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].

App Stores can also be mined for relationships *between different apps* (as well as the data available for a *particular* app of interest). Such cross-app analysis can allow developers to understand the market place into which they seek to deploy their products. The keynote covers recent results on migration of features through app stores, obtained by mining app descriptions [24]. It also briefly reviews recent results [21, 19] on the impact of releases (relative to previous performance). We argue that causal impact analysis has a significant role to play in App Store Mining and Analysis.

There are many other exciting directions for App Store Analysis, and time sadly only permits a few of these directions to be explored in the keynote. For example, Gorla et al. [6] used API calls to understand how anomalous API calls can highlight aberrant or otherwise suspicious behaviour, while Syer et al. [25] used API calls to understand the relationship between defects and platform dependence. A comprehensive survey of App Store Analysis for Software Engineering can be found in the authors' recent survey [22].

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.

Acknowledgement: Work funded by EPSRC programme grant DAASE: Dynamic Adaptive Automated Software Engineering.

MobileSoft'16, May 16-17, 2016, Austin, TX, USA
© 2016 ACM. ISBN 978-1-4503-3900-1/16/05…$15.00
DOI: http://dx.doi.org/10.1145/2884781.2884791

3. REFERENCES

[1] E. T. Barr, Y. Brun, P. Devanbu, M. Harman, and F. Sarro. The plastic surgery hypothesis. In 22nd *ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE 2014)*, pages 306–317, Hong Kong, China, November 2014.

[2] E. T. Barr, M. Harman, Y. Jia, A. Marginean, and J. Petke. Automated software transplantation. In *Proceedings of the 2015 International Symposium on Software Testing and Analysis, ISSTA 2015, Baltimore, MD, USA, July 12-17, 2015*, pages 257–269, 2015.

[3] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo. The oracle problem in software testing: A survey. *IEEE Transactions on Software Engineering*, 41(5):507–525, May 2015.

[4] B. Bruce, J. Petke, and M. Harman. Reducing energy consumption using genetic improvement. In *Genetic and evolutionary computation conference (GECCO 2015)*, pages 1327–1334, Madrid, Spain, July 2015.

[5] M. Gabel and Z. Su. A study of the uniqueness of source code. In 18th *ACM SIGSOFT international symposium on foundations of software engineering (FSE 2010)*, pages 147–156, Santa Fe, New Mexico, USA, 7-11 Nov. 2010. ACM.

[6] A. Gorla, I. Tavecchia, F. Gross, and A. Zeller. Checking app behavior against app descriptions. In *36th International Conference on Software Engineering (ICSE 2014)*, pages 1025–1035, 2014.

[7] E. Guzman and W. Maalej. How do users like this feature? a fine grained sentiment analysis of app reviews. In *Requirements Engineering (RE 2014)*, pages 153–162, Aug 2014.

[8] M. Harman, Y. Jia, W. B. Langdon, J. Petke, I. H. Moghadam, S. Yoo, and F. Wu. Genetic improvement for adaptive software engineering. In 9th *International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2014)*, pages 1–4, New York, NY, USA, 2014. ACM.

[9] M. Harman, Y. Jia, P. R. Mateo, and M. Polo. Angels and monsters: an empirical investigation of potential test effectiveness and efficiency improvement from strongly subsuming higher order mutation. In *ACM/IEEE International Conference on Automated Software Engineering (ASE '14)*, pages 397–408, 2014.

[10] M. Harman, Y. Jia, and Y. Zhang. App store mining and analysis: MSR for App Stores. In 9th *Working Conference on Mining Software Repositories (MSR 2012)*, Zurich, Switzerland, 2-3 June 2012.

[11] M. Harman, Y. Jia, and Y. Zhang. Achievements, open problems and challenges for search based software testing (keynote). In 8th *IEEE International Conference on Software Testing, Verification and Validation (ICST 2014)*, Graz, Austria, April 2015.

[12] M. Harman, W. B. Langdon, and Y. Jia. Babel pidgin: SBSE can grow and graft entirely new functionality into a real world system. In *SSBSE*, 2014.

[13] M. Harman, W. B. Langdon, Y. Jia, D. R. White, A. Arcuri, and J. A. Clark. The GISMOE challenge: Constructing the pareto program surface using genetic programming to find better programs (keynote paper). In 27th *IEEE/ACM International Conference on Automated Software Engineering (ASE 2012)*, pages 1–14, Essen, Germany, September 2012.

[14] M. Harman, W. B. Langdon, and W. Weimer. Genetic programming for reverse engineering (keynote paper). In R. Oliveto and R. Robbes, editors, 20th *Working Conference on Reverse Engineering (WCRE 2013)*, Koblenz, Germany, 14-17 October 2013. IEEE.

[15] A. E. Hassan. The Road Ahead for Mining Software Repositories. In *Proceedings of the Interlational Conference on Frontiers of Software Maintenance (FoSM '08)*, pages 48–57, Beijing, China, 28 Sept.-4 Oct. 2008. IEEE.

[16] C. Iacob and R. Harrison. Retrieving and Analyzing Mobile App Feature Requests from Online Reviews. In *Proceedings of the 10th Working Conference on Mining Software Repositories (MSR '13)*, San Francisco, California, USA, 18-19 May 2013.

[17] H. Khalid, E. Shihab, M. Nagappan, and A. Hassan. What do mobile app users complain about? A study on free iOS apps. *IEEE Software*, 32(3):70–77, 2014.

[18] W. B. Langdon and M. Harman. Optimising existing software with genetic programming. *IEEE Transactions on Evolutionary Computation (TEVC)*, 19(1):118–135, Feb 2015.

[19] W. Martin. Causal impact for app store analysis. In *ICSE Student Research Competition*, 2016.

[20] W. Martin, M. Harman, Y. Jia, F. Sarro, and Y. Zhang. The app sampling problem for app store mining. In *12th IEEE/ACM Working Conference on Mining Software Repositories, MSR 2015, Florence, Italy, May 16-17, 2015*, pages 123–133, 2015.

[21] W. Martin, F. Sarro, and M. Harman. Causal impact analysis applied to app releases in google play and windows phone store. Technical Report RN/15/07, UCL Computer Science Department, December 2015.

[22] W. Martin, F. Sarro, Y. Jia, Y. Zhang, and M. Harman. A survey of app store analysis for software engineering. Technical Report RN/16/02, UCL Computer Science Department, January 2016.

[23] E. Omar, S. Ghosh, and D. Whitley. Comparing search techniques for finding subtle higher order mutants. In *Conference on Genetic and Evolutionary Computation (GECCO 2014)*, pages 1271–1278. ACM, 2014.

[24] F. Sarro, A. A. Al-Subaihin, M. Harman, Y. Jia, W. Martin, and Y. Zhang. Feature lifecycles as they spread, migrate, remain, and die in app stores. In *23rd IEEE International Requirements Engineering Conference, RE 2015, Ottawa, ON, Canada, August 24-28, 2015*, pages 76–85, 2015.

[25] M. D. Syer, M. Nagappan, B. Adams, and A. E. Hassan. Studying the relationship between source code quality and mobile platform dependence. *Software Quality Journal*, 2014. To appear; available online.

[26] D. R. White, J. Clark, J. Jacob, and S. Poulding. Searching for resource-efficient programs: Low-power pseudorandom number generators. In *2008 Genetic and Evolutionary Computation Conference (GECCO 2008)*, pages 1775–1782, Atlanta, USA, July 2008.

[27] F. Wu, M. Harman, Y. Jia, J. Krinke, and W. Weimer. Deep parameter optimisation. In *Genetic and evolutionary computation conference (GECCO 2015)*, pages 1375–1382, Madrid, Spain, July 2015.