Guided GUI Testing of Android Apps with Minimal Restart and Approximate Learning

Wontae Choi, George Necula, and Koushik Sen

University of California, Berkeley

OOPSLA 2013 @ Indianapolis, USA
Modern platforms use rich GUI

Android  iOS  HTML5  Windows 8

This talk: automated GUI testing
Working Example
Working Example

- Android app
- 3 consecutive EULA screens
- Touching **No** will terminate the app
- **Scroll** does nothing
Working Example

- Android app
- 3 consecutive EULA screens
- Touching **No** will terminate the app
- **Scroll** does nothing
Random Testing
(Existing Idea)

License Term #1
EULA 1 ($q_1$)
Yes No

License Term #2
EULA 2 ($q_2$)
Yes No

License Term #3
EULA 3 ($q_3$)
Yes No

Yes

Yes

Yes

Sanity

Action 1
Action 2
Action 3

Main ($q_M$)
Random Testing
(Existing Idea)

- Randomly tries the enabled events
- Good to find shallow bugs

- May not reach deep program states
  - Difficult to reach the main screen
    - Takes 24 events + 7 restarts (on average)
    - Optimum: 3 events
Model Based Testing

(Existing Idea)

A behavioral model is provided
Computes test cases from the model
- [Scroll-Down, Scroll-Up, Yes, Yes, Scroll-Down, Scroll-Up, Yes] covers all states and non-terminating transitions

Manually providing a model is tedious and error-prone
Testing with Model Learning

(Existing Idea)
Testing with Model Learning

(Existing Idea)

Exercise

T with ML Engine

App

License Term #1
Yes | No

License Term #2
Yes | No

EULA 1 (e1)
EULA 2 (e2)
Testing with Model Learning

(Existing Idea)

T with ML Engine

Exercise

Program State

App

License Term #1
Yes | No
eula 1 (x1)

License Term #2
Yes | No
eula 2 (x2)
Testing with Model Learning

(Existing Idea)

Model

T with ML Engine

Exercise

Program State

App

Learn

Yes

Scroll Up

Yes

No

Scroll Down

Yes

No

Scroll Up

Yes

No

Scroll Down

q1

q2

q3

End
Testing with Model Learning

(Existing Idea)

Model

Guide

T with ML Engine

Exercise

Program State

App

<table>
<thead>
<tr>
<th>Model State</th>
<th>Guide</th>
<th>Exercise</th>
</tr>
</thead>
<tbody>
<tr>
<td>q1</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>q2</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>q3</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

Program State

Scroll Up

Scroll Down

Learn

<table>
<thead>
<tr>
<th>EULA 1 (e1)</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>License Term #1</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EULA 2 (e2)</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>License Term #2</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
Testing with Model Learning

(Existing Idea)

Learn a model during testing
Guide testing using a model
Testing with Model Learning

(Existing Idea)

Learn a model during testing
Guide testing using a model

$L^*$ experience

- $L^*$ is a widely-used learning algorithm
- $L^*$ based testing is less effective than random testing
  - Conservative learning: large model
  - Frequently restarts a target application
Testing with Model Learning

(Existing Idea)

Learn a model during testing
Guide testing using a model

$L^*$ experience

- $L^*$ is a widely-used learning algorithm
- $L^*$ based testing is less effective than random testing
  - Conservative learning: large model
  - Frequently restarts a target application

with similar states!
Testing with Model Learning

(Existing Idea)

Learn a model during testing
Guide testing using a model

$L^*$ experience
- $L^*$ is a widely-used learning algorithm
- Less effective than random testing
- Conservative learning: large model
- Frequently restarts a target application

execution cost
1 restart = 5 GUI events

with similar states!
SwiftHand Algorithm

Key Insight
SwiftHand Algorithm

Key Insight

- Explore diverse program states quickly
  - Optimistically keep the model small
  - Refine the model when encountering inconsistency
SwiftHand Algorithm

Key Insight

• Explore diverse program states quickly
  - Optimistically keep the model small
  - Refine the model when encountering inconsistency

• Be aware of the cost model for different events
  - Try to reach a goal state without restart
SwiftHand with Example

Initialization

Actual Application

Model Learned

License Term #1
EULA 1 ($q_1$)

License Term #2
EULA 2 ($q_2$)

License Term #3
EULA 3 ($q_3$)

Sanity
Action 1
Action 2
Action 3

Main ($q_M$)
SwiftHand constructs a model via observing the application screen.
SwiftHand constructs a model via observing the application screen.
SwiftHand constructs a model via observing the application screen.
SwiftHand with Example

**Initialization**

**Actual Application**

SwiftHand constructs a model via observing the application screen.

**Model Learned**

SwiftHand constructs a model via observing the application screen.
SwiftHand with Example

SwiftHand constructs a model via observing the application screen.

**Initialization**

**Actual Application**

- License Term #1
  - EULA 1 (q₁)
  - [ ] Yes
  - [X] No

- License Term #2
  - EULA 2 (q₂)
  - [ ] Yes
  - [X] No

- License Term #3
  - EULA 3 (q₃)
  - [ ] Yes
  - [X] No

- Sanity
  - Action 1
  - Action 2
  - Action 3

**Model Learned**

- SD
- SU
- Touch (7)

- Yes
- No

- q₁

get GUI tree
SwiftHand constructs a model via observing the application screen.
SwiftHand with Example

Iteration 1

Actual Application

Model Learned

pick a target
SwiftHand with Example

**Iteration 1**

**Actual Application**

- License Term #1 (Yes | No)
- License Term #2 (Yes | No)
- License Term #3 (Yes | No)

**Model Learned**

<table>
<thead>
<tr>
<th>Sanity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action 1</td>
</tr>
<tr>
<td>Action 2</td>
</tr>
<tr>
<td>Action 3</td>
</tr>
</tbody>
</table>

SwiftHand decides the next state to visit based on the learned partial model.
SwiftHand with Example

**Actual Application**

- License Term #1
  - EULA 1 (q₁)
  - Yes
- License Term #2
  - EULA 2 (q₂)
  - Yes
- License Term #3
  - EULA 3 (q₃)
  - Yes

**Model Learned**

- State: q₁
- Actions: SD, SU
- Transitions:
  - Yes: SD
  - No: SU

**Iteration 1**

SwiftHand decides the next state to visit based on the learned partial model.
SwiftHand with Example

Actual Application

Model Learned

SwiftHand decides the next state to visit based on the learned partial model.
SwiftHand decides the next state to visit based on the learned partial model.
**SwiftHand with Example**

*Iteration 1*

**Actual Application**

- License Term #1
  - EULA 1 (\(q_1\))
  - Yes | No
- License Term #2
  - EULA 2 (\(q_2\))
  - Yes | No
- License Term #3
  - EULA 3 (\(q_3\))
  - Yes | No

**Model Learned**

![Diagram](https://via.placeholder.com/150)

- SD
- SU
- Yes
- No

---

W.Choi@OOPSLA'13 10
SwiftHand with Example

Iteration 1

Actual Application

Model Learned

EULA 1 \(q_1\)
- Yes
- No

EULA 2 \(q_2\)
- Yes
- No

EULA 3 \(q_3\)
- Yes
- No

License Term #1
- Yes
- No

License Term #2
- Yes
- No

License Term #3
- Yes
- No

Sanity
- Action 1
- Action 2
- Action 3

Main \(q_m\)

$q_1$
- Yes
- No

SD ↓

SU

$q_2$
- Yes
- No

abstract

$q_1$
SwiftHand with Example

Iteration 1

Actual Application

Model Learned

License Term #1
Yes | No
EULA 1 (q_1)

License Term #2
Yes | No
EULA 2 (q_2)

License Term #3
Yes | No
EULA 3 (q_3)

Sanity

Action 1
Action 2
Action 3

Main (q_M)

abstract

new

q_2
Yes

No

SD

SU

Yes

No

q_1

?
SwiftHand with Example

Iteration 1

Actual Application

Model Learned

Yes

No

abstract

new

add new state
**SwiftHand** with Example

**Iteration 1**

**Actual Application**

- License Term #1: EULA 1 (q₁)
  - Yes | No
- License Term #2: EULA 2 (q₂)
  - Yes | No
- License Term #3: EULA 3 (q₃)
  - Yes | No

**Model Learned**

- Sanity
  - Action 1
  - Action 2
  - Action 3

- Main (q₃)

- q₂
  - Yes
  - No

- Abstract

- New

- Add the new state
SwiftHand with Example

**Iteration 1**

**Actual Application**

<table>
<thead>
<tr>
<th>License Term #1</th>
<th>License Term #2</th>
<th>License Term #3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Model Learned**

```
q1
  Yes -> q2
  No  -> SD
  ?    -> SU

q2
  Yes -> ?
  No  -> ?
```

- `q1`: Start state
- `q2`: New state
- `SD`: Sanitize Data
- `SU`: Suppress User
- `Yes`, `No`: Transition conditions
- `?`: Unknown state

- abstract
- add the new state
SwiftHand with Example

Iteration 1

Actual Application

Model Learned

The new screen has different enabled inputs. Add a new state to the model
SwiftHand with Example

Iteration 2

Actual Application

Model Learned

pick a target
SwiftHand with Example

Actual Application

Model Learned

Choose a target state reachable from the current model state (avoid restart)
SwiftHand with Example

Actual Application

Model Learned

Choose a target state reachable from the current model state (avoid restart)
Actual Application

Model Learned

Choose a target state reachable from the current model state (avoid restart)
SwiftHand with Example

**Iteration 2**

**Actual Application**

- License Term #1: Yes
- License Term #2: Yes
- License Term #3: Yes

**Model Learned**

- **Sanity**
  - Action 1
  - Action 2
  - Action 3

- **Main (q_3)**

- **q1**
  - SD
  - SU
  - Yes
  - No

- **q2**
  - Yes
  - No

*pick a target*
SwiftHand with Example

Iteration 2

Actual Application

Model Learned

execute
SwiftHand with Example

**Iteration 2**

**Actual Application**

- License Term #1
  - Yes
  - No
- License Term #2
  - Yes
  - No
- License Term #3
  - Yes
  - No

**Model Learned**

- EULA 1 (q1)
- EULA 2 (q2)
- EULA 3 (q3)

<table>
<thead>
<tr>
<th>Sanity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action 1</td>
</tr>
<tr>
<td>Action 2</td>
</tr>
<tr>
<td>Action 3</td>
</tr>
</tbody>
</table>

- SD
- SU

- q1
  - Yes
  - No

- q2
  - Yes
  - No

W.Chi@OOPSLA'13
SwiftHand with Example

Iteration 2

Actual Application

- License Term #1
  - Yes: EULA 1 (q₁)
  - No: EULA 2 (q₂)

- License Term #2
  - Yes: EULA 2 (q₂)
  - No: EULA 3 (q₃)

- License Term #3
  - Yes: EULA 3 (q₃)
  - No: nothing

Model Learned

- q₁
  - SD: Yes
  - SU: No

- q₂
  - Yes: EULA 1 (q₁)
  - No: EULA 2 (q₂)

- q₃
  - SD: Yes
  - SU: No

abstract
SwiftHand with Example

**Iteration 2**

**Actual Application**

```
<table>
<thead>
<tr>
<th></th>
<th>License Term #1</th>
<th>License Term #2</th>
<th>License Term #3</th>
</tr>
</thead>
<tbody>
<tr>
<td>EULA 1 ($q_1$)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>EULA 2 ($q_2$)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>EULA 3 ($q_3$)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
```

**Model Learned**

```
\[
\begin{array}{c}
q_1 \\
q_2 \\
q_3
\end{array}
\]
SwiftHand with Example

Iteration 2

Actual Application

Model Learned

Abstract

merge similar states
**SwiftHand with Example**

**Actual Application**

- License Term #1
  - EULA 1 ($q_1$)
    - Yes
    - No

- License Term #2
  - EULA 2 ($q_2$)
    - Yes
    - No

- License Term #3
  - EULA 3 ($q_3$)
    - Yes
    - No

**Model Learned**

-_sd
- su

$q_3$

- Yes
- No

**Iteration 2**

- merge similar states

- abstract

- similar
**SwiftHand with Example**

**Iteration 2**

**Actual Application**

- License Term #1 (Yes | No)
- License Term #2 (Yes | No)
- License Term #3 (Yes | No)

**Model Learned**

- Sanit
  - Action 1
  - Action 2
  - Action 3

- Similar states

- Merge similar states

q3

- Yes
- No

q1,3

- Yes
- No

q2

- Yes
- No

- Merge similar states

- Similar
SwiftHand with Example

Iteration 2

Actual Application

Model Learned

Optimistic merging
Expectation: a new screen will lead to interesting states
**SwiftHand with Example**

**Iteration 3**

**Actual Application**

- License Term #1
  - Yes
  - No
- License Term #2
  - Yes
  - No
- License Term #3
  - Yes
  - No

**Model Learned**

- SD
- SU

**q1,3**

- Yes
- No

**q2**

- Yes
- No

**pick a target**
SwiftHand with Example

Iteration 3

Actual Application

Model Learned

pick a target
SwiftHand with Example

Iteration 3

Actual Application

Model Learned

pick a target

W.Choi@OOPSLA'13
SwiftHand with Example

Iteration 3

Actual Application

Model Learned

execute
SwiftHand with Example

Iteration 3

Actual Application

Model Learned

Abstract
SwiftHand with Example

Iteration 3

Actual Application

Model Learned

Action 1
Action 2
Action 3

QM

Sanity
Action 1
Action 2
Action 3

SD
SU

Yes
No

execute

abstract inconsistent
SwiftHand with Example

Iteration 3

Actual Application

Model Learned

Some of previous merges were to aggressive
SwiftHand with Example

Iteration 3

Actual Application

Model Learned

$q_1$ → $q_2$ → $q_3$ → $q_M$

EULA 1 ($\alpha_1$) → Yes → License Term #1

EULA 2 ($\alpha_2$) → Yes → License Term #2

EULA 3 ($\alpha_3$) → Yes → License Term #3

Sanity Table:

- Action 1
- Action 2
- Action 3

Main ($\alpha_M$)

SD

SU

Yes

No

Yes

No

Yes

No
SwiftHand with Example

Iteration 3

Actual Application

Model Learned

reconstruct based on collected traces
SwiftHand with Example

Iteration 3

Actual Application

- License Term #1
  - Yes
- License Term #2
  - Yes
- License Term #3
  - Yes

Model Learned

- EULA 1 (q₁)
- EULA 2 (q₂)
- EULA 3 (q₃)

- SD
- SU

- q₁ → q₂
- q₂ → q₃
- q₃ → qM

- Yes
- No

- Action 1
- Action 2
- Action 3

- Sanity

No
Yes

Reconstruction uses off-line learning algorithm

(*without re-executing the program)
SwiftHand

Summary

• Combining learning and GUI testing
  - FSM model (state = screen / transition = event)
  - **Approximate learning** with optimistic state merging
    • helps to explore interesting states quickly
  - **Avoids restarts**
    • helps to reduce testing time
  - Perform passive learning
    • helps to keep model consistent with the app
Experiments

Setup
- SwiftHand vs Random vs $L^*$
- 10 Android apps (F-droid app market)
- 3 hours of testing per strategy
- On an Android emulator

Measurement
- Branch coverage (final coverage, coverage speed)
Experiments - Coverage

![Graph showing branch coverage ratio over time for different methods: SwiftHand, random, lstar. The x-axis represents time in minutes, ranging from 0 to 180, and the y-axis represents the weight of branch coverage ratio, ranging from 0 to 0.7. The graph demonstrates the relative performance of each method over time.](image-url)
Experiments - Coverage

- **SwiftHand** has the fastest coverage speed
- **SwiftHand** has the highest final coverage
Experiments - Coverage

- **SwiftHand** has the fastest coverage speed
- **SwiftHand** has the highest final coverage

**SwiftHand** >> **Random** >> **L***
(for all 10 apps)
Experiments - Cost of Test

20 test runs / point
Experiments - Cost of Test

Time comparison:
one $L^*$ run = 7 SwiftHand runs = 4 Random runs
Experiments - Cost of Test

20 test runs / point

Time comparison:
one $L^*$ run = 7 SwiftHand runs = 4 Random runs

$L^* \gg$ Random $\gg$ SwiftHand
(for all 10 apps)
Challenges and Future Work

• External states

App state = Local state + External state

How to clean restart an app?
Challenges and Future Work

- External states
- String inputs

Application behavior depends on strings
How to pick strings well?
Challenges and Future Work

- External states
- String inputs
- Combinatorial explosion

Component local states are multiplied in the model. How to shrink search space?
Challenges and Future Work

- External states
- String inputs
- Combinatorial explosion

Component local states are multiplied in the model. How to shrink search space?

W.Choi@OOPSLA’13
Challenges and Future Work

- External states
- String inputs
- Combinatorial explosion

Component local states are multiplied in the model.
How to shrink search space?
Related Works

Automated model based GUI testing

• GUITAR
  - B.Nguyen, B.Robbins, I.Benerjee, and A.Memo, ASE 2013
  - Atiff Memon, STVR 2007

• CrawlJax

Testing with model learning

• Survey papers
  - K.Meinke, F.Niu, and M.Sindhu, ISoLa 2011
  - D.Lee and M.Yannakakis, IEEE 1996
Conclusion

Guided GUI Testing of Android Apps with Minimal Restart and Approximate Learning
Conclusion

Guided GUI Testing of Android Apps with Minimal Restart and Approximate Learning

It worked.
Conclusion

Guided GUI Testing of Android Apps with Minimal Restart and Approximate Learning

It worked.

Thank you!