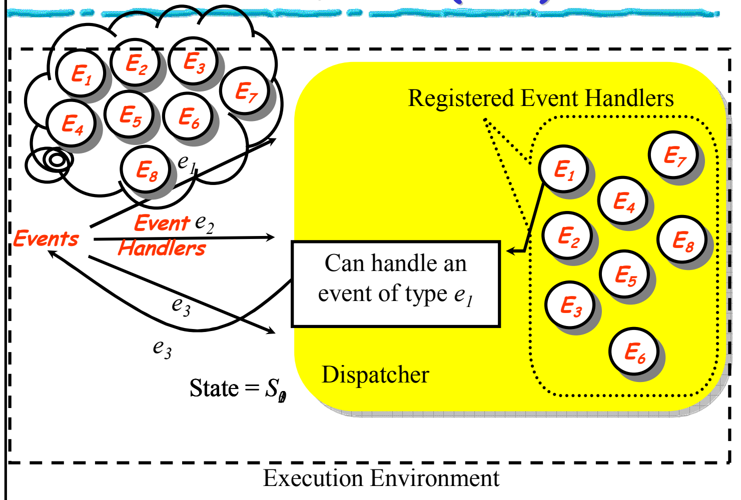
Automated Model-Based Testing of Event-driven Software Applications

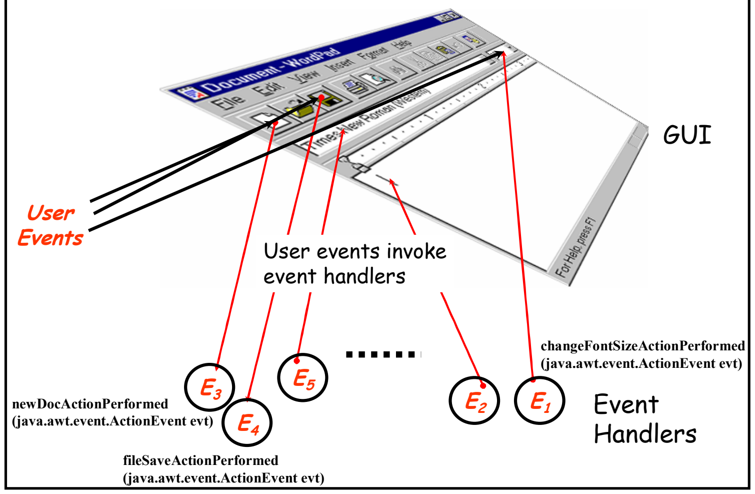
Atif M. Memon
atif@cs.umd.edu

Department of Computer Science
&
Institute for Advanced Computer Studies
University of Maryland

Event-driven Software (EDS) in Action!



GUIs are Event-Driven Software


Document - WordPad
File Edit View Insert Format Help
GUI
User Events
User events invoke event handlers
changeFontSizeActionPerformed (java.awt.event.ActionEvent evt)
newDocActionPerformed (java.awt.event.ActionEvent evt)
fileSaveActionPerformed (java.awt.event.ActionEvent evt)
E3
E4
E5
E2
E1
Event Handlers

Choose Your Favorite EDS!

- Graphical-user interfaces
- Web applications
- Network protocol implementations
- Middleware
- Object-oriented software

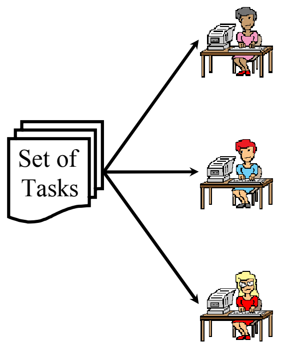
- Robots - man-machine interfaces
- Multi-agent based systems

Focus on GUIs

- Simple model of an event
- A user action
  - click-on-File-menu,
  - click-on-OK-button,
  - type-in-textbox()
- Complex interactions
- Large space of event interactions
  - Number grows exponentially with length

•GUI Testing: Pitfalls and Process, Atif M. Memon, ***IEEE Computer***, vol. 35, issue. 5, 2002.
•Advances in GUI Testing, Atif M. Memon, ***Highly Dependable Software, (M. V. Zelkowitz ed.), Advances in Computers***, Academic Press, vol. 58, pp. 149-201, 2003.

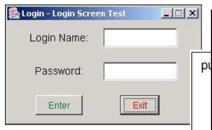
State of the Practice – Manual



- Very few test cases
- Oracle: mostly visual
- Test "common" sequences
  - Bad Idea
    - What is "common"?
- Try some "uncommon" sequences

- Test cases not reusable
  - Must do it again when app changes

State of the Practice – Code Tests

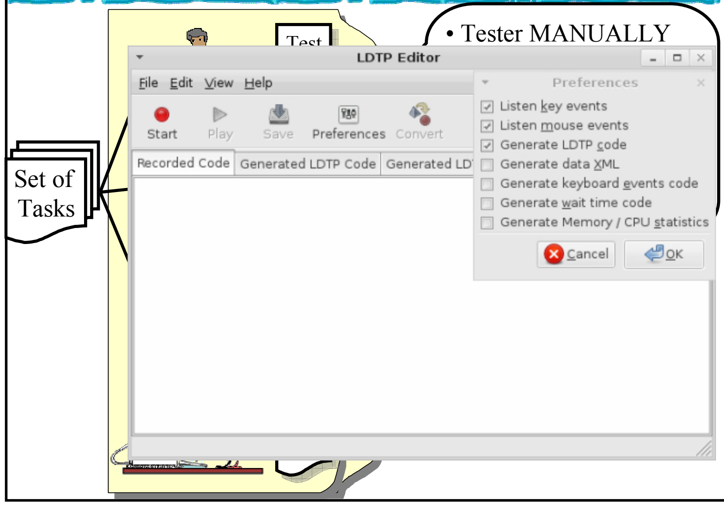


```
def cb ():
    callbackState.set ()
    waittillguiexist ('dlgReplace')
    click ('dlgReplace', 'btnClose')
    callbackState.clear ()
    callbackRunning.set ()
    print 'callbackend'
onwindowcreate ('Replace', cb)
click ('*gedit', 'btnReplace')
click ('*gedit', 'btnOpen')
waittillguiexist ('dlgOpenFiles...')
click ('dlgOpenFiles...', 'btnClose')
if callbackState.isSet ():
    print 'Waiting for callback to
    complete'
    callbackRunning.wait ()
    print 'callbackset'
print 'test end'
```

JFCUnit
- Can be replayed automatically
- Multiple machines
- Regression testing: GUI evolves
- Oracle: what to check?
- Still have "Few tests" problem

Several other tools
- LDTP
- GUITAR

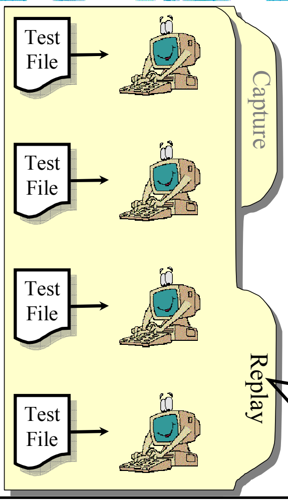
State of the Practice – Capture/Replay



• Tester MANUALLY

- [✓] Listen key events
- [✓] Listen mouse events
- [✓] Generate LDTP code
- [ ] Generate data XML
- [ ] Generate keyboard events code
- [ ] Generate wait time code
- [ ] Generate Memory / CPU statistics

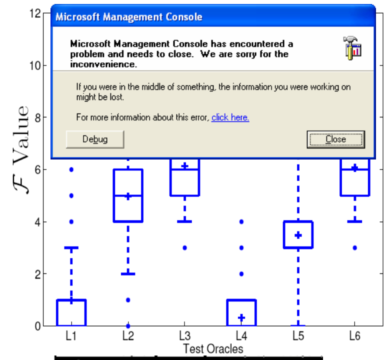
State of the Practice – Capture/Replay



- Tool replays tester's actions
- **Verifys application response against stored response**

•**DEMO**

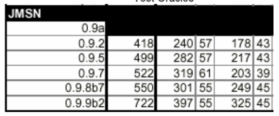
State of the World


Microsoft Management Console

Microsoft Management Console has encountered a problem and needs to close. We are sorry for the inconvenience.

If you were in the middle of something, the information you were working on might be lost.

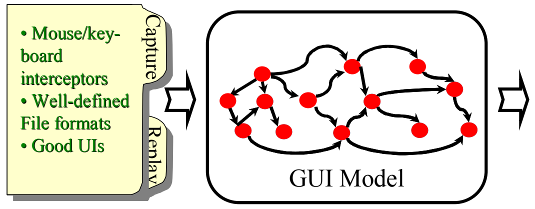
For more information about this error, click here.



JMSN					
0.9a					
0.9.2	418	240	57	178	43
0.9.5	499	282	57	217	43
0.9.7	522	319	61	203	39
0.9.8b7	550	301	55	249	45
0.9.9b2	722	397	55	325	45

These problems persist
- Regression testing: GUI evolves
  - "Low ROI"
- Oracle: what to check?
  - "Tool made us miss this error"
- Too few tests
  - "But you shouldn't have done that"
  - DEMO
- More philosophical question remains
  - How to test GUIs and other large event-driven applications?

Our Approach



- Start with capture/replay tools
  - Well-engineered
  - Test management

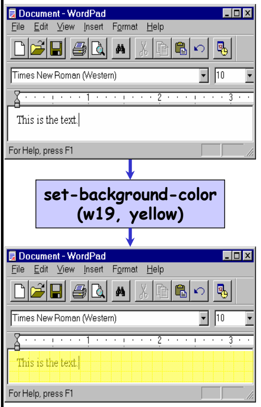
- (widgets, value, property) triples
- Pre/postconditions
  - AI planning
- Directed graph models
  - Event-flow graph
  - Event-interaction graph
- Covering arrays
- Probabilistic state machines

- What is the cost of creating and maintaining the model?

AI Planning

- Create planning operators
  - Pre- postconditions for each event
- AI planner generates test cases
- Application of postconditions creates test oracle
- For modified GUI
  - Change operators
  - Replanning

Operator Example


Document - WordPad
File Edit View Insert Format Help
Times New Roman (Western)
10
This is the text.
For Help, press F1
set-background-color (w19, yellow)
Document - WordPad
File Edit View Insert Format Help
Times New Roman (Western)
10
This is the text.
For Help, press F1

Operator :: *set-background-color*
Parameters: wX: window; col: color;
Precondition:
    isCurrent(wX),
    background-color(wX, oldColor),
    oldColor != col.

Effects:
    background-color(wX, col).

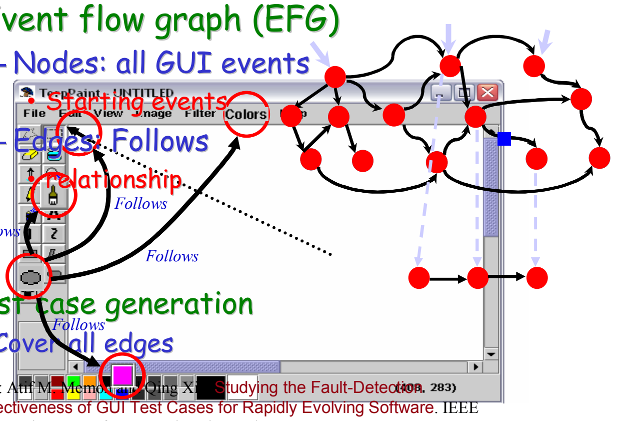
- Huge investment in operator creation and maintenance
  - Works well for small number of events

Directed Graph Models

- Model the space of GUI interactions as a graph
  - i.e., given a GUI, create a graph model of all the possible sequences that a user can execute
  - Use the model to generate event sequences

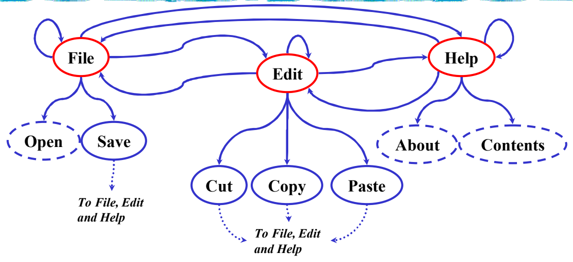
Sampling The Event-Interaction Space

- Event flow graph (EFG)
  - Nodes: all GUI events
    - Starting events
  - Edges: Follows
    - relationship
- Test case generation
  - Cover all edges



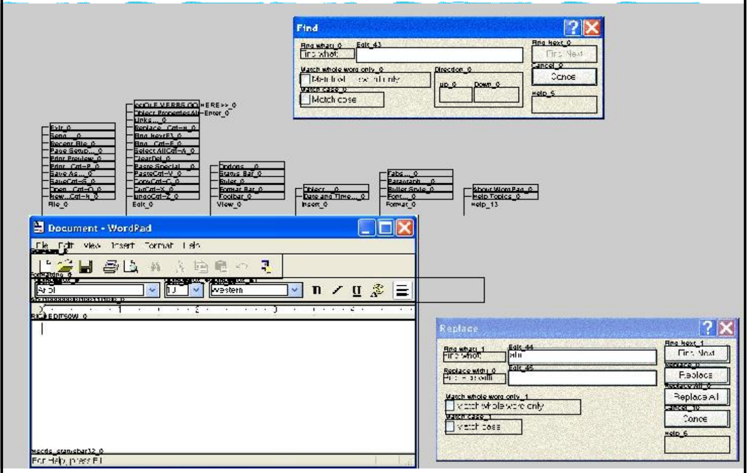
See: Atif M. Memon and Qing Xie. Studying the Fault-Detection Effectiveness of GUI Test Cases for Rapidly Evolving Software. IEEE Transactions on Software Engineering, vol. 31, no. 10, 2005, pp. 884-896.

Creating Event-Flow Graphs

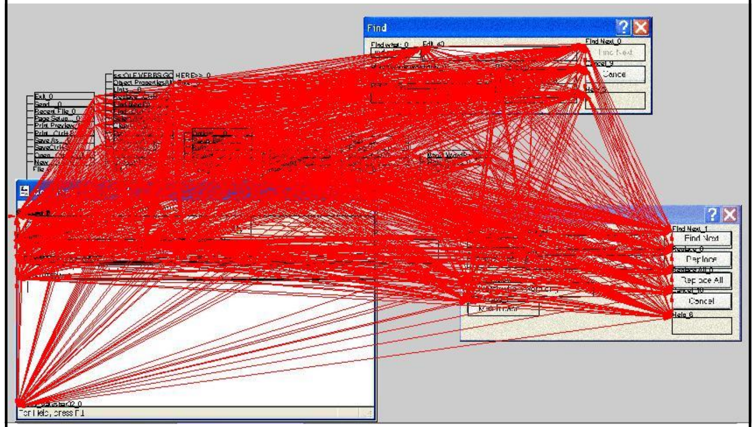


- How to create the event-flow graph?
  - Manually?
    - Too large for non-trivial GUIs

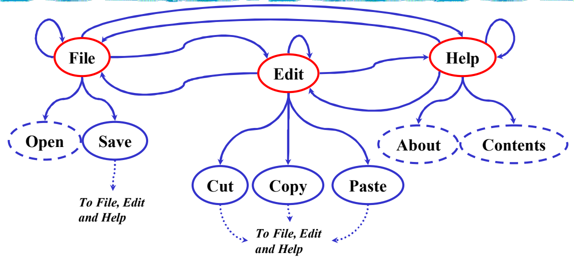
A Part of MS WordPad



Its Event-Flow Graph



Automation



- How to create the event-flow graph?
  - Automatically
    - Reverse Engineering
      - GOAL: Obtain the Event-Flow Graph
      - Fully automatically
      - No source code

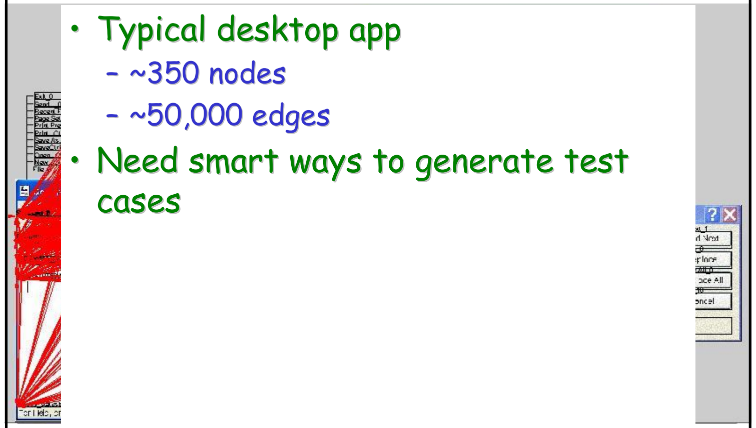
Reverse Engineering – GUI Ripping

- Dynamic algorithm
  - No need for source code
- Execute the GUI-based software
  - Traverse the GUI
    - Obtain handle of first window
    - Use windowing API to extract widgets/menus
- Apply transformations
  - Based on GUI dialogs
  - GUI hierarchy
  - Enabled/disabled widgets
- Traverse multiple times if needed

- Engineering Issues
  - Understanding platform-specific GUI frameworks
    - OS-specific GUI handling
  - Introspection
  - Windowing API
  - Java Swing API
  - Interaction between Java and the OS
- Result – Generic process for GUI Ripping
- MS Windows, Java Swing
- Immediate impact – Obtained EFGs for large GUIs in a few minutes
- DEMO

GUI Ripping: Reverse Engineering of Graphical User Interfaces for Testing, Atif M. Memon, Ishan Banerjee*, and Adithya Nagarajan*, ***Proceedings of the IEEE 10th Working Conference on Reverse Engineering***, pp. 260-269, Nov. 13-16 2003.

Impact on others' research: "design mentoring" based on evolution analysis; introspective approach to "marking" GUIs; unsupervised user modeling

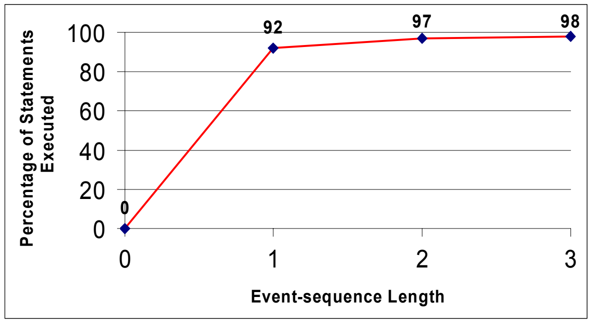


- Typical desktop app
  - ~350 nodes
  - ~50,000 edges
- Need smart ways to generate test cases

Impact of GUI Ripper

- A way to generate test cases for large GUIs
  - Examine execution results to better understand the nature of GUI software
- Enabled experimentation
  - Study the characteristics of test cases
  - Reduce the event-flow graph
    - represent "important" interactions
- Developed "event-space exploration strategies" (ESES)
  - E.g., "Repairing" test cases for regression testing

Nature of GUI Software

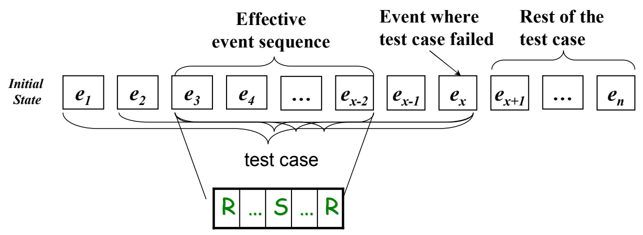


- Showed that length 1 and 2 event sequences detect faults
- But certain length 3 (and more) sequences detect additional faults
- Although they do not add much to code coverage
  - One of the first to show that EDS (at least GUIs) require different testing techniques

Enabled Experimentation

- Generate large numbers of test cases
  - Various types
    - Random
    - Event-flow graph edge adequate
    - Code-coverage adequate
    - Covering arrays
- Millions of test cases
  - 120 machine cluster
  - CONDOR jobs on UMIACS clusters
- Study the execution results and improve testing techniques

Dissecting Failed Test Cases



R = reaching events that open menus/windows
W = events that open windows
T= termination events that close windows
S = system-interaction events (e.g., CUT, COPY, PASTE)

Understanding the Effective Event Sequence

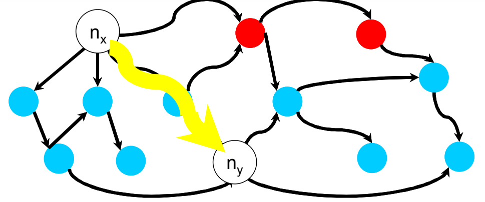
Pattern	Effective Event Sequence Structure	e_x	# Failures
1	R*	S	676
		W	6
2	R*S	S	431
		W	1
3	R*SR+	S	19
4	R*SR*(SR*)+	S	142

R = reaching events that open menus/windows
W = events that open windows
T= termination events that close windows
S = system-interaction events (e.g., CUT, COPY, PASTE)

Generate these effective sequences automatically

Definitions

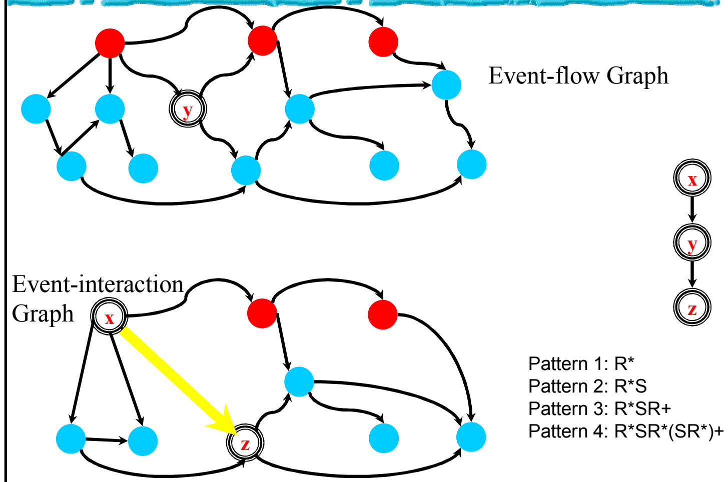
- **Definition:** An event-flow-path $< n_1; n_2; \ldots; n_k >$ is *interaction-free* iff none of $n_2, \ldots, n_{k-1}$ represent system-interaction events. □



- **Definition:** A system-interaction event e_x *interacts-with* system-interaction event e_y iff there is at least one interaction-free event-flow-path from the node n_x (that represents e_x) to the node n_y (that represents e_y). □

Studying the Fault-Detection Effectiveness of GUI Test Cases for Rapidly Evolving Software, Atif M. Memon and Qing Xie*, ***IEEE Transactions on Software Engineering***, IEEE Computer Society Press, vol. 31, no. 10, pp. 884-896, Oct. 2005.

Event Interaction Graph



Pattern 1: R*
Pattern 2: R*S
Pattern 3: R*SR+
Pattern 4: R*SR*(SR*)+

Event-interaction Graph (EIG)

- Event-interaction graphs
  - Higher level of abstraction than event-flow graphs
  - Edges represent longer "important" paths in the GUI
- New test adequacy criteria
  - Event-flow graph interaction-free path coverage
  - Event-interaction graph edge coverage

•"Using a Pilot Study to Derive a GUI Model for Automated Testing," by Qing Xie and Atif M. Memon, ACM Trans. on Softw. Eng. and Method.
•Agile Quality Assurance Techniques for GUI-Based Applications, Qing Xie* and Atif M. Memon, ***Agile Software Development Quality Assurance***, to appear 2007.
•Rapid 'Crash Testing' for Continuously Evolving GUI-Based Software Applications, Qing Xie* and Atif M. Memon, ***Proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM 2005).***

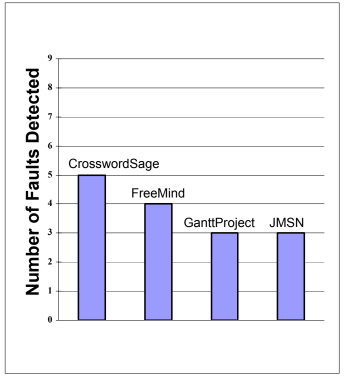
Full Automation

- Process
  - Reverse engineer application
  - Generate event-flow graph
  - Transform to event-interaction graph
  - Use our new test-adequacy criteria to generate test cases (e.g., cover all edges – important sequences of events in a GUI)
  - Use test executor to run all test cases
- Test Oracle
  - Assertions in the code
  - Invariants - Diakon
  - "Did the application crash?"

Automated Model-based Testing of Community-Driven Open Source GUI Applications, Qing Xie* and Atif M. Memon, ***Proceedings of the 22nd IEEE International Conference on Software Maintenance (ICSM 2006)***.

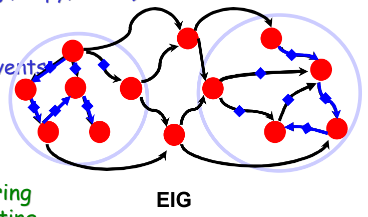
Lets See How It Works!

- Point to the CVS head
  - Push the button
  - Read error report
- What happens
  - Gets code from CVS head
  - Builds
  - Reverse engineers the event-flow graph
  - Creates EIG
  - Generates test cases to cover all the edges
    - 2-way covering
  - Runs them
- SourceForge.net
  - Four applications



Digging Deeper!

- **Intuition**
  - Non-interacting events (e.g., Save, Find)
  - Interacting events (e.g., Copy, Paste)
- **Key Idea**
  - Identify interacting events
  - Mark the EIG edges (Annotated graph)
  - Generate 3-way, 4-way, … covering test cases for interacting events only



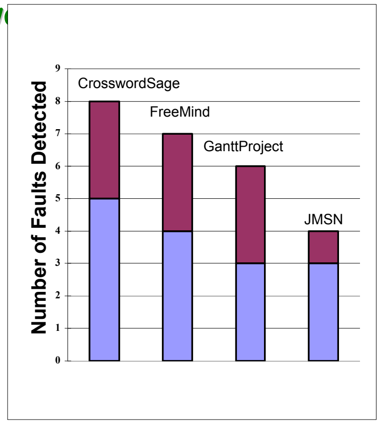
"Using GUI Run-Time State as Feedback to Generate Test Cases" by Xun Yuan and Atif M. Memon. In *ICSE '07: Proceedings of the 29th International Conference on Software Engineering*, May 23-25, 2007, pp. 396-405.

Identifying Interacting Events

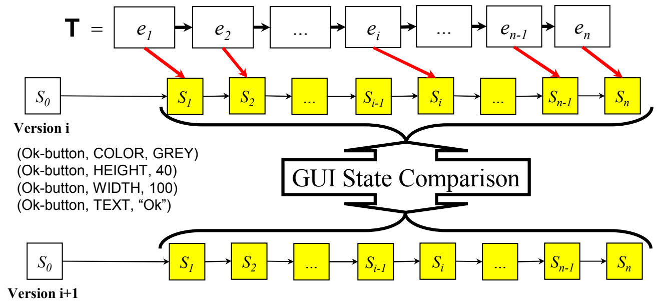
- High-level overview of approach
  - Observe how events execute on the GUI
  - Events interact if they influence one another's execution
    - Execute event e2; execute event sequence <e1, e2>
    - Did e1 influence e2's execution?
    - If YES, then they must be tested further; annotate the <e1, e2> edge in graph
- Use feedback
  - Generate seed suite
    - 2-way covering test cases
  - Run test cases
    - Need to obtain sets of GUI states
  - Collect GUI run-time states as feedback
  - Analyze feedback and obtain interacting event sets
  - Generate new test cases
    - 3-way, 4-way, … covering test cases

Did We Do Better?

- Compare feedback-based approach to 2-w



Test Oracle for Regression Testing



T = GUI test case of length *n*

e_i = i^{th} GUI event of test case

S_o = Initial State of the GUI

Empirical Evaluation of the Fault-detection Effectiveness of Smoke Regression Test Cases for GUI-based Software, Atif M. Memon and Qing Xie*, ***Proceedings of the 20th IEEE International Conference on Software Maintenance 2004 (ICSM 2004)***, Chicago, IL, USA, pp. 8-17, Sep. 11-17, 2004.

GUI Test Oracles from Specs

- For each event, develop
  - Pre-conditions
    - Necessary for an event to execute
    - E.g., (OK-button, Active, TRUE)
  - Effects
    - How the event changes the GUI
    - E.g., (FindWindow, isVisible, FALSE)
- Pre-conditions/effects checked during test execution

What Test Oracle Should I use for Effective GUI Testing? Atif M. Memon, Ishan Banerjee*, and Adithya Nagarajan*, ***Proceedings of the IEEE International Conference on Automated Software Engineering (ASE 2003)***, Montreal, Quebec, Canada, pp. 164-173, Oct. 6-10 2003.

Mixing and Matching

- Test-case Generation Criteria
  - Cover all event-interaction graph edges
  - Cover n-way event interactions

- Test-oracles
  - Check for crashes
  - Compare to previous version
  - Use pre-conditions/effects

Testing Technique

Comprehensive
Crash Testing
Smoke Testing
GUI Testing

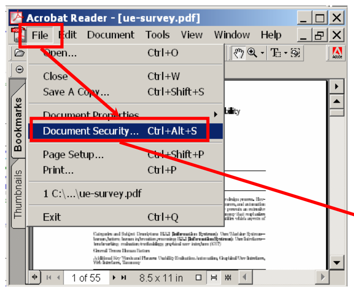
Studied characteristics of faults and GUI tests
Extending the work – Jaymie Strecker (current PhD student)

Empirically compared test oracles for effectiveness
•First such study

•Designing and Comparing Automated Test Oracles for GUI-based Software Applications, Qing Xie* and Atif M. Memon, *ACM Transactions on Software Engineering and Methodology*.
•Studying the Characteristics of a 'Good' GUI Test Suite, Qing Xie* and Atif M. Memon, ***Proceedings of the 17th IEEE International Symposium on Software Reliability Engineering (ISSRE 2006)***, Raleigh, NC, USA, Nov. 6-10 2006.

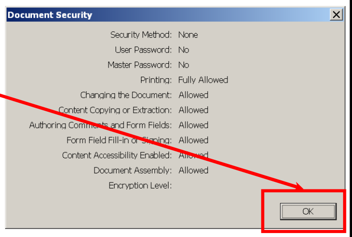
GUI Regression Testing Problem

Acrobat Reader 5.0


Acrobat Reader - [ue-survey.pdf]
File Edit Document Tools View Window Help
Open... Ctrl+O
Close Ctrl+W
Save A Copy... Ctrl+Shift+S
Document Properties
Document Security... Ctrl+Alt+S
Page Setup... Ctrl+Shift+P
Print... Ctrl+P
1 C:\...\ue-survey.pdf
Exit Ctrl+Q
Bookmarks
Thumbnails
1 of 55
8.5 x 11 in

A test case of length 3

$\boxed{\text{File}} \rightarrow \boxed{\text{Document Security}} \rightarrow \boxed{\text{OK}}$


Document Security

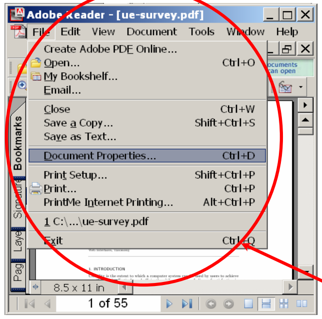
Security Method: None
User Password: No
Master Password: No
Printing: Fully Allowed
Changing the Document: Allowed
Content Copying or Extraction: Allowed
Authoring Comments and Form Fields: Allowed
Form Field Fill-in or Signing: Allowed
Content Accessibility Enabled: Allowed
Document Assembly: Allowed
Encryption Level:

OK

A Model-Based Approach to Automatically Repair GUI Test Cases for Regression Testing, Atif M. Memon, ***ACM Transactions on Software Engineering and Methodology.***

GUI Regression Testing Problem

Acrobat Reader 6.0


Adobe Reader - [ue-survey.pdf]
File Edit View Document Tools Window Help
Create Adobe PDF Online...
Open... Ctrl+O
My Bookshelf...
Email...
Close Ctrl+W
Save a Copy... Shift+Ctrl+S
Save as Text...
Document Properties... Ctrl+D
Print Setup... Shift+Ctrl+P
Print... Ctrl+P
PrintMe Internet Printing... Alt+Ctrl+P
1 C:\...\ue-survey.pdf
Exit Ctrl+Q
Bookmarks
8.5 x 11 in
1 of 55

A test case of length 3

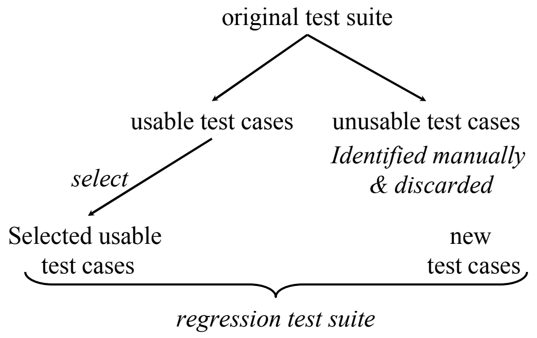
$\boxed{\text{File}} \rightarrow \boxed{\text{Document Security}} \rightarrow \boxed{\text{OK}}$

"Because GUI test cases are sequences of events, as many as 75% may become unusable"
Foundations of Software Engineering, 2003

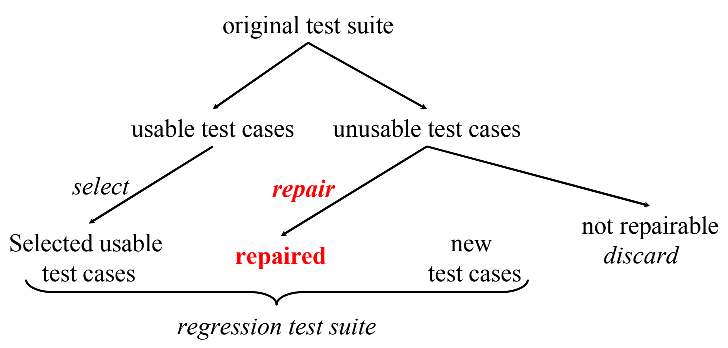
Event "Document Security" no longer in menu

Test case cannot be executed!!!

State of the Practice



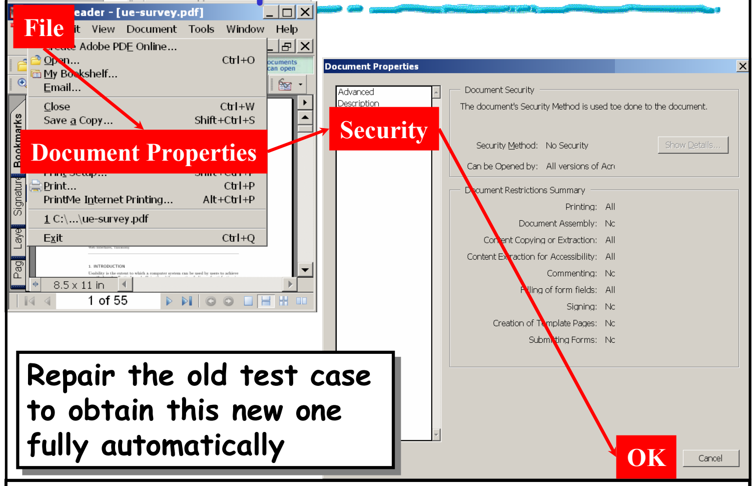
ESES to Repair Test Cases



Automated GUI Regression Testing Using AI Planning, Atif M. Memon, Artificial Intelligence Methods in Software Testing, (A. Kandel, H. Bunke and M. Last ed.), ***World Scientific Series in Machine Perception and Artificial Intelligence***, World Scientific Publishing Co., vol. 56, pp. 51-100, 2004.

First time test cases have ever been automatically repaired

A Repaired Test Case



Document Properties

Document Security

The document's Security Method is used toe done to the document.

Security Method: No Security

Can be Opened by: All versions of Acr

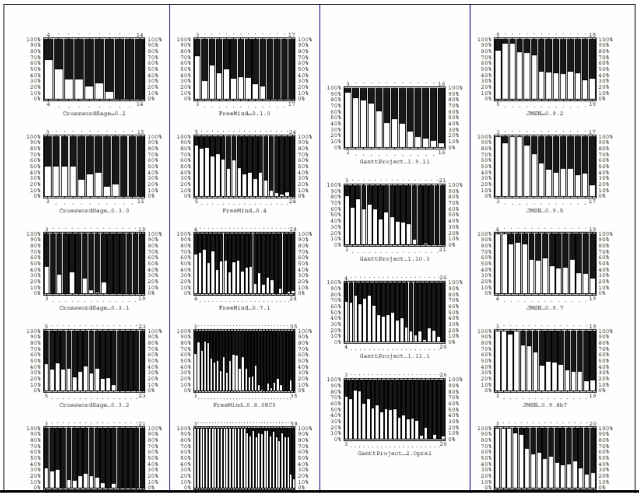
Document Restrictions Summary

Printing: All
Document Assembly: Nc
Content Copying or Extraction: All
Content Extraction for Accessibility: All
Commenting: Nc
Filling of form fields: All
Signing: Nc
Creation of Template Pages: Nc
Submitting Forms: Nc

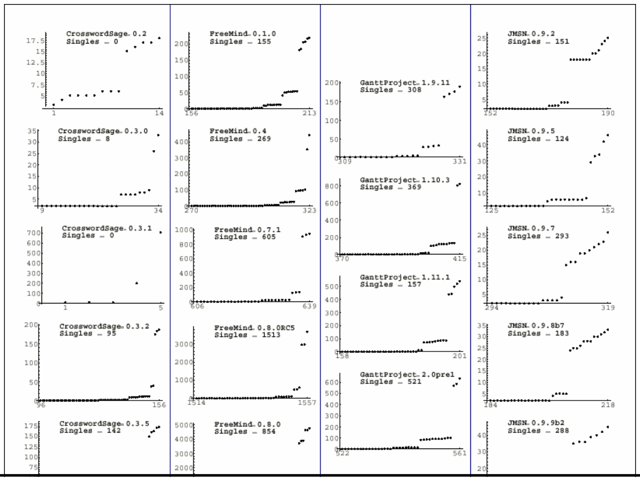
Repair the old test case to obtain this new one fully automatically

Impact on others' research: Repairing session data to improve web regression testing

Percentage of Test Cases that Remained Usable



Impact of Deleted Events on the Number of Unusable Test Cases



GUI Tools & Experimentation Subjects

- GUITAR
  - http://guitar.cs.umd.edu
- "Benchmarks" – TerpOffice & SourceForge Apps
- Six Terpoffice applications and six SourceForge applications
- For TerpOffice
  - Requirements and design documents
  - CVS history
  - 100's of Bug reports
  - 10000's of test cases; JUnit + GUITAR
  - Test oracles
  - 100's of fault seeded versions
  - Five versions (one per year)
  - CMSC 435 project is more realistic
  - Already used by other researchers
    - Static analysis (rpi.edu)
    - Interaction testing using covering arrays (unl.edu)
    - Prioritization using interaction coverage (umn.edu)
    - Studying GUI failures (ICSE 2005) (ncsu.edu)
    - Refactoring GUI code (waterloo.edu.ca)
- Shared process diagrams/artifacts

An Event-Flow Model of GUI-Based Applications for Testing, Atif M. Memon, ***Software Testing, Verification & Reliability****, John Wiley & Sons, Inc.*

Additional Contributions

- Getting to know GUI faults better
  - Jaymie Strecker (current PhD student)
  - **"Relationships Between Test Suites, Faults, and Fault Detection in GUI Testing"** by Jaymie Strecker and Atif M. Memon. In *ICST '08: Proceedings of the First international conference on Software Testing, Verification, and Validation, 2008.*
  - **"Faults' Context Matters"** by Jaymie Strecker and Atif M. Memon. In *Proceedings of The Fourth International Workshop on Software Quality Assurance (SOQUA '07).*
- Transient and persistent failures
  - "Smart" light-weight test oracles
  - Using Transient/Persistent Errors to Develop Automated Test Oracles for Event-driven Software, Atif M. Memon and Qing Xie*, ***Proceedings of the 19th IEEE International Conference on Automated Software Engineering 2004 (ASE 2004)***, Linz, Austria, pp. 186-195, Sep. 20-24, 2004.
- Employ GUI user profiles for testing
  - Annotating the edges of event-flow graphs
    - Already applied to GUI-component testing
    - Employing User Profiles to Test a New Version of a GUI Component in its Context of Use, Atif M. Memon, ***Software Quality Journal***, Springer Inc.
  - N-gram approach
    - Penelope Brooks (current PhD student)
    - "Automated GUI Testing Guided by Usage Profiles" by Penelope Brooks and Atif M. Memon. In ASE '07: Proceedings of the 22nd IEEE international conference on Automated software engineering, 2007.

Additional Contributions (contd…)

- Combinatorial techniques
  - "Covering Array Sampling of Input Event Sequences for Automated GUI Testing" by Xun Yuan, Myra Cohen. and Atif M. Memon, in ASE '07: Proceedings of the 22nd IEEE international conference on Automated software engineering, 2007.
  - "Test Suite Prioritization by Interaction Coverage" by Renee C. Bryce and Atif M. Memon. In Proceedings of The Workshop on Domain-Specific Approaches to Software Test Automation (DoSTA 2007.
- New model of components for improved testability
  - A Process and Role-Based Taxonomy of Techniques to Make Testable COTS Components, Atif M. Memon, ***Testing Commercial-off-the-shelf Components and Systems***, (S. Beydeda and V. Gruhn ed.), Springer, pp. 109-140, 2004.
- New testing criteria
  - Call-stack coverage
    - Scott McMaster (current PhD student)
    - **"Call-Stack Coverage for GUI Test-Suite Reduction"** by Scott McMaster and Atif M. Memon. *IEEE Trans. Softw. Eng.*, 2008.
    - **"Fault Detection Probability Analysis for Coverage-Based Test Suite Reduction"** by Scott McMaster and Atif M. Memon. In *ICSM '07: Proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM'07)*, (Paris, France), 2007.
    - Call Stack Coverage for GUI Test-Suite Reduction, Scott McMaster* and Atif M. Memon, **Proceedings of the 17th IEEE International Symposium on Software Reliability Engineering (ISSRE 2006)**, Raleigh, NC, USA, Nov. 6-10 2006.
    - Call Stack Coverage for Test Suite Reduction, Scott McMaster* and Atif M. Memon, **Proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM 2005)**, Budapest, Hungary, pp. 473-482, Sep. 25-30, 2005.