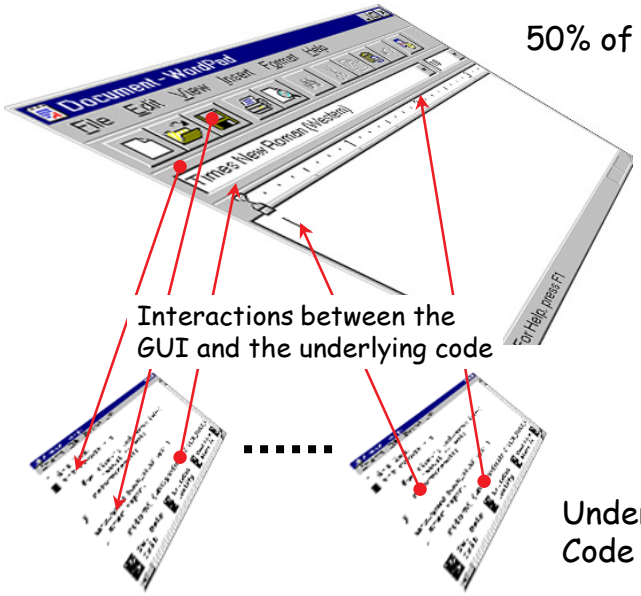
Dept. of Computer Science
&
Fraunhofer Center for Empirical Software Engineering
University of Maryland

Mary Lou Soffa
soffa@cs.pitt.edu

Dept. of Computer Science
University of Pittsburgh

2

Graphical User Interfaces



50% of code

GUI

Interactions between the GUI and the underlying code

Underlying Code

New Event-flow Model

3

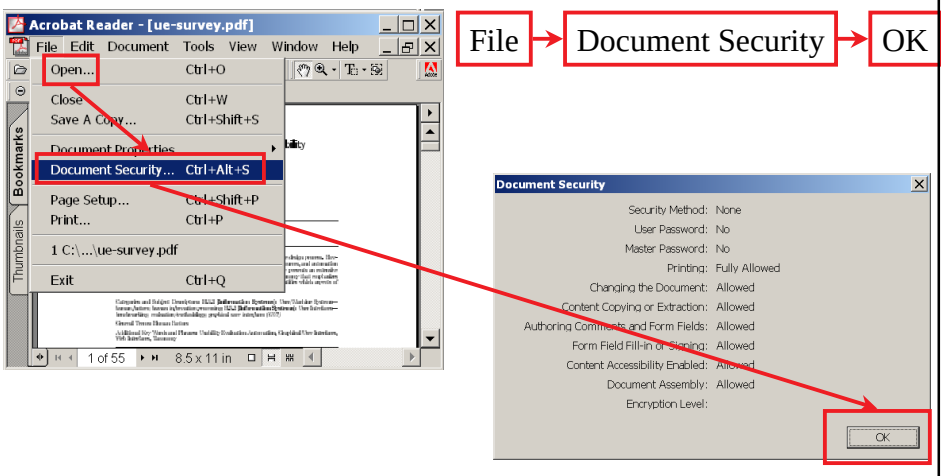
- Event-flow of a GUI
  - User interactions are sequences of events
  - Test case is also modeled as a sequence of events (TSE 2001, ICSE 1999)
  - Test oracles also at event level (FSE 2000, ASE 2003)
  - Test coverage defined in terms of events (FSE 2001)
- Enabled successful development of tools and techniques to test GUIs
  - Now a new ***Regression Testing*** technique based on the event-flow model

GUI Regression Testing Problem

4

Adobe Acrobat 5.0

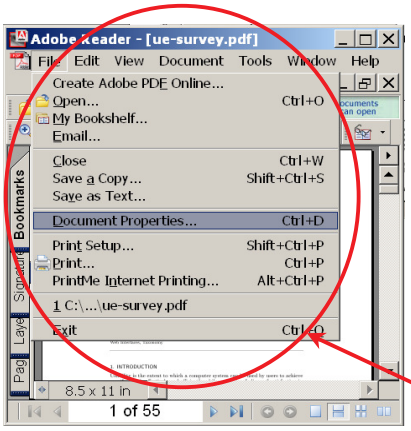
A test case of length 3



GUI Regression Testing Problem

5

Adobe Acrobat 6.0



A test case of length 3

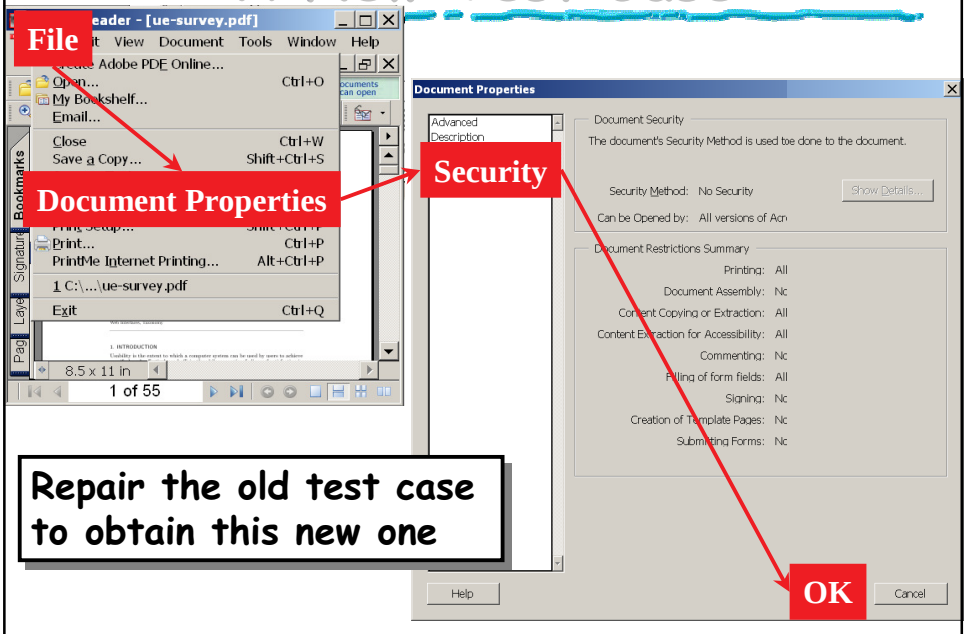
File → Document Security → OK

Event "Document Security" no longer in menu

Test case cannot be executed!!!

A New Test Case

6



Repair the old test case to obtain this new one

Outline

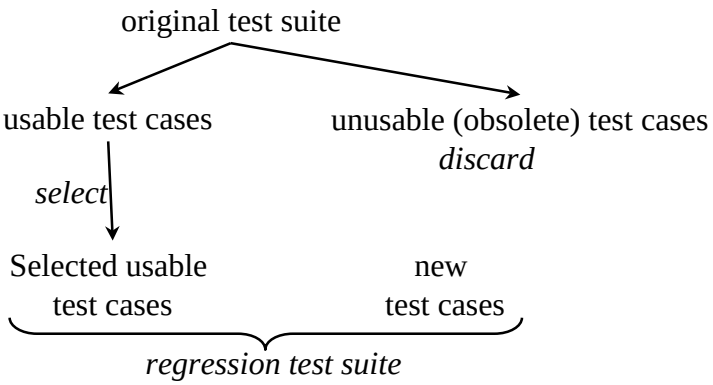
7

- Traditional regression testing
- Creating GUI test cases
- Event-flow model
- Repairing test cases
- Case studies
- Conclusions and future work

Traditional Regression Testing

8

- Retest-all
- Selective regression testing process



Challenges for GUIs

9

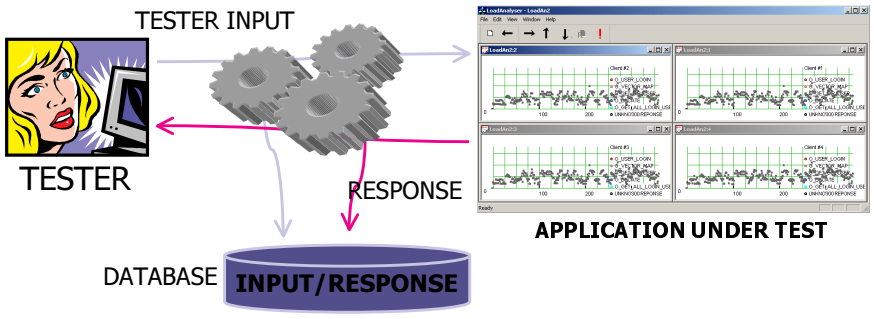
- Test case generation is manual
  - Small number of test cases
  - Each test case is valuable
- Changes result in a large number of obsolete test cases
  - Structural and layout changes
- Frequent modifications
  - Driven by constant user feedback
  - Need frequent testing

Capture/replay tools

10

- Tools for generating GUI test cases

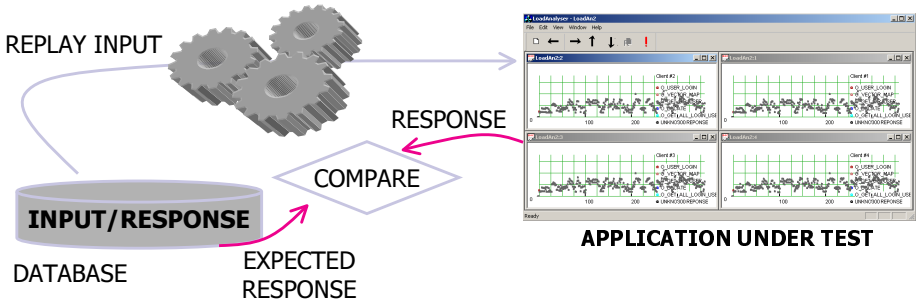
Capture
✓ User **MANUALLY** performs events on the GUI
✓ Tool records all user inputs and application response



Replay

11

Replay
- ✓ Tool replays users actions
- ✓ Modifications made to enhance testing of the GUI
- ✓ Verify application response against expected response



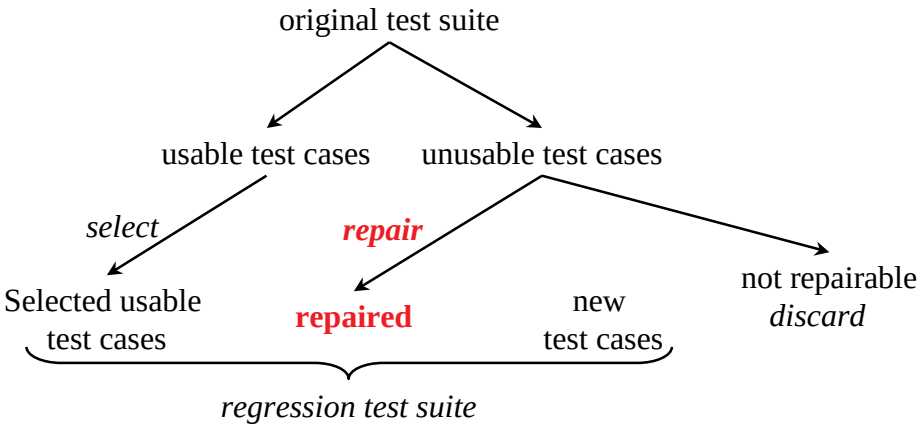
Our Approach

12

- Typical test cases
  - 10 to 50 events
  - 5 to 10 minutes to create
- For a software
  - Tester may create 200-300 test cases
  - Each test case is valuable
- Approach
  - Repair unusable test cases

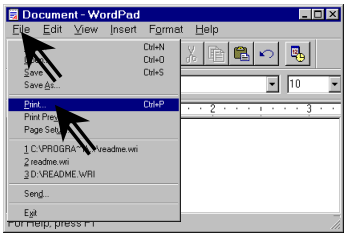
GUI Regression Testing

13



The Event-flow Model

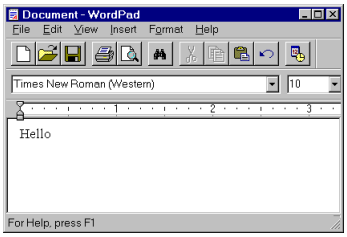
14

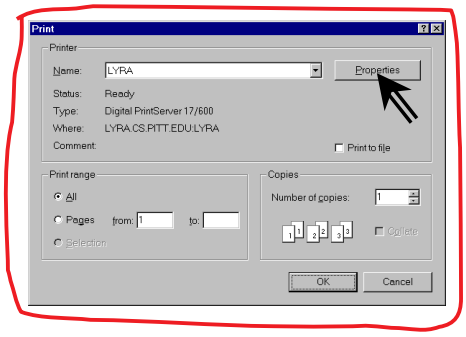


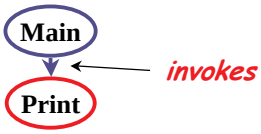
Main

Modal Windows in GUIs

15

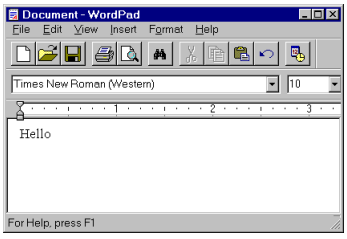


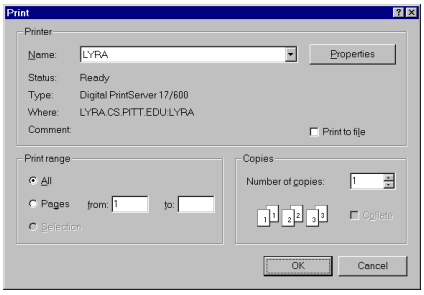


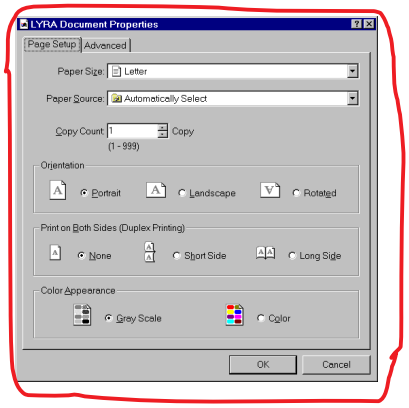


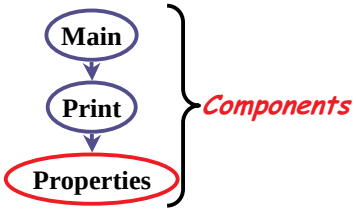
Modal Windows in GUIs

16

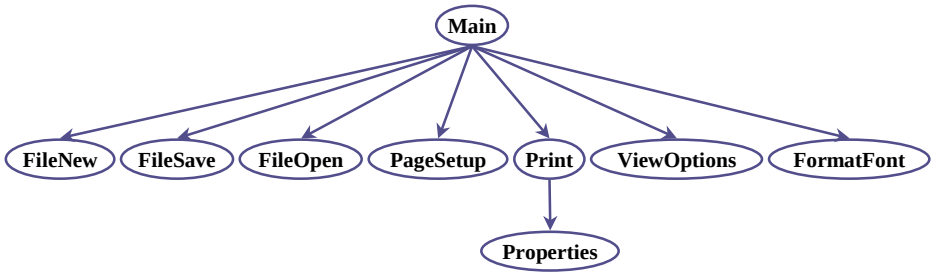








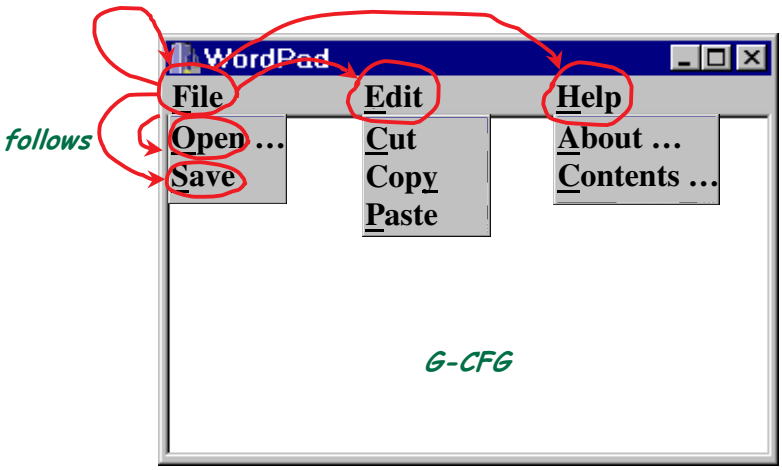
G-call Graph



Definition: G-call Graph is a triple $\langle N, R, B \rangle$

- N is the set of components in the GUI
- $R \in N$ is a designated component called the ***Main*** component
- B is the set of directed edges showing the invokes relation between components, i.e., $(C_x, C_y) \in B$ iff C_x invokes C_y.

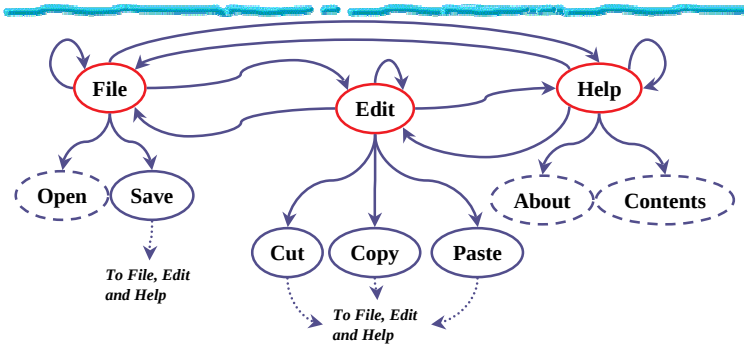
Representing a Component



Definition: Event e_x follows e_y iff e_x can be performed immediately after e_y.

G-CFG

19



Definition: G-CFG is a 4-tuple <*V, E, B, I*>

- *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

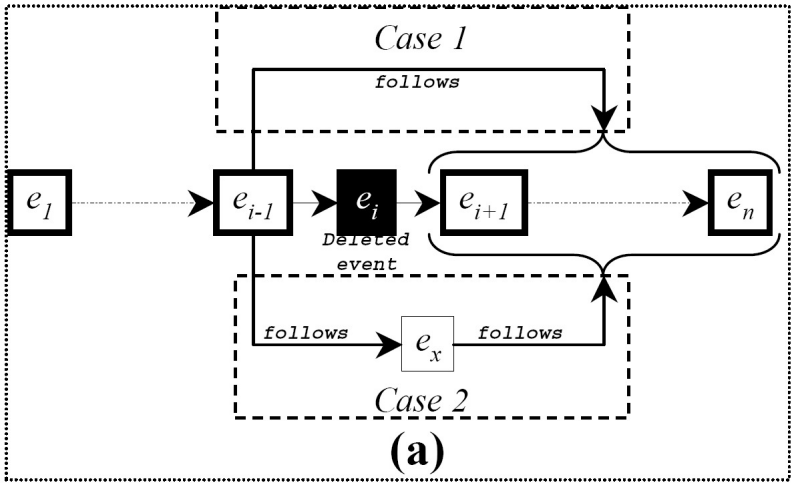
20

- 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

The Repairing Process

21

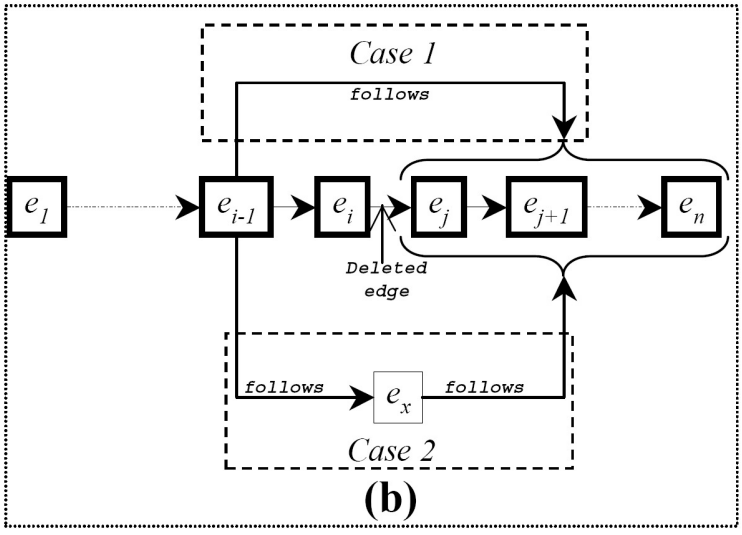
- A deleted event



The Repairing Process

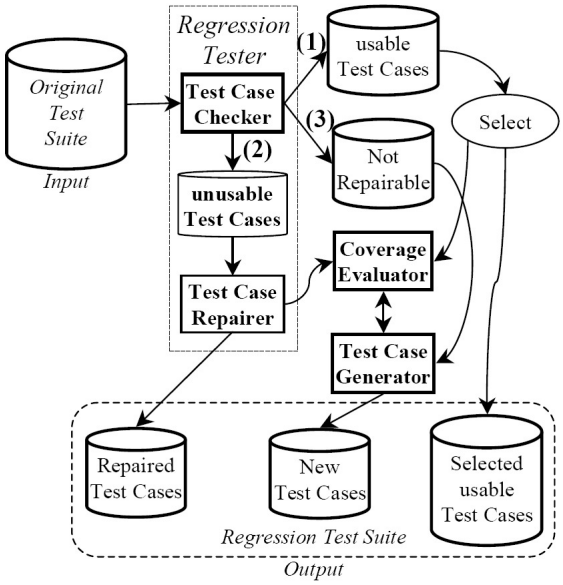
22

- A deleted edge



Putting it All Together

23



Case Studies

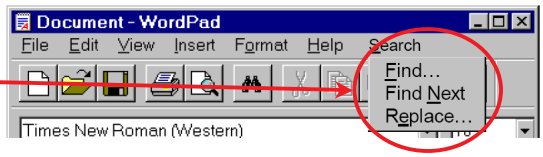
24

- Questions
  - How many test cases are made unusable by GUI modifications across versions?
  - How many unusable test cases are repairable?
  - How much time does the checker and repairing processes take?

Subject Applications

- Two programs
  - Each with two versions
    - Adobe's Acrobat Reader
      - version 4.0 (for Linux)
        - » 15 components with 176 events (not counting short-cuts)
      - version 5.0 (for MS Windows 2000)
        - » 25 components with 351 events
    - our own implementation of MS WordPad
      - 36 modal windows, and 362 events (not counting short-cuts)

Menu added to version 2



Case Study 1

- Performance and effectiveness of the regression testing technique
  - 400 test cases generated manually using a capture/replay tool for each subject application
    - Adobe's Acrobat Reader = 7.59 hours
    - changes in the GUI made 296 (74%) test cases unusable
    - time taken for the checker was 6.5 sec
    - remaining 104 (26%) test cases were usable
    - repaired 211 (71.3%) of unusable test cases
    - total time taken for repairer was 17.76 sec.

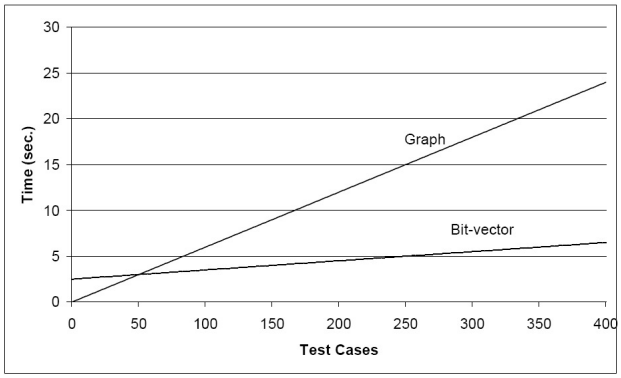
Case Study 1

- Comparable results for Wordpad
  - modifications affected 210 test cases
  - checker took 6.15 sec
  - repaired all 210 test cases
    - all that was needed was to replace Edit with Search in each
  - time taken was 18.01 sec

Step	Subject Application	Time
Manual Generation	Reader	7.59 hrs.
	WordPad	5.47 hrs.
Checker	Reader	6.5 sec.
	WordPad	6.15 sec.
Repairer	Reader	17.76 sec.
	WordPad	18.01 sec.

Case Study 2

- Two different types of checkers
  - Simple, graph traversal based
  - We implemented a bit-vector based algorithm



Conclusions & Future Work

- Repairing "obsolete" test cases
- Detailed experiments
  - Different types of test cases
  - Many subject applications
- Use approach for other event-based software
- Repair obsolete test cases for conventional software