Mobile App and App Store Analysis, Testing and Optimisation

M. Harman, A. Al-Subaihin, Y. Jia, W. Martin, F. Sarro and Y. Zhang
University College London, CREST Centre UCLappA Group, London, WC1E 6BT, UK.

ABSTRACT
This talk presents results on analysis and testing of mobile apps and app stores, reviewing the work of the UCL App Analysis Group (UCLappA) on App Store Mining and Analysis. The talk also covers the work of the UCL CREST centre on Genetic Improvement, applicable to app improvement and optimisation.

Categories and Subject Descriptors
D.2.5 [Software Engineering]

Keywords
Mining Software Repositories, App Store Mining, App Store Analysis, Software Testing, Energy Consumption

1. APP STORE ANALYSIS
This two page summary paper provides an outline of the material presented in the keynote talk at MobileSoft 2016 by Mark Harman, with pointers to the literature for details of the results covered. The focus of the keynote is app store mining and analysis. We start from the position that app stores provide a wealth of information making them well-suited to software repository mining. However, unlike traditional software repository mining [15], app store mining benefits from a combination of technical, business and customer facing information [10].

App stores began to appear in 2008 and have quickly become populated with millions of apps, instantly providing feedback between users and developers. Early studies found a correlation between rating and popularity [10]. This provided the initial evidence that developers would clearly need to take an interest in such data, mined from app stores. These findings have made the analysis of app reviews an interesting research problem aimed at helping developers to manage the wealth of information newly available to them [7, 16, 17]. Nevertheless, care is required, because such research must account for the app sampling problem [20].

2. APP TESTING AND OPTIMISATION
Recent advances in automated test input generation [11] make it possible to automatically generate inputs that cover, for example, white box structure or subtle faults that may be present [9, 23]. Such advances in testing have helped make possible a new approach to software improvement that has come to be known as ‘Genetic Improvement’ [13, 18, 26].

Genetic Improvement treats the source code of an existing system as genetic material, using computational search to find new versions of the system that improve some property of interest, while remaining otherwise faithful to the behaviour of the original. The original program is used as test oracle [3], while automated test data generation is used to assess faithfulness. Recent empirical results concerning code uniqueness [5] and the graftability of code modifications from existing code bases [1] have provided evidence that large existing systems contain a surprising amount of useful ‘genetic material’ for such improvement.

From the point of view of mobile apps, there are many attractive targets for genetic improvement, including energy, bandwidth and execution time reduction. The keynote will conclude with some recent results from our work on genetic improvement for energy optimisation [4], deep parameter exposition [27], dreaming smart phones [8] and automated software transplantation [2, 12, 14], explaining how these techniques can be used for mobile app optimisation.

3. REFERENCES


