UNIVERSITA' DEGLI STUDI DI NAPOLI FEDERICO II
Facoltà di Ingegneria
Corso di Studi in Ingegneria Informatica

tesi di laurea

Strumento e tecnica a supporto del crash testing automatico di applicazioni mobili basato sul sistema operativo Android

Anno Accademico 2010/2011

relatore

Ch.mo prof. Porfirio Tramontana

correlatore

ing. Domenico Amalfitano

candidato

Sergio Principe

Matr. 534/3103

Problema

- Lo sviluppo di applicazioni per dispositivi mobili sta crescendo in modo esponenziale. Da qui nasce la necessità di effettuare un testing efficiente ed efficace per individuare il maggior numero possibile di malfunzionamenti con il minimo sforzo.

Obiettivo

- Proporre nuove tecniche e nuovi processi in grado di consentire l'automazione del procedimento di testing, evidenziando quelle che sono le problematiche attinenti a questo tipo di attività

Lavoro Svolto

- Realizzazione e sperimentazione di un tool per la generazione automatica di casi di test per applicazioni Android

Testing delle applicazioni attraverso la GUI

L'obiettivo di questa tesi è la realizzazione di uno strumento per l'esplorazione automatica delle interfacce utente di una applicazione Android, testandole nel contempo allo scopo di rilevare malfunzionamenti di tipo crash.

Possibili approcci al test attraverso la GUI

1)Random testing della GUI (Monkey): viene testata l'applicazione generando automaticamente input ed eventi casuali sulle interfacce.

2)Model based: viene costruito un modello delle interfacce dell'applicazione e percorrendo tale modello si ottengono casi di test.

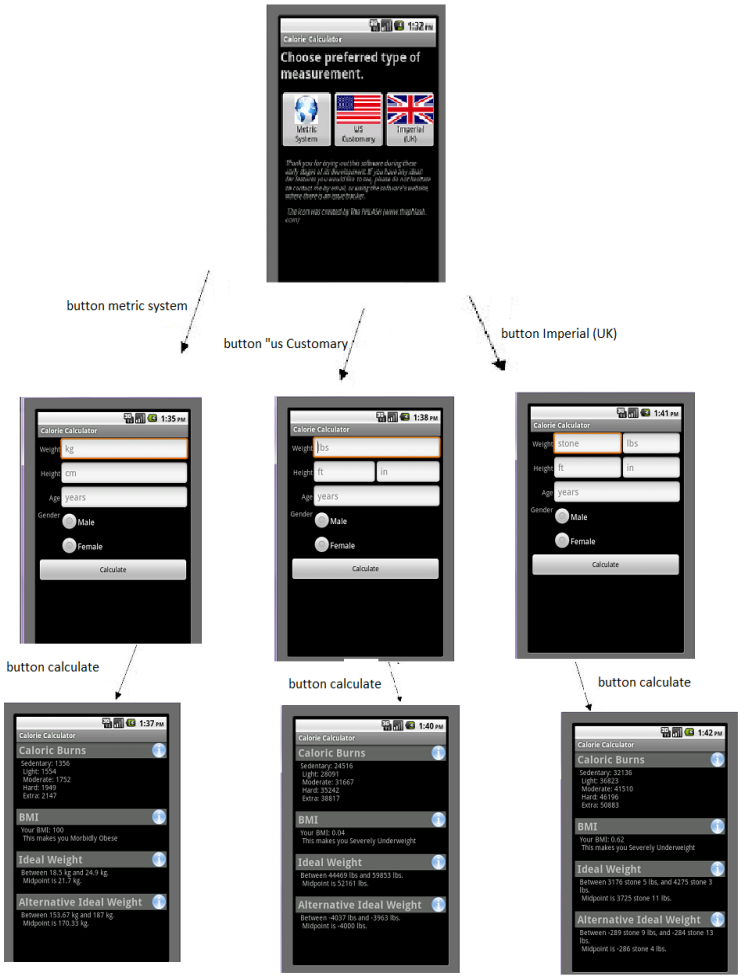
E' stato scelto l'approccio Model based.

Modello GUI Tree

Il modello del GUI-Tree rappresenta un albero della struttura della GUI di una applicazione. Ogni *nodo* del grafo rappresenta una particolare interfaccia della GUI e ogni *lato* rappresenta una transizione scatenata da un evento utente

Le applicazioni Android sono effettivamente considerate Event Driven Software (EDS), cioè il comportamento è guidato da diversi tipi di eventi scatenati dall'utente, che vanno a modificare l'interfaccia.

Due interfacce sono considerate equivalenti se hanno lo stesso nome e lo stesso numero di widget.



GUI-Ripping

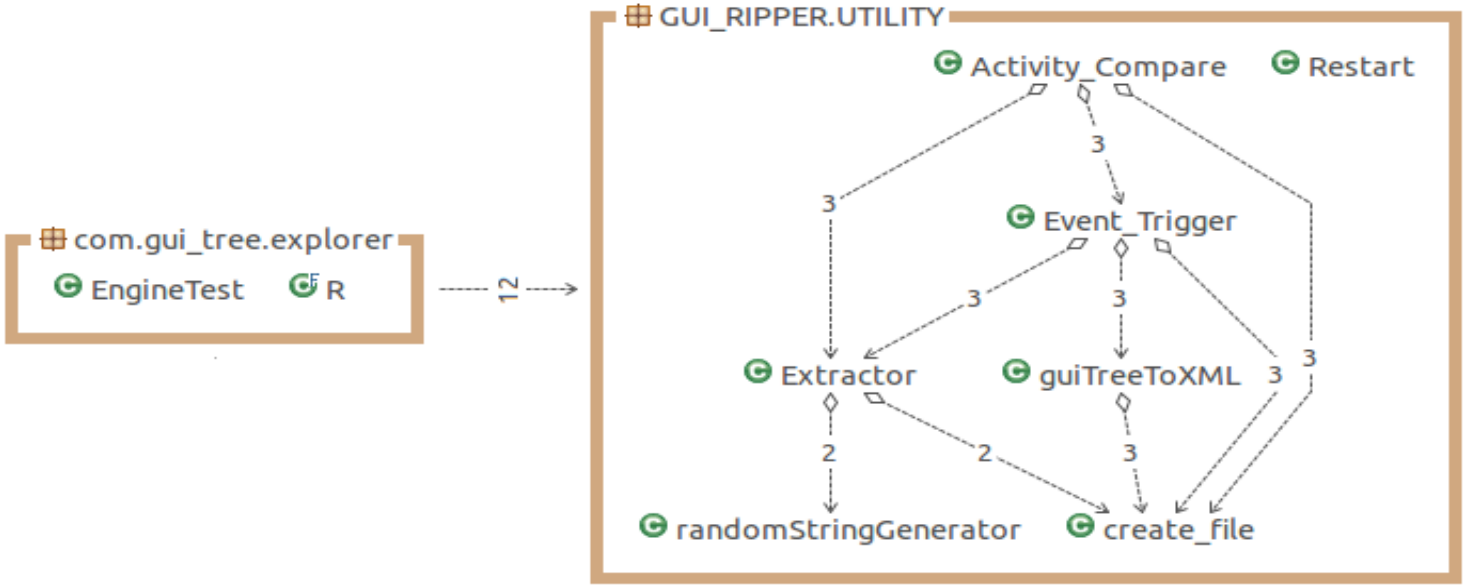
- Il GUI-ripping, è un processo dinamico utilizzato per esplorare automaticamente la GUI di una applicazione.

- In particolare tramite il GUI-ripping, verranno analizzate le varie interfacce che compongono l'applicazione analizzando i widget che vanno a definire l'interfaccia.

- Con il GUI-Ripping è possibile ottenere il modello del GUI-Tree, attraverso un algoritmo di esplorazione automatica delle GUI guidato dagli eventi scatenati sulle interfacce.

- Il modello ricavato via GUI Ripping è la base per la generazione automatica di casi di test

Struttura del tool sviluppato



1. **com.gui_tree.Explorer:**
   - è composto dalla sola classe EngineTest.java, la quale a sua volta contiene il metodo di test.
   - si occupa della chiamata alle varie funzioni che consentono l'esecuzione del tool.
2. **GUI_RIPPER.UTILITY:**
   - questo package contiene le classi che sono state create e che vengono richiamate da EngineTest
   - esegue l'algoritmo di esplorazione per rilevare i widget che possono essere scatenati

Utilizzo del tool sviluppato

Configurazione del tool:

- Eliminazione firma sviluppatore
- Installazione apk su emulatore Android
- Inserimento package e nome prima Activity dell'apk

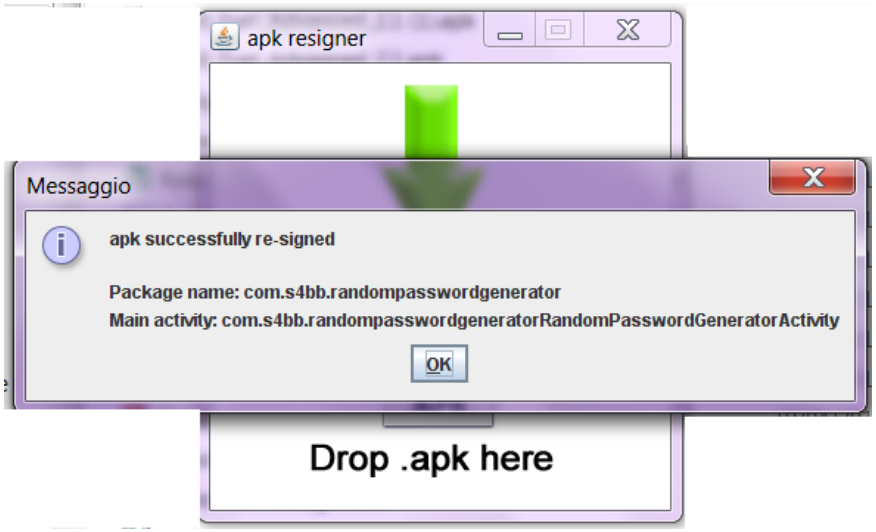
Il tool di Ripper avrà il seguente comportamento in pseudocodice:

```
1   avvia applicazione;
2   individua i widget dell'interfaccia;
3   push (pila, lista eventi scatenabili sui widget);
4   while (pila non è vuota){
4           Evento e = pop(pila);
5           riavvia applicazione e raggiungi l'interfaccia e.interfaccia;
6           Inserisci input e scatena e ;
7           if(l'interfaccia ottenuta è inedita){
8                   push (pila, lista eventi scatenabili sui widget);
9                   }
10          }
11  output formato xml contenente la descrizione delle interfacce attraversate e degli eventi
    scatenati;
```

Tools utilizzati

Per implementare il processo di testing ed effettuare gli esperimenti sono stati utilizzati:

- **Android Virtual Device**
  - Crea un dispositivo virtuale con determinate caratteristiche hardware.
- **Android Emulator**
  - Emula il funzionamento del sistema operativo Android.
- **Robotium**
  - Framework utile per l'automatizzazione dell'interazione con la GUI.
- **Re-sign**
  - Framework utile per eliminare la firma dello sviluppatore dell'applicazione

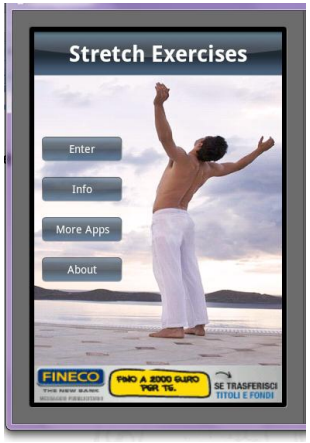


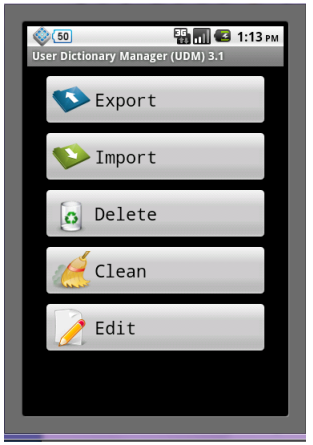
Sperimentazione

Obiettivo ->Valutare l'efficacia della tecnica nel rilevare i crash in applicazioni Android reali.

Sono state testate diverse applicazioni con caratteristiche diverse:

- **Stratch Exercises ->** Applicazione che provvede a mostrare come effettuare stetching senza recarsi in paletra, ma semplicemente da casa seguendo le istruzioni dell'applicazione.
- **Calorie Counter ->** Applicazione che calcola come ottenere il proprio peso forma in base all'area geografica in cui ci troviamo, permettendoci quindi di scegliere le tipologie di misurazione.
- **Calcolatrice ->** Applicazione che simula una calcolatrice scientifica.
- **Mileage Tracker ->** Applicazione che tiene traccia del chilometraggio effettuato con la propria vettura.
- **User Dictionary Manager ->** Applicazione che consente all'utente di calcolare il proprio indice di massa corporea.





Esperimenti condotti

- Di seguito riportiamo una tabella con i risultati conseguiti dal testing delle applicazioni.
- Vengono riportati anche i limiti del tool sviluppato, che sono stati individuati durante il test di alcune applicazioni.

Nome dell'applicazione	Profondità albero GUI	Numero tracce di test raccolte	Tempo impiegato dal tool	Limiti del tool rilevati	Crash dell'applicazione rilevato
Stretch Exercises	4	8	212,326 s	Nessuno	Nessuno
Calorie Calculator	3	6	182,254 s	Nessuno	Nessuno
Calcolatrice	4	13	337,740 s	Nessuno	NumberFormatException
Mileage Tracker	2	6	97,881 s	Impossibilità nel testare editText predefinite	ActivityNotFoundException
User Dictionary Manager	3	7	102,55 s	Nessuno	Nessuno

Conclusioni

- Il tool è in grado di effettuare il ripping di una interfaccia utente e allo stesso tempo di memorizzare le informazioni riguardanti le interfaccie esplorate
- Il tool è stato in grado di rilevare dei crash in alcune applicazioni testate
- Sono stati individuati diversi limiti del tool

Sviluppi futuri

- Implementare il tool con l'esecuzione di ulteriori sperimentazioni protese a validare maggiormente sia il processo che il tool sviluppato.
- Implementare il tool considerando nuovi eventi da scatenare su nuovi widget
- Considerare altri widget come input