Automatically Repairing Broken Workflows for Evolving GUI Applications

Sai Zhang
University of Washington

Joint work with: Hao Lü, Michael D. Ernst
End-user’s workflow

A workflow = A sequence of UI actions for a specific task

Example:
A 3-action workflow of creating a crossword puzzle:
1. Click menu item
2. Fill in textbox
3. Click OK
GUI evolution can break workflows

Version 0.3

(First action in creating a puzzle)

Version 0.35

The workflow is broken!
Goal: repair a broken workflow

- Suggest a “replacement action” for a broken action
  - No change to the code
  - Help users perform the same task, but adapt to the new GUI

(Suggested by our technique: FlowFixer, since both invoke method “showCrosswordBuilder”)

Replacement action:

Click “New Crossword”
GUIs keep evolving all the time
GUIs keep evolving all the time
GUIs keep evolving all the time
GUIs keep evolving all the time
GUIs keep evolving all the time
GUIs keep evolving all the time

GUI evolution can break workflows!
Broken workflows in practice

- **Affect user experience** (focus of this talk)
  - 100+ posts

Example: the ribbon UI in Office 2007

- **Impact automated testing**
  - mimic workflows
  - **30 – 70%** of them are broken in GUI evolution
    [Memon’03, Grechanik’09, Daniel’11]

*Tedious and challenging to resolve them manually*
The “action semantics” challenge

- A UI action’s effect cannot be observed statically.

- Repairing broken workflows needs to:
  - distinguish actions that *look similar* but have *different results*.
  - identify *different* UI actions that may perform the *same* task.

*Requires knowing the “what the action does”*.
Outline

• Problem
• Technique
• Evaluation
• Related Work
• Contributions
Key insights of FlowFixer

• The underlying code implementing the same functionality stays relatively the same between versions

• “action semantics” ≈ the invoked methods

• UI Actions invoking similar methods are likely to perform similar tasks
An overview of the FlowFixer technique

Old version

1. Click “New Crossword”
2. ... 
3. Method matching

Replacement actions:
1. Click “New Crossword”
2. ...

New version

1. Click “New Crossword”
2. ... 
3. CrosswordSolverPanel<init>()

Random testing

1. Click “New Crossword” 1/3
2. Click “Save Crossword” 1/3
3. Click “Solve New Crossword” 1/3
The FlowFixer technique

A broken workflow

Old version

instrument

New version

Record all methods invoked by the broken action

User demonstrates the workflow up to the broken action

User demonstrates the workflow up to the broken action

(the first action is broken)
The FlowFixer technique

A broken workflow

Old version

Instrument

Instrumented version

New version

an execution trace

Static Method Matching

Match each method invoked by the broken action in the new version
The FlowFixer technique

A broken workflow

---

Instrumented version

Old version

New version

Instrumented version

Static Method Matching

Matched Methods

(in the new version)

Random testing

Action $\rightarrow$ method mapping

$\rightarrow f1(), f2(), f3()$

$\rightarrow f1(), f4()$

Randomly execute each applicable UI action, and recursively explore UI actions on new screens
The FlowFixer technique

For each invoked method, find all actions invoking it.

The weight of each action is \textit{inversely proportional} to the number of all possible invoking actions.
Outline

• Problem
• Technique
• Evaluation
• Related Work
• Contributions
Research questions

• How effective is FlowFixer in repairing broken workflows?
  – Accuracy
  – Efficiency

• Comparison with a GUI-comparison-based technique
  [Grechanik’09]
Subject programs and broken workflows

<table>
<thead>
<tr>
<th>Subject</th>
<th>Versions</th>
<th>LOC</th>
<th>ΔLOC</th>
<th>#Broken workflows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crossword</td>
<td>0.3 → 0.35</td>
<td>3,087</td>
<td>1,386</td>
<td>1</td>
</tr>
<tr>
<td>JEdit</td>
<td>2.5 → 2.6</td>
<td>32,607</td>
<td>5,017</td>
<td>1</td>
</tr>
<tr>
<td>Gantt Project</td>
<td>2.0.1 → 2.5.4</td>
<td>55,009</td>
<td>3,777</td>
<td>5</td>
</tr>
<tr>
<td>JabRef</td>
<td>2.0 → 2.8.1</td>
<td>83,447</td>
<td>38,992</td>
<td>3</td>
</tr>
<tr>
<td>Freemind</td>
<td>0.71 → 0.8</td>
<td>70,430</td>
<td>10,757</td>
<td>6</td>
</tr>
</tbody>
</table>

Popular software, being actively developed for 3—12 years

- Selection of broken workflows
  - 356 documented workflows, 70 are broken, 16 have distinct root causes
  - Exclude trivial UI changes, e.g.,
    - swapping two neighboring menu items
    - move a button to a different location on the same panel.
FlowFixer’s accuracy

- Measured by the **absolute rank** of the **correct** actions

- FlowFixer can repair **15** broken workflows
FlowFixer’s efficiency

• Random testing
  – 27 mins per application
    (A one-time cost, shared by different workflows)

• User demonstration
  – < 1 min per workflow
    (assuming the old version is installed)

• Action recommendation
  – 4 mins per workflow
An example repair

Save current state

Fill the textbox to save the current state

Gantt Project version 2.0

Gantt Project version 2.5
An example repair

Save current state

UndoableEditImpl.createTemporaryFile
Comparison with an existing technique

• **REST**: a GUI-comparison-based technique [Grechanik’09]
  – A black-box approach
  – Compare GUIs of two versions to identify modified UI elements
  – Identifies **affected** actions, but gives **no** repair suggestion
Comparison with an existing technique

- **REST**: a GUI-comparison-based technique [Grechanik’09]
  - A black-box approach
  - Compare GUIs of two versions to identify modified UI elements
  - Identifies *affected* actions, but gives *no* repair suggestion

- **Extend REST** for workflow repair
  - Recommend actions on the *matched* UI element of the *new* version
REST vs. FlowFixer

Fail to fix 10 workflows

Fail to fix 1 workflow

6 workflows fixed

15 workflows fixed

REST

FlowFixer
Why REST did not work well?

- **REST** only repairs 6 workflows where a UI element is moved to a different location
  - Ineffective for non-trivial UI changes
    - UI label change
    - UI element change
    - UI action change
  
- **FlowFixer** repairs 15 broken workflows
  - Execute UI actions and observe their consequences

REST’s **black-box approach is not aware of the “action semantics”**
Experimental conclusions

• FlowFixer is accurate and efficient in repairing broken workflows

• FlowFixer achieves better results than a GUI-comparison-based technique
Outline

• Problem
• Technique
• Evaluation
• Related Work
• Contributions
Related work

• **Test repair**
  
  ReAssert [Daniel’09], REST [Grechanik’09], Guitar [Memon’04],
  Genetic approach [Huang’10], WATER [Choudhary’11] …

  *Make obsoleted tests compilable without preserving its original semantics.*

  **Not** applicable to repairing broken workflows.

• **Program repair**
  
  GenProg [Weimer’09], ClearView [Perkins’09], PAR [Kim’13]…

  *Search patches for bugs.*

  **Not** applicable to broken workflows caused by UI changes.

• **Change analysis**
  
  Chianti [Ren’05], SemDiff [Dagenais’08], RefactoringCrawler [Dig’05],
  Hybrid approach [Wang’12] …

  *Identify code-level changes and compute the effects.*

  **Not** applicable for repairing UI-level workflows.
Outline

• Problem
• Technique
• Evaluation
• Related Work
• Contributions
Future directions

• User study

• Extend FlowFixer to repair UI test scripts
  – Lift *syntax-correcting* repair to *semantics-preserving* repair

• Integrate FlowFixer into software evolution
  – Proactively finding broken workflows
  – Summarize UI-level changes
  – Automatically update user manual
  – Help users learn new GUI features
Contributions

• A technique to repair broken workflows
  
  analyze method invocations and evolution to reason about fix actions
  
  – fully automated
  
  – handles non-trivial code changes

• Experiments that demonstrate its usefulness
  
  – Accurate and efficient
    
    • Fixed 15 out of 16 broken workflows
  
  – Outperforms alternative techniques

• The FlowFixer tool implementation:

  http://workflow-repairer.googlecode.com
[Backup Slides]
What if multiple actions are broken?

- Use FlowFixer in an interactive way

FlowFixer

Fix action
1.
2.
3. ...

Might be a different broken action!

FlowFixer

Fix action
1.
2.
3. ...

Fixed!
FlowFixer’s recommendation limitation

• Recommends one replacement action for a broken action

• Does not support recommending:
  – A **sequence** of actions for **one** action
  – **One** action for a **sequence** of actions
  – A **sequence** of actions for a **sequence** of actions
Why does this simple random testing work?

- Goal:
  - Identify “signature” method for each UI action
  - NOT achieve good coverage

- The “signature” method is often easy to reach:

  - Event handler, shared by many actions
  - A “signature” method, only invoked by “Clicking New Crossword”
  - Other methods. Requires certain states

- Symbolic, model-based techniques might achieve better results, but are more expensive to use