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

Tesi di laurea specialistica

SPERIMENTAZIONI DI TECNICHE DI TESTING STATICO PER APPLICAZIONI ANDROID

Anno Accademico 2011/2012

Relatore
Prof. Porfirio Tramontana

Candidato
Pasquale Giacomino
Matr. 885/349

Scopo

Sperimentazioni di Tecniche di Testing statico per applicazioni Android

Obiettivo

Analizzare e confrontare le Tecniche di Testing Statico per applicazioni Android

Android

Il mercato smartphone è dominato da Android.

Worldwide Smartphone Sales to End Users by Operating System in 1Q12 (Thousands of Units)

Operating System	1Q12 Units	1Q12 Market Share (%)	1Q11 Units	1Q11 Market Share (%)
Android	81,067.4	56.1	36,350.1	36.4
iOS	33,120.5	22.9	16,883.2	16.9
Symbian	12,466.9	8.6	27,598.5	27.7
Research In Motion	9,939.3	6.9	13,004.0	13.0
Bada	3,842.2	2.7	1,862.2	1.9
Microsoft	2,712.5	1.9	2,582.1	2.6
Others	1,242.9	0.9	1,495.0	1.5
Total	**144,391.7**	**100.099,775.0**		**100.0**

Source: Gartner (May 2012)

Testing Dinamico

E' il processo di valutazione di un sistema software o di un suo componente basato sull'osservazione del suo comportamento in esecuzione.

Testing Statico

E' il processo di valutazione di un sistema o di un suo componente basato sulla sua forma, sulla sua struttura, sul suo contenuto o sulla documentazione.

Android Lint

✓ E' un nuovo strumento introdotto in ADT 16 creato da Google
✓ Analizza il codice sorgente di un progetto Android in cerca di potenziali errori.

Android Lint si presta soprattutto alla scoperta delle seguenti classi di problemi o errori :

- Traduzioni e traduzioni mancanti;
- Layout
- Problemi di prestazioni;
- Risorse non utilizzate ;
- Dimensioni matrice incoerenti;
- Problemi di accessibilità e internazionalizzazione;
- Problemi di icone;
- Problemi di usabilità ;
- Errori nel manifest.

FindBugs

✓ E' uno strumento open source.
✓ Ispeziona possibili errori di programmazione

FindBugs si presta soprattutto alla scoperta delle seguenti classi di problemi o errori:

• Problemi di correttezza per thread singoli;
• Problemi di correttezza nella sincronizzazione di più thread cooperanti;
• Questioni di performance;
• Problemi di sicurezza o vulnerabilità del codice;

CheckStyle

✓ E' uno strumento open source.
✓ Aiuta a garantire che il codice Java aderisca ad una serie di standard di codifica.

CheskStyle si presta soprattutto alla scoperta delle seguenti classi di problemi o errori:

• Commenti Javadoc per classi, attributi e metodi;
• Convenzioni di denominazione per attributi e metodi;
• Limite del numero di parametri per le funzioni;
• Presenza di intestazioni obbligatorie;
• L'uso delle importazioni di pacchetti e di classi;
• Best practices per la costruzione delle classi;
• Controllo codice duplicato.

CodePro Analytix

- ✓ E' uno strumento dinamico creato da Google.
- ✓ Rileva, il mancato rispetto degli standard di codifica predefiniti, sicurezza e convenzioni di stili.

CodePro Analytix si presta soprattutto alla scoperta delle seguenti classi di problemi o errori:

- Stile di codifica
- Commenti
- Codice morto
- Eccezioni
- Formattazione
- Utilizzo di importazione

PMD

✓ E' un strumento open source creato da un gruppo.
✓ Effettua un'analisi sintattica del programma o della libreria.

PMD si presta soprattutto alla scoperta delle seguenti classi di problemi o errori:

- Possibili bug;
- Codice morto;
- Codice non ottimale;
- Espressioni troppo complicate;
- Codice duplicato.

MotoDev App Validator

✓ E' un strumento open source creato da Motorola.
✓ Effettua un'analisi statica del programma.

MotoDev App Validator si presta soprattutto alla scoperta delle seguenti classi di problemi o errori:

- Esamina le autorizzazioni;
- Cerca situazioni in cui l'applicazione potrebbe essere respinta da Google Play;
- Verifica la compatibilità del dispositivo;
- Problemi di traduzione;
- Dimensionamento immagini;
- Problemi con il codice Java;
- Errori nel manifest.

Sperimentazione

Sono state selezionate cinque applicazioni Android Open Source

- Aard Dictionary 1.4.1;
- And Bible 1.3.0;
- Book Catalogue 3.8.1;
- TomDroid 0.5.0;
- WordPress for Android r394.

Nessun tool è in grado di dare la certezza dell'assenza di problemi nel codice. La ricerca di errori di programmazione soffre essenzialmente di due problematiche:

falsi positivi: alcuni dei problemi individuati in realtà non lo sono.

falsi negativi: non tutti i bug presenti nel codice vengono individuati.

Risultati Sperimentazione

App	Tool	Rule	Quantità
AndBible	*Lint*	Performance	44
AndBible	*FindBugs*	Performance	16
AndBible	*CheckStyle*	Class Design	741
AndBible	*CodePro*	Performance	66
AndBible	*PMD*	Design	419
AndBible	*MotoDev*	Layout Checker	30
BookCatalogue	*Lint*	Performance	151
BookCatalogue	*FindBugs*	Performance	26
BookCatalogue	*CheckStyle*	Class Design	707
BookCatalogue	*CodePro*	Performance	66
BookCatalogue	*PMD*	Design	419
BookCatalogue	*MotoDev*	Layout Checker	45
TomDroid	*Lint*	Performance	28
TomDroid	*FindBugs*	Performance	11
TomDroid	*CheckStyle*	Class Design	118
TomDroid	*CodePro*	Performance	73
TomDroid	*PMD*	Design	82
TomDroid	*MotoDev*	Layout Checker	0

Risulati Sperimentazione

App	Tool	Rule	Quantità
ArdDictionary	*Lint*	Performance	6
ArdDictionary	*FindBugs*	Performance	2
ArdDictionary	*CheckStyle*	Class Design	149
ArdDictionary	*CodePro*	Performance	20
ArdDictionary	*PMD*	Design	68
ArdDictionary	*MotoDev*	Layout Checker	0
WordPress	*Lint*	Performance	235
WordPress	*FindBugs*	Performance	18
WordPress	*CheckStyle*	Class Design	738
WordPress	*CodePro*	Performance	50
WordPress	*PMD*	Design	312
WordPress	*MotoDev*	Layout Checker	72

Conclusioni

- Code Pro Analityx è semplice da configurare, a differenza degli altri trova meno falsi problemi;
- con PMD è semplice costruire nuove regole inoltre cura più la forma come le parentesi graffe;
- FindBugs verifica il consumo di risorse e problemi architetturali;
- CheckStyle è uno strumento per la ricerca di caratteri di tabulazione molte volte inutile;
- Lint e MotoDev possono lavorare in sincronia e funzionano bene per problematiche che riguardano esclusivamente Android;

Non esistono strumenti che garantiscano una soluzione completa al problema dell'analisi e della verifica di codice Java al fine di scoprire potenziali errori. Come si evince dalla sperimentazione tutti i tool trovano possibili errori, ma sarà compito del programmatore utilizzare o meno l'avviso.

Sviluppi futuri

- ✓ Inserire annotazioni all'interno del codice sotto analisi.
- ✓ Miglioramento degli stessi algoritmi di analisi e l'introduzione di tecniche di filtraggio dell'output.