Using Execution Feedback in Test Case Generation

CMSC 737 – Presentation
Bao Nguyen
baonn@cs.umd.edu

Two Strategies

- **Static plan**
  - Category Partition
  - Data flow analysis (path, branch, def-use, etc)
  - Predicate based: BOR – BRO
  - => Try to guess ahead!!!
- **Dynamic plan**
  - Execution information as feedback
  - Generating test cases on the fly

What's next…

- Test case generation based on execution feedback
- Case study: two recent papers in ICSE'07
- What I'm looking at
- Conclusion

What's next…

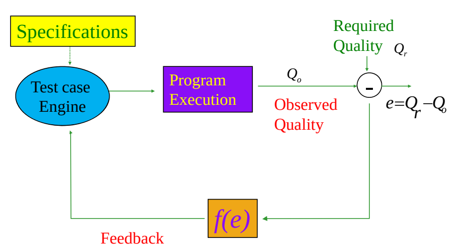
- Test case generation based on execution feedback
- Case study: two recent papers in ICSE'07
- What I'm looking at
- Conclusion

Why Static Plans Are Not Enough?

- Software is so complex:
  - Empirical products => Difficult to formalize
  - Subjective solutions => Difficult to use
  - Human actuators => Difficult to manage
  - Intelligent products => Difficult to predict
  - Dynamic environments => Difficult to address

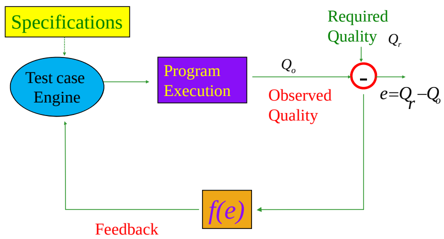
Hard to predict ahead !!!
Hard to exhaustedly test !!!

General Framework



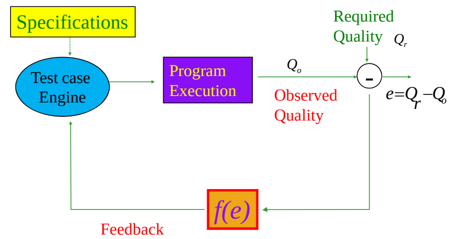
(Adapted from *"Software Cybernetics: Progress and Challenges" - Aditya P. Mathur*)

General Framework



(Adapted from *"Software Cybernetics: Progress and Challenges" - Aditya P. Mathur*)

General Framework



(Adapted from *"Software Cybernetics: Progress and Challenges" - Aditya P. Mathur*)

What's next…

- Execution feedback based test case generation
- Case study: two recent papers in ICSE'07
  - **OOP Testing:** Pacheco, et al. *"Feedback-Directed Random Test Generation"*
  - **GUI Testing:** Yuan and Memon. *"Using GUI Run-Time State as Feedback to Generate Test Cases"*
- What I'm looking at
- Conclusion

What's next…

- Execution feedback based test case generation
- Case study: two recent papers in ICSE'07
  - **OOP Testing:** Pacheco, et al. *"Feedback-Directed Random Test Generation"*
  - **GUI Testing: Yuan and Memon.** ***"Using GUI Run-Time State as Feedback to Generate Test Cases"***
- What I'm looking at
- Conclusion

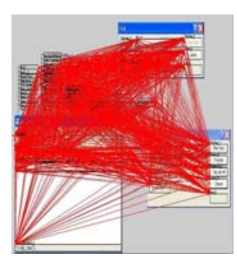
Using GUI Run-Time State as Feedback to Generate Test Cases

Xun Yuan and Atif Memon
ICSE'07

Motivations of the paper

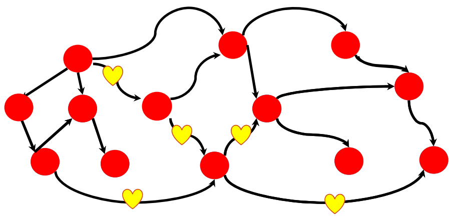
- Previous work
  - 1-way: Crash Test
  - 2-way: Smoke Test
- Longer test cases detected additional faults
- Unable to run multi-way test coverage
  - 2-way run for months [TSE'05]

➡ Try to prune edge



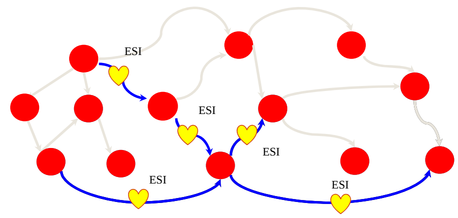
Key idea

- Use GUI states as feedback to identify "important" edges
  - Called Event Semantic Interaction Edges
- Generate new longer test cases covering those edges



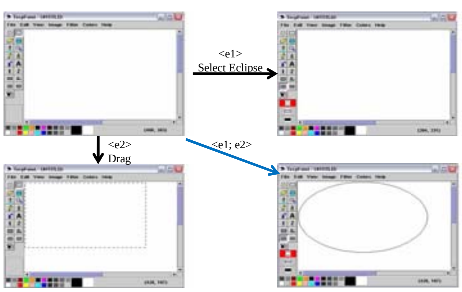
Key idea

- Use GUI states as feedback to identify "important" edges
  - Called Event Semantic Interaction Edges
- Generate new longer test cases covering those edges



Event Semantic Interaction

- Heuristic: Two events executed together results differently than executed in isolation => *semantic interaction*



Six predicates for modeless windows

Predicate 1:

$\exists w \in W; p \in P; v \in V; v' \in V;$

s.t: $((v \neq v') \wedge ((w, p, v) \in \{S_0 \cap e_1(S_0) \cap e_2(S_0)\}) \wedge ((w, p, v') \in e_2(e_1(S_0))));$

Predicate 2:

$\exists w \in W; p \in P; v \in V; v' \in V; v'' \in V;$

s.t: $((v \neq v') \wedge (v' \neq v'') \wedge ((w, p, v) \in \{S_0 \cap e_2(S_0)\}) \wedge ((w, p, v') \in e_1(S_0)) \wedge ((w, p, v'') \in e_2(e_1(S_0))));$

Predicate 3:

$\exists w \in W; p \in P; v \in V; v' \in V; v'' \in V;$

s.t: $((v \neq v') \wedge (v' \neq v'') \wedge ((w, p, v) \in \{S_0 \cap e_1(S_0)\}) \wedge ((w, p, v') \in e_2(S_0)) \wedge ((w, p, v'') \in e_2(e_1(S_0))));$

Predicate 4:

$\exists w \in W; p \in P; v \in V; v' \in V; v'' \in V; \underline{v} \in V;$

s.t: $((v \neq v') \wedge (v \neq v'') \wedge (v'' \neq \underline{v}) \wedge ((w, p, v) \in S_0) \wedge ((w, p, v') \in e_1(S_0)\}) \wedge ((w, p, v'') \in e_2(S_0)\}) \wedge ((w, p, \underline{v}) \in e_2(e_1(S_0))));$

Predicate 5:

$\exists w \in W; p \in P; v \in V; v' \in V; v'' \in V;$

s.t: $((v' \neq v'') \wedge (((w, *, *) \notin S_0) \wedge (((w, p, v) \in e_1(S_0)) \vee ((w, p, v') \in e_2(S_0))) \wedge ((w, p, v'') \in e_2(e_1(S_0))));$

Predicate 6:

$\exists w \in W; \mathit{ENABLED} \in P; \mathit{TRUE} \in V; \mathit{FALSE} \in V;$

s.t: $(((w, \mathit{ENABLED}, \mathit{FALSE}) \in S_0) \wedge ((w, \mathit{ENABLED}, \mathit{TRUE}) \in e_1(S_0)) \wedge \mathbf{EXEC}(e_2, w));$

(More details refer to **"A comprehensive framework for testing graphical user interfaces"** Atif M. Memon.Ph.D. dissertation, 2001)

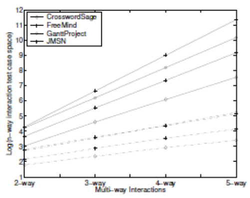
Three contexts for events wrt windows

- **Context 1 :** events in modeless window
- **Context 2 :** events in same modal window
  - $e_x(S)$: the GUI state after executing $<e_x; \mathbf{\mathit{TERM}}>$, x= 1, 2
  - $e_2(e_1(S))$: the GUI state after executing sequence $<e_1; e_2; \mathbf{\mathit{TERM}}>$
- **Context 3 :** events in parent and child modal window
  - $e_1(S)$: the GUI state after executing $<e_1; \mathbf{\mathit{TERM}}>$
  - $e_2(e_1(S))$: the GUI state after executing sequence $<e_1; \mathbf{\mathit{TERM}}; e_2>$

Experiments

- Subject applications: three OSS
  - CrosswordSage 0.3.5
  - FreeMind 0.8.0
  - GanttProject 2.0.1
  - JMSN 0.9.9b2
- Test oracle
  - Program crashes

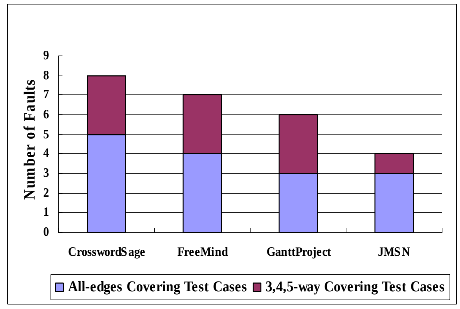
Result - Test case reduction



2-way	3-way	4-way	5-way
	99.78%	99.97%	99.99%

Table 1: Percentages of test case reduction

Result - Faults detection



Lessons learned

- Event handlers are implemented in multiple classes
- A large input space is needed
- Crash because the permutations of events

=> Need longer test cases???

Conclusion of this paper

- Contributions
  - A new GUI model for test-case generation
  - A new relationships among GUI events (i.e. context)
  - A utilization of GUI state as feedback
  - A fully automatic end-to-end GUI testing process
  - A demonstration
- Future work
  - Simplify 6 predicates and 3 contexts
  - Identify and classify events dominating ESI
  - Minimize number of test cases
  - Apply feedback technique to objects outside GUI

What's next…

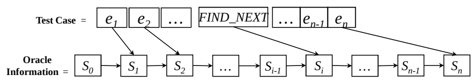
- Test case generation based on execution feedback
- Case study: two recent papers in ICSE'07
- What I'm looking at
- Conclusion

What I'm looking at

- Push test case generation and test case execution closer
  - Generate new test cases *during* the execution
  - Utilize the feedback immediately

A case study

- *Adaptive test oracles:* the QoS idea



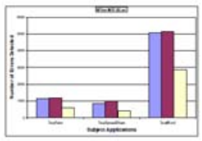


Figure 10. Error detection of $O_{...}$

"Using Transient/Persistent Errors to develop Automated Test Oracles for Event-Driven Software"
Atif M. Memon and Qing Xie. – ASE'04

Conclusion

- Software is dynamic so we need a dynamic approach
- Using feedback in software testing is feasible
- Somewhat related to control theories
  (i.e. software cybernetic)
- Drawback: Like hill climbing
  => local optimization
  - Can mutants (like in GA) overcome this?
  - Systematically vs. Randomly

Questions

- What does "**Event Semantic Interaction**" in section 4 mean?
- What are the threats to validity and what are the weaknesses in Xun's experiments?