Empirical Evaluation of the Fault-detection Effectiveness of Smoke Regression Test Cases for GUI-based Software

Qing Xie
(qing@cs.umd.edu)

Atif Memon, Qing Xie

Department of Computer Science
University of Maryland, College Park

1

What is Smoke Test?

- Smoke test
  - Borrowed from hardware testing
    - A relatively simple check to see whether the product "smokes"
  - Check basic functionality of software
    - Not exhaustive
- Daily/nightly build
  - Software is compiled, linked and (re)tested on a daily basis
  - "Good" build if pass all smoke tests

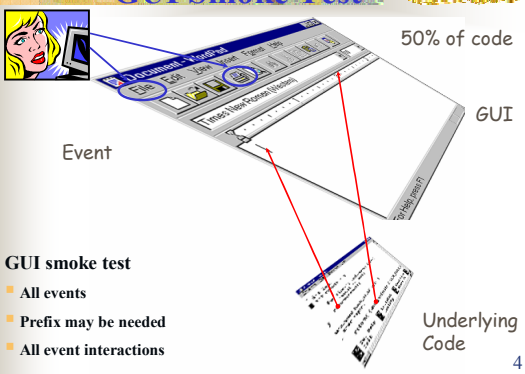
2

Current Practice of Smoke Testing

- Software
  - Microsoft Windows Server 2003
  - GNU projects
  - Mozilla
  - WINE
  - AceDB
- Tools
  - CruiseControl
  - IncrediBuild

3

GUI Smoke Test



GUI smoke test

- **All events**
- **Prefix may be needed**
- **All event interactions**

4

Current Practice of GUI Testing

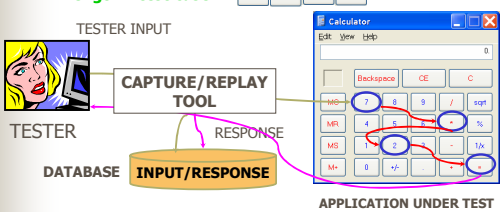
- Tools
  - Capture/replay (record/playback)
    - WinRunner
  - Extensions of JUnit (programming test cases)
    - JFCUnit
    - Pounder
    - Abbot

5

Capture/replay Tool

Capture

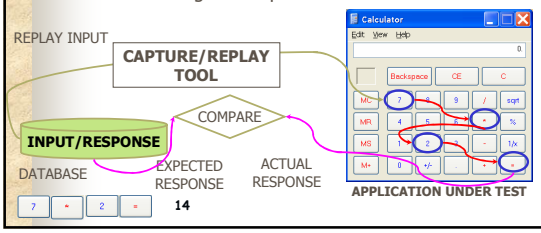
- Tester **MANUALLY** performs events on the software
- Tool records all user inputs and application response
- GUI test case
  - **Sequence of events**
  - **Length 4 test case :** 7 * 2 -



6

Capture/replay Tool

Replay
- Retrieves test case
- Replays it on modified application
- Gets actual response
- Verifies it against expected



7

What is Needed?

- Goal: Automatically test GUI every night
- Model of GUI
  - Event Flow Graph
  - State of GUI
  - Obtain automatically using reverse engineering technique
- Use the model
  - Smoke test case generator
  - Expected state generator
- Test executor

* Fully automated

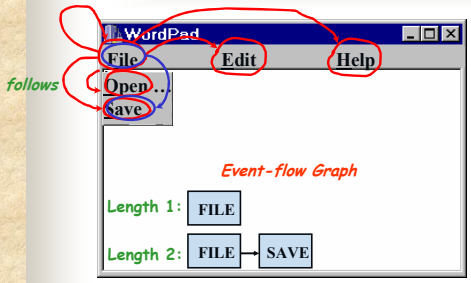
8

Smoke Tests for GUIs

- GUI test case
  - Sequence of events
- Test case length
  - Length 1 – all events in the GUI
  - Length 2 – all possible test cases of the form $<e_i, e_j>$, where event e_j can be performed immediately after event e_i
- Purpose is to test basic functionality quickly
  - Execute all GUI events and event interactions
- Smoke test suite
  - All length 1 and 2 test cases
  - Necessary prefix

9

GUI Model – Event Flow Graph

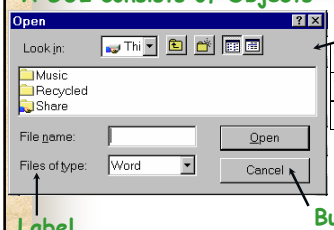


Definition: Event e_y follows e_x iff e_y can be performed immediately after e_x.

10

State of GUI

A GUI consists of Objects



Form

Window State	wsNormal
Width	1088
AutoScroll	TRUE

Label

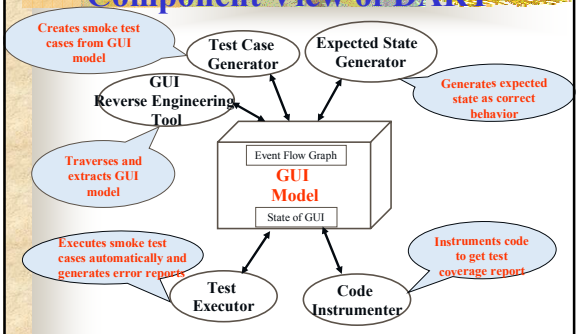
Align	alNone
Caption	Files of type:
Color	clBtnFace
Font	(tFont)

Button

Caption	Cancel
Enabled	TRUE
Visible	TRUE
Height	65

11

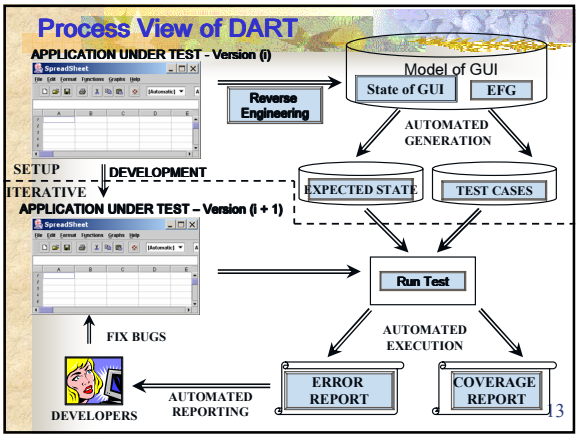
Component View of DART



DART: Daily Automated Regression Tester

12

Process View of DART



Experiments

- Fault detection ability
  - Total faults detected
  - Relationship between fault detection ability and smoke test case length
- Code coverage
  - Relationship between code coverage and smoke test cases
  - Unexecuted code
- Cost

Experimental Process

- Choose subject applications
- Generate smoke test cases with expected state
- Seed faults
- Execute test cases
  - Compare actual GUI state to the expected state
- Measure
  - Number of Faults detected
  - Code Coverage
  - Time
  - Space

Subject Applications

Subject Application	Windows	Widgets	LOC	Classes	Methods	Branches
TerpWord	11	132	4893	104	236	452
TerpSpreadSheet	9	165	12791	125	579	1521
TerpPaint	10	220	18376	219	644	1277
TerpCalc	1	92	9916	141	446	1306
TOTAL	31	609	45976	589	1905	4556

Test Cases Generation

	Potential Test Cases			Actual Generated Test Cases			
	Length			Length			
Subject Application	1	2	3	1	2	3	Total
TerpWord	126	1140	12461	126	1140	3880	5146
TerpSpreadSheet	162	2742	56076	162	2742	2318	5222
TerpPaint	215	8077	502133	215	8077	0	8292
TerpCalc	87	7366	623702	87	7366	0	7453
TOTAL	590	19325	1194372	590	19325	6198	26113

Fault Seeding

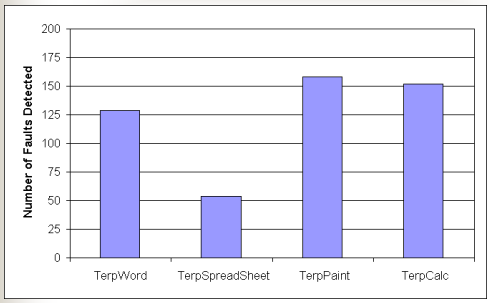
- Seed 200 faults in each application
  - Create 200 versions for each application
  - Exactly one fault in each version
- History-based
  - From a bug tracking tool *bugzilla*

Test Case Execution

- Execute all test cases (5000+) on all 200 versions of the 4 subject applications
  - 5000 * 200 * 4 = 4,000,000+
  - Pentium 4 2.2GHz 256MB RAM
- Compare actual and expected state
- Report mismatches and crashes

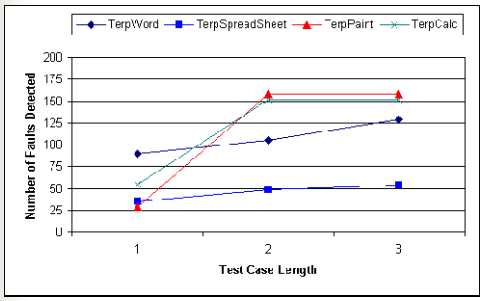
19

Total Faults Detected



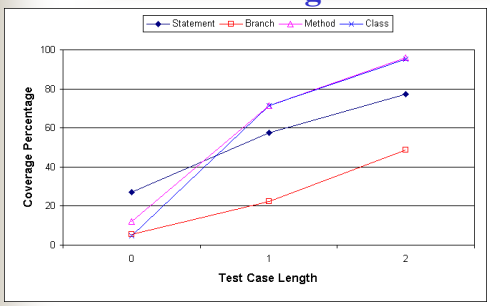
20

Faults Detected vs. Test Case Length



21

Smoke Tests and Code Coverage



TerpCalc

22

Unexecuted Code

- Some mouse/keyboard events not generated (40%)
  - event handlers (*e.g.*, right-click) not executed
- Exceptions not raised (30%)
  - accounted for a large percentage of missed code
- Unable to execute code related to some widgets (10%)
  - *e.g.*, the close button in all windows
- Controlled environment (10%)
  - reset environment variables before each run
  - code related not executed (*e.g.*, list of recently accessed files)
- Some require longer than 2 events (10%)

23

Conclusions

- Short GUI smoke tests are effective
- There are classes of faults that cannot be detected
- Short smoke tests execute a large percentage of code
- Smoke testing process is feasible in terms of time and storage space
- Future Work
  - Increase code coverage
  - Increase completeness of expected state generator
  - Combine GUI-based smoke test and code-based smoke test

24

THANK YOU

http://guitar.cs.umd.edu

25