Automated model-based android GUI testing using multi-level GUI Comparison Criteria

happyBiker, haxxorrman

Flatulenza, University of Prone

homemadebike.9999999@stud.unipronax.fritt, lexusGTsportage.99999999@stud.unipronax.fritt

January 21, 2017

Overview

GUI testing automatico(1)

Perché dedicare particolare attenzione alle applicazioni Android?

Attualmente le applicazioni android comprendono quasi il 90% del mercato di software mobile.

Cos'è il GUI testing?

È una metodologia di testing che esplora una parte dello spazio degli stati del programma testato dando gli input al programma tramite l'interfaccia grafica.

Perché automatizzare il testing?

Il motivo principale è il costo, l'esecuzione manuale dei test ha il costo eccessivo per molte applicazioni. Inoltre l'esecuzione manuale dei test richiede tanto tempo e a volte non è in grado di individuare tutti gli errori.

GUI testing automatico(2)

Attualmente sul mercato si trovano dei tool per il GUI testing automatico che generano gli input in modo alleatorio, tentando di generare una sequenza di input che provoca il malfunzionamento del software. Nel testo viene citato un tool chiamato *Android Monkey* che permette questo tipo di test.

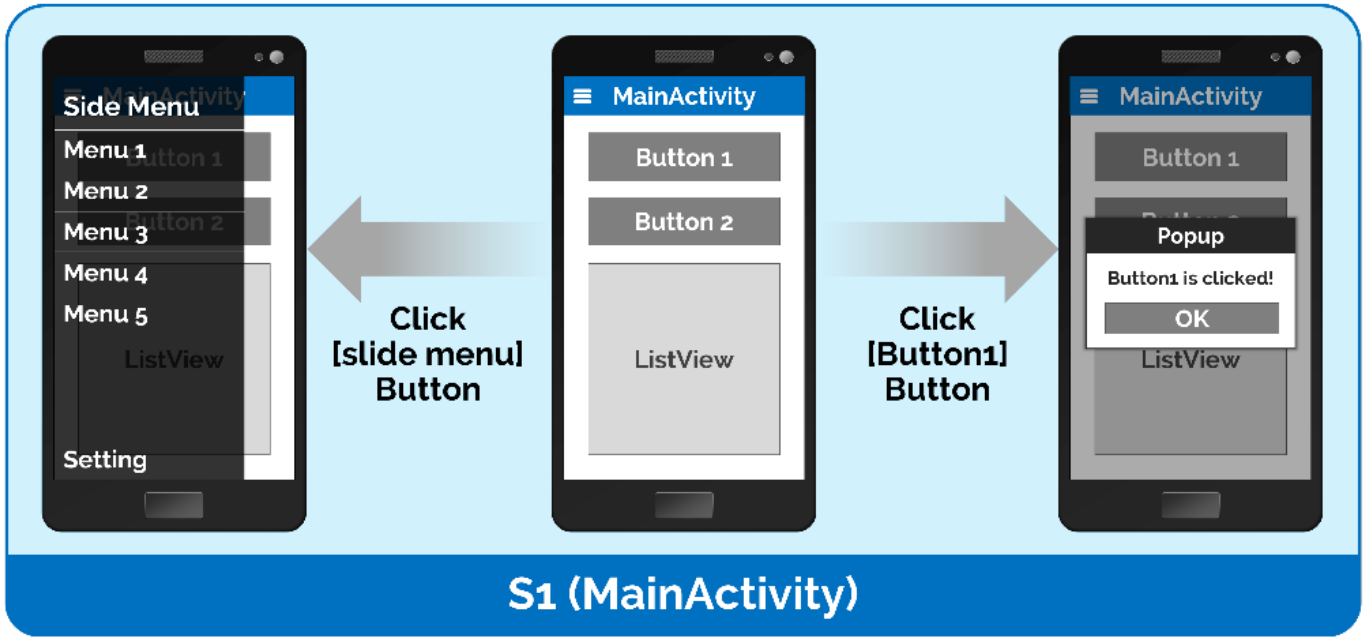
Dato che si tratta essenzialmente di una ricerca locale nello spazio degli stati, questi metodi sono soggetti alle problematiche degli algoritmi di ricerca locale. Per esempio possono valutare lo stesso stato più volte.

L'approccio proposto dagli autori è di costruire un grafo degli stati della GUI, utilizzando dei criteri di equivalenza (GUICC) per distinguere gli stati.

Modello GUI

Un altro approccio prevede l'uso del modello della GUI. Il modello GUI è un automa a stati finiti, dove gli stati dell'automa corrispondono agli stati dell'interfaccia grafica. L'automa transisce da uno stato all'altro al verificarsi di un evento. Il modello GUI viene rappresentrato da un multigrafo dove i nodi corrispondono agli stati e gli archi corrispondono alle transizioni. Ogni arco è etichettato dall'evento che provoca la transizione. Il concetto fondamentale necessario per la costruzione di questo grafo è il criterio di equivalenza degli stati.

Le applicazioni Android spesso generano parte dell'interfaccia grafica in modo dinamico. Per modellare correttamente la GUI biogna tener conto delle componenti dinamiche.



Un criterio d'equivalenza debole come "Nome Activity" in questo caso non sarebbe in grado di accorgersi della differenza tra gli stati.

Multi-Level GUI Comparison Criteria (Multi-Level GUICC)

La struttura a livelli definita dagli autori del paper è composta da:

- **Nome pacchetto**. Quando l'activity corrente non appartiene allo stesso pacchetto dell'applicazione testata significa che l'applicazione testata è stata terminata oppure ha invocato un altra applicazione, in ogni caso lo stato corrente è considerato lo stato terminale.
- **Nome Activity**. Quando l'activity corrente ha un nome diverso da quella precedente si tratta di uno stato diverso.
- **Composizione Widget non-eseguibili**
- **Composizione Widget eseguibili**
- **Contenuti**

Generazione del modello

Dato che le applicazioni Android in genere non contengono un modello della GUI costruito dallo sviluppatore è stato sviluppato un metodo per dedurre il modello (*Model learning*) da un applicazione tramite *reverse engineering*(*GUI Ripping*).

Generazione del modello(2)

Per costruire il grafo della GUI vengono eseguiti iterativamente questi passaggi:

- Generare un evento amissibile nello stato corrente.
- Verificare tramite i criteri d'equivalenza se lo stato della GUI è cambiato. In caso affermativo ci sono tre casi:
  - Il nuovo stato è già presente come nodo nel grafo. Allora si aggiunge un arco dallo stato precedente al nuovo stato etichettato dall'evento che porta in questo stato.
  - Non esiste un nodo corrispondente, allora abbiamo scoperto uno stato nuovo. Viene generato un nuovo nodo e generato l'arco come nel primo caso.
  - Lo stato risultato è uno stato terminale, allora l'applicazione è terminata in seguito all'evento.(Terminazione normale o anormale dovuta ad un errore/eccezione).

  In caso contrario si aggiunge un *loop*(un arco che origina e termina nello stesso nodo) etichettato dall'evento.

Architettura del sistema

Motore di testing

Viene eseguito lato controllore.

- **Esecutore test** Sceglie l'input da mandare al software testato secondo un algoritmo(BFS).
- **Generatore del grafo** descritto sopra.
- **Error checker** Controlla la presenza di errori.

Strato di comunicazione

Serve per trasferire informazioni fra la macchina controllore e il dispositivo Android che esegue il software testato.

Event agent

Software eseguito sul dispositivo Android. Riceve messaggi dal contollore e genera input al software testato corrispondenti.

Paragraphs of Text

Sed iaculis dapibus gravida. Morbi sed tortor erat, nec interdum arcu. Sed id lorem lectus. Quisque viverra augue id sem ornare non aliquam nibh tristique. Aenean in ligula nisl. Nulla sed tellus ipsum. Donec vestibulum ligula non lorem vulputate fermentum accumsan neque mollis.

Sed diam enim, sagittis nec condimentum sit amet, ullamcorper sit amet libero. Aliquam vel dui orci, a porta odio. Nullam id suscipit ipsum. Aenean lobortis commodo sem, ut commodo leo gravida vitae. Pellentesque vehicula ante iaculis arcu pretium rutrum eget sit amet purus. Integer ornare nulla quis neque ultrices lobortis. Vestibulum ultrices tincidunt libero, quis commodo erat ullamcorper id.

Bullet Points

- Lorem ipsum dolor sit amet, consectetur adipiscing elit
- Aliquam blandit faucibus nisi, sit amet dapibus enim tempus eu
- Nulla commodo, erat quis gravida posuere, elit lacus lobortis est, quis porttitor odio mauris at libero
- Nam cursus est eget velit posuere pellentesque
- Vestibulum faucibus velit a augue condimentum quis convallis nulla gravida

Blocks of Highlighted Text

Block 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer lectus nisl, ultricies in feugiat rutrum, porttitor sit amet augue. Aliquam ut tortor mauris. Sed volutpat ante purus, quis accumsan dolor.

Block 2

Pellentesque sed tellus purus. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Vestibulum quis magna at risus dictum tempor eu vitae velit.

Block 3

Suspendisse tincidunt sagittis gravida. Curabitur condimentum, enim sed venenatis rutrum, ipsum neque consectetur orci, sed blandit justo nisi ac lacus.

Multiple Columns

Heading

1. Statement
2. Explanation
3. Example

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer lectus nisl, ultricies in feugiat rutrum, porttitor sit amet augue. Aliquam ut tortor mauris. Sed volutpat ante purus, quis accumsan dolor.

Table

Treatments	Response 1	Response 2
Treatment 1	0.0003262	0.562
Treatment 2	0.0015681	0.910
Treatment 3	0.0009271	0.296

Table: Table caption

Theorem

Theorem (Mass–energy equivalence)

$E = mc^2$

Verbatim

Example (Theorem Slide Code)

```
\begin{frame}
\frametitle{Theorem}
\begin{theorem}[Mass--energy equivalence]
$E = mc^2$
\end{theorem}
\end{frame}
```

Figure

Uncomment the code on this slide to include your own image from the same directory as the template .TeX file.

Citation

An example of the `\cite` command to cite within the presentation:

This statement requires citation [Smith, 2012].

References

John Smith (2012)

Title of the publication

Journal Name 12(3), 45 – 678.

The End