Università degli Studi di Salerno

DIPARTIMENTO DI SCIENZE AZIENDALI - MANAGEMENT & INNOVATION SYSTEMS

Dottorato di Ricerca in
Management and Information Technology
Curriculum Informatica, Sistemi Informativi e Tecnologie del Software
Ciclo XVI

Tesi di dottorato in:

Methods and Tools for Focusing and Prioritizing the Testing Effort

Coordinatore del Dottorato:
Prof. Andrea De Lucia

Tutor:
Prof. Andrea De Lucia

Candidato:
Dario Di Nucci

Anno Accademico 2016-2017

SOMMARIO

Il testing del software è largamente riconosciuto come una parte essenziale del processo di sviluppo software, rappresentando comunque un'attività estremamente costosa. Il costo totale del testing è stato stimato costituire almeno metà del costo totale di sviluppo. Nonostante la sua importanza, comunque, studi recenti hanno mostrato che gli sviluppatori raramente testano le loro applicazioni e la maggioranza delle sessioni di programmazione finiscono senza che nessun test sia stato eseguito. Quindi, nuovi metodi e strumenti capaci di meglio allocare gli sviluppatori sono necessari per aumentare la robustezza dei sistemi e ridurre i costi del testing.

Le risorse disponibili dovrebbero essere efficacemente allocate tra le parti del codice sorgente che hanno più probabilità di contenere difetti. In questa tesi ci focalizziamo su tre attività per focalizzare e prioritizzare il costo del testing, in particolare predizione dei difetti, prioritizzazione dei casi di test e rilevazione dei code smell relativi a problemi energetici. Quindi, nonostante lo sforzo profuso dalla comunità scientifica nelle ultime decadi attraverso la conduzione di studi empirici e la proposta di nuovi approcci che hanno portato risultati interessanti, nel contesto della nostra ricerca abbiamo sottolineato alcuni aspetti che potrebbero essere migliorati e proposto studi empirici e nuovi approcci.

Nel contesto della predizione dei difetti, abbiamo proposto due nuove misure, DEVELOPER'S STRUCTURAL AND SEMANTIC SCATTERING. Queste metriche sfruttano la presenza di cambiamenti dispersi che rendono gli sviluppatori più inclini ad introdurre difetti. I risultati del nostro studio empirico mostrano la superiorità del nostro modello rispetto a quelli basati su metriche di processo e di prodotto. In seguito, abbiamo sviluppato un modello "ibrido" che fornisce un miglioramento medio in termini di accuratezza. Oltre ad analizzare i predittori, abbiamo sviluppato un nuovo classificatore adattivo, che dinamicamente raccomanda il classificatore capace di predire in maniera migliore la difettosità di una classe, basandosi sulle caratteristiche strutturali della stessa. I modelli basati su questo classificatori riesco ad essere più efficaci rispetto a quelli basati su classificatori semplici, così come quelli basati sulla tecnica di ensemble detta VALIDATION AND VOTING nel contesto della predizione dei difetti intra-progetto. In seguito

abbiamo proposto uno studio replica nel contesto della predizione di difetti intra- e inter-progetto. Abbiamo analizzato il comportamento di sette metodi ensemble. I risultati mostrano che il problema è ancora lontano dall'essere risolto e che l'uso delle tecniche di ensemble non fornisce benefici evidenti rispetto ai classificatori semplici, indipendentemente dalla strategia utilizzata per costruire il modello. Infine, abbiamo confermato, nel contesto dei modelli basati su tecniche di ensemble, i risultati di studi precedenti che hanno dimostrato che i modelli per la predizione dei difetti inter-progetto funzionano peggio di quelli intra-progetto, essendo comunque più robusti alla variabilità delle performance.

Rispetto al problema di prioritizzazione dei casi di test, abbiamo proposto un algoritmo genetico basato sull'indicatore dell'ipervolume. Abbiamo fornito una validazione estesa degli approcci basati sull'ipervolume e dello stato dell'arte utilizzando fino a cinque criteri di testing. I nostri risultati suggeriscono che l'ordinamento fornito da HGA è più efficace rispetto a quelli prodotti dagli algoritmi dello stato dell'arte. Inoltre, il nostro algoritmo è molto più veloce e la sua efficacia non diminuisce quando la dimensione del programma software o della test suite cresce.

Per gestire i problemi relativi all'efficienza energetica delle applicazioni mobile e quindi ridurre il costo del testing di questo aspetto non funzionale, abbiamo sviluppato due nuovi strumenti software. PETrA è capace di estrarre il profilo energetico delle applicazioni mobile, mentre aDoctor è un rilevatore di code smell capace di identificare 15 dei code smells specifici per applicazioni Android definiti da Reimann *et al.*. Abbiamo analizzato l'impatto di questi smell, attraverso un grande studio empirico con l'obiettivo di determinare in che modo i code smell relativi ai metodi del codice sorgente delle applicazioni mobile influenzano il consumo energetico e se le operazioni di refactoring applicate per rimuoverli migliorano l'efficienza energetica dei metodi rifattorizzati. I risultati del nostro studio sottolineano che i metodi affetti da code smell consumano fino a 385% più energia rispetto ai metodi non affetti da smell. Un'analisi a grana fine rivela l'esistenza di quattro energy smell. Infine, abbiamo sottolineato l'utilità del refactoring come un mezzo per migliorare l'efficienza energetica attraverso la rimozione dei code smell. In dettaglio, abbiamo trovato che sia possibile migliorare l'efficienza energetica dei metodi del codice sorgente fino al 90% attraverso il refactoring dei code smell.

Infine, forniamo un insieme di problemi aperti che dovrebbero essere affrontati dalla comunità scientifica nel futuro.