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

ABSTRACT

Software testing is widely recognized as an essential part of any software development process, representing however an extremely expensive activity. The overall cost of testing has been estimated at being at least half of the entire development cost, if not more. Despite its importance, however, recent studies showed that developers rarely test their application and most programming sessions end without any test execution. Indeed, new methods and tools able to better allocating the developers effort are needed in order to increment the system reliability and to reduce the testing costs.

The resources available should be allocated effectively upon the portions of the source code that are more likely to contain bugs. In this thesis we focus on three activities able to prioritize the testing effort, specifically bug prediction, test case prioritization, and detection of code smell able to fix energy issues. Indeed, despite the effort devoted by the research community in the last decades through the conduction of empirical studies and the devising of new approaches led to interesting results, in the context of our research we highlighted some aspects that might be improved and proposed empirical investigations and novel approaches.

In the context of bug prediction, we devised two novel metrics, namely the DEVELOPER'S STRUCTURAL AND SEMANTIC SCATTERING. These metrics exploit the presence of scattering changes that make developers more error-prone. The results of our the empirical study show the superiority of our model with respect to baselines based on product metrics and process metrics. Afterwards, we devised a "hybrid" model providing an average improvement in terms of prediction accuracy. Besides analyzing on predictors, we proposed a novel adaptive prediction classifier, which dynamically recommends the classifier able to better predict the bug-proneness of a class, based on the structural characteristics of the class. The models based on this classifier are able to outperform models based on stand-alone classifiers, as well as those based on the VALIDATION AND VOTING ensemble technique in the context of within-project bug prediction. Laterly, we performed a differentiated replication study in the contexts of cross-project and within-project bug prediction. We analyzed the behavior of seven ensemble methods. The results show that the problem is still far from being solved and that the use of ensemble techniques does not provide evident benefits with respect to

stand-alone classifiers, independently from the strategy adopted to build model. Finally, we confirmed, in the context of ensemble-based models, the findings of previous studies that demonstrated that cross-project bug prediction models perform worse than within-project ones, being however more robust to performance variability.

With respect to the test case prioritization problem, we proposed a genetic algorithm based on the hypervolume indicator. We provided an extensive evaluation of Hypervolume-based and state-of-the-art approaches when dealing with up to five testing criteria. Our results suggest that the test ordering produced by HGA is more cost-effective than those produce by state-of-the-art algorithms. Moreover, our algorithm is much more faster and its efficiency does not decrease as the size of the software program and of the test suite increase.

To cope with energy efficiency issues of mobile applications and thus reducing the effort needed to test this non-functional aspect, we devised two novel software tools. PETrA is able to extract the energy profile of mobile applications, while aDoctor is a code smell detector able to identify 15 Android-specific code smells defined by Reimann *et al.*. We analyzed the impact of these smells, by a large empirical study with the aim of determining to what extent code smells affecting source code methods of mobile applications influence energy efficiency and whether refactoring operations applied to remove them directly improve the energy efficiency of refactored methods. The results of our study highlight that methods affected by code smells consume up to 385% more energy than methods not affected by any smell. A fine-grained analysis reveals the existence of four specific energy-smells. Finally, we also shed light on the usefulness of refactoring as a way for improving energy efficiency by code smell removal. Specifically, we found that it is possible to improve the energy efficiency of source code methods by up to 90% through refactoring code smells.

Finally, we provide a set of open issues that should be addressed by the research community in the future.