ISSN 2186-7437

NII Shonan Meeting Report

No. 2015-15

Mobile App Store Analytics

Meiyappan Nagappan
Ahmed E. Hassan
Yasutaka Kamei

October 19–22, 2015

National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-Ku, Tokyo, Japan

Mobile App Store Analytics

Organizers:
Meiyappan Nagappan (Rochester Institute of Technology)
Ahmed E. Hassan (Queen's University)
Yasutaka Kamei (Kyushu Univeristy)

October 19–22, 2015

Today, software engineering research focuses on traditional software systems like the Firefox web browser or Microsoft Windows, which take years to develop and teams of designers, developers and testers. Software engineering is rapidly changing though. Emerging domains, such as mobile devices, are growing rapidly and depend heavily on new software operating systems like Android and the applications that they run, commonly referred to as apps. Over the past few years, we have seen a boom in the popularity of mobile devices and mobile apps which run on these devices. Recent market studies predict that the global mobile app economy is expected to be worth $143 billion by 2016. Thus there exists considerable motivation for the research community to solve the challenges faced by the mobile app developers.

However, unlike traditional software, the distribution mechanism for mobile apps are very different — they are released through app markets (e.g., Google Play and Apple's App store). The key differentiating factor in an app store is that it is a democratic platform, i.e., both large companies with established products Adobe Reader from Adobe, and Timberman from Digital Melody (a company with 5 employees), can release their apps through the same mechanism for the users to download and install. The data that these mobile app markets contain can be used by software engineering researchers to compile new empirical results that can help mobile app developers.

Additionally, the app markets allow users to post reviews of the apps. This is very different from traditional software. Mobile app developers get continuous feedback from users that can be leveraged to help them. For example, prior work leveraged user reviews to extract user-faced issues and new requirements. However, today the review system for mobile apps is identical to that of books sold on an e-commerce website such as Amazon. While books are products too, they are very different from mobile apps in that books are not updated every few weeks like most mobile apps. Therefore in the case of books, the ratings and reviews collected for a book is all with respect to one version of a book, while the ratings and reviews collected for an app are about all the versions of an app. Hence the question arises whether the review systems of books is the best system for mobile apps or not?

Finally, the app stores provide a central location for all the apps, making it easy for researchers to mine the store for meta-data of the apps and the apps themselves. Using the data from the app stores, several companies like App

Annie, and Distimo have even built successful businesses selling intelligence gained from observing the evolution of several hundred thousand apps in the app stores.

However, there is no central venue to bring all the cross-disciplinary researchers together. There is therefore, a dire need for a community to be built around the line of research with respect to mobile app store analytics. The proposed seminar would focus on research where the Mobile App Stores, and the data that they have are mined, for insights into mobile app development. We intend to bring researchers in multiple disciplines from around the world in one place to discuss the future directions in the area of mobile app store analytics.

Meeting Schedule

- Oct. 18th, Sunday Evening
  - Welcome Reception
- Oct. 19th, Monday Morning
  - Opening
  - Self Introduction
  - Short Presentations
- Oct. 19th, Monday Afternoon
  - Short Presentations
  - Breakout Sessions
    - Linking App Store Data
    - Monetization in a 2-sided Market
    - Malware and Testing
- Oct. 20th, Tuesday Morning
  - Presentations from Each Breakout Session
  - Keynote: Audris Mockus
- Oct. 20th, Tuesday Afternoon
  - Short Presentations
  - Breakout Sessions
    - Cross Platform App Development
    - Data Repositories and Tools
    - Release Management
- Oct. 21st, Wednesday Morning
  - Presentations from Each Breakout Session
  - Keynote: Andrew Hoefel
  - Short Presentations
- Oct. 21st, Wednesday Afternoon
  - Excursion and Dinner in Kamakura
- Oct. 22nd, Thursday Morning
  - Keynote: Michele Lanza
  - Discussion on Mobile App Store Analytics Workshop
  - Wrap up

Overview of Talks

Reproducing Context-sensitive Crashes of Mobile Apps using Crowdsourced Monitoring

Bram Adams, Ecole Polytechnique de Montreal

While the number of mobile apps published by app stores keeps on increasing, the quality of these apps varies widely. Unfortunately, for many apps, end-users continue experiencing bugs and crashes once installed on their mobile device. While this is annoying for the end users, it definitely is for the developers of an app, as they need to determine as fast as possible how to reproduce reported crashes before finding the root cause of the crashes. Given the heterogeneity in hardware, mobile platform releases, and types of users, the reproduction step currently is one of the major challenges of app developers. This presentation therefore introduces MoTiF, a crowdsourced approach to support developers in automatically reproducing context-sensitive crashes faced by end-users in the wild. In particular, by analyzing recurrent patterns in crash data, the shortest sequence of events reproducing a crash is derived, and turned into a test suite. We evaluated MoTiF on concrete crashes that were crowd-sourced or randomly generated on 5 Android apps, showing that MoTiF can reproduce existing crashes effectively.

Improving Mobile Security with Static and Dynamic Code Analysis Techniques - Part 2

Steven Arzt, TU-Darmstadt

With more and more apps from various vendors on smartphones and tablets, it becomes vital to assess the quality and trustworthy of these apps. Do they steal private information or compromise the security of sensitive corporate information? Do they contain security vulnerabilities that can be exploited by attackers? With the size of current app stores, these questions can no longer be answered by investigating each app by hand. Instead, powerful code analysis techniques for mobile apps are required. In this presentation, we present the specific challenges for static and dynamic code analysis on Android and show how our toolchain addresses these challenges. We present the FlowDroid static data flow tracker, the DroidBench micro-benchmark suite, the Reviser algorithm for incremental analysis, the SuSi tool for automatically identifying sources and sinks, and the DroidForce tool for enforcing runtime policies. We conclude on how these approaches and tools can be used as building blocks for future research in the field.

Mining behavior of Android apps: Can test input generation tools help?

Alessandra Gorla, IMDEA Software Institute

Most mining techniques for Android use static information such as metadata listed in the app manifest, and use static analysis to extract features describing the behavior from implementation. In this talk we will discuss how test

input generation techniques for Android can be used to collect more accurate information about how Android apps behave.

An Empirical Study of Emergency Updates for Top Android Mobile Apps

Safwat Hassan, Queen's University

The mobile app market continues to grow at a tremendous rate. The market provides a convenient and efficient distribution mechanism for updating apps. App developers continuously leverage such mechanism to update their apps at a rapid pace. The mechanism is ideal for releasing emergency updates (i.e., updates that are released soon after the previous update). In this talk we present and discuss our study for the emergency updates in the Google Play Store. Examining more than 44,000 updates of over 10,000 mobile apps in the Google Play Store, we identify 1,000 emergency updates. By studying the characteristics of such emergency updates, we find different patterns of emergency updates. We manually examine each pattern and document its causes and impact on users experience. App developers should carefully avoid these patterns in order to improve the experience of users.

Green-Star: Energy Star-like ratings for Apps

Abram Hindle, University of Alberta

Energy Star exists to rate home appliances and other machinery on its energy efficiency. An oven is measured by how much electricity it takes to cook, a refrigerator is measured on how much energy it needs to cool. But how we apply the same methodology to an app? Green-Star: Software Application Energy Consumption Ratings (SAECR) seeks to break down applications by their tasks and rank applications based on their energy efficiency per task. This allows stranded consumers to use the most energy efficient to maintain their cell phone's battery life, while making energy trade-offs when they have easy access to chargers. We argue that energy efficiency information is hard for developers and consumers to extract so we call for the energy efficiency certification of applications hosted on app stores with methodologies like Green Star/SAECR.

Improving mobile app testing with mobile store analysis

Yue Jia, University College London

Mobile App store not only brings new practices to software engineering, but also offers a wide and rich source of information about apps which has never before been available to software developers. This information includes both informal specifications and new metrics which can provide new opportunity to enhance testing in mobile apps. Missing informal specification can be automatically learnt from the descriptions and reviews of apps using natural language processing techniques, which can be used to construct new test cases. We can also prioritise test cases based on the importance of features which can be learnt from app popularity metrics, typically expressed in the number or

rank of downloads. This talk present some results on testing mobile apps using multi-objective search from the UCL app Analysis Group (UCLappA), and will give some directions for future work.

Finding anomalies in Android apps looking at what happens behind the scene

Konstantin Kuznetsov, Saarland University

Google Play store offers millions of applications that are advertised to serve almost any user needs. The real behavior of apps, though, can differ from what is announced. This may be caused by improper design, malicious payload included in the package, and defects in the code, just to mention a few. In this talk we discuss a group of techniques to identify abnormal, i.e. unexpected, behavior of Android application. All these techniques examine program features, which express the app behavior, and meta-data, which may provide a specific context to support learning. Our Chabada framework groups apps by their description topics, and then identifies outliers in each group with respect to their API usage. Our second technique - Mudflow - assesses sensitive data treatment. Assuming that similar sensitive data should be processed in similar way, it determines malicious apps as being anomalies against benign-ware according to their data flows. Finally, we present a concept of UI slicing - a method to associate app's user interface with program behavior and identify suspicious apps, whose behavior does not correspond to their appearance.

Country Differences in Mobile App User Behaviour and Challenges for Software Engineering

Soo Ling Lim, University College of London

Mobile app stores are highly competitive markets where developers need to cater to a large number of users spanning multiple countries. We hypothesized that there exist country differences in mobile app user behavior and conducted one of the largest surveys to date of app users across the world, in order to identify the precise nature of those differences. The survey investigated user adoption of the app store concept, app needs, and rationale for selecting or abandoning an app. We collected data from more than 15 countries, including USA, China, Japan, Germany, France, Brazil, UK, Italy, Russia, India, Canada, Spain, Australia, Mexico, and South Korea. Analysis of data provided by 4,824 participants showed significant differences in app user behaviors across countries, for example users from USA are more likely to download medical apps, users from UK and Canada are more likely to be influenced by price, users from Japan and Australia are less likely to rate apps. Analysis of the results revealed new challenges to market-driven software engineering related to packaging requirements, feature space, quality expectations, app store dependency, price sensitivity, and ecosystem effect.

Automated Analysis of Energy Efficiency and Execution Performance for Android Apps

Yepang Liu, Hong Kong Univ. of Science and Technology

Mobile applications' energy efficiency and performance have a vital impact on user experience. However, many mobile applications on market suffer from bugs that can cause significant energy waste and performance degradation, thereby losing their competitive edge. Locating these bugs is labor-intensive and thus automated diagnosis is highly desirable. Unfortunately, people have limited understanding of these bugs and there are no clear criteria to facilitate automated analysis of mobile applications' energy efficiency or execution performance. To bridge the gap, we conducted two large-scale empirical studies of real-world energy and performance bugs from popular Android applications. We studied the characteristics of these bugs and identified several common causes of energy waste and performance degradation.

For energy bugs, we observed that (1) forgetting to deactivate device sensors or wake locks after use and (2) ineffectively utilizing sensory data can cause serious energy waste. To help developers detect such energy bugs, we proposed a dynamic analysis technique GreenDroid. GreenDroid automatically generates user interaction event sequences to systematically execute an Android application for state space exploration. During execution, it tracks the transformation, propagation and consumption of sensory data and analyzes whether the data are effectively utilized by the application to bring users perceptible benefits. It also closely monitors whether device sensors and wake locks are properly deactivated after use. We evaluated GreenDroid using 14 popular open-source Android applications. GreenDroid successfully located 13 real energy bugs in these applications and additionally found two previously-unknown bugs that were later confirmed by developers.

For performance bugs, we observed that (1) conducting lengthy operations in an application's main thread and (2) frequently invoking heavy-weight program callbacks can seriously reduce the responsiveness of an application. To help developers detect such performance bugs, we designed a light-weight static analysis technique PerfChecker. PerfChecker automatically scans an Android application's bytecode and identifies a set of checkpoints whose efficiency is critical. It then analyzes whether the checkpoints' implementation satisfies the efficiency rules formulated from our empirical study. We evaluated PerfChecker with 39 popular and large-scale Android applications (29 open-source and 10 commercial) and a widely-used library. PerfChecker successfully detected 178 previously-unknown performance bugs, among which 88 were quickly confirmed by developers and 20 critical ones were fixed soon afterwards. We also confirmed via comparison experiments that fixing our detected performance bugs can significantly improve the performance of the corresponding applications.

Detection and Family Identification of Android Malware

Sam Malek, George Mason University

The number of Android malware apps are increasing very quickly. Simply detecting and removing malware apps is insufcient, since they can damage or

alter other les, data, or settings; install additional applications; etc. To determine such behavior, a security engineer can signicantly benet from identifying the specic family to which an Android malware belongs. Additionally, techniques for detecting Android malware, and determining their families, lack the ability to handle certain obfuscations that aim to thwart detection. Moreover, some prior techniques face scalability issues, preventing them from detecting malware in a timely manner. To address these challenges, we present a novel machine learning-based Android-malware detection and family-identication approach, RevealDroid, that leverages a small, simple set of selectable featuresof which the simplest set of features achieves obfuscation resiliency, efciency of analysis, and accuracy. This result is highly surprising, given that a wide variety of techniques require complex program analyses (e.g., precise data-ow analysis) or large sets of features (e.g., hundreds of thousands of features), leading to scalability problems and lack of resilience to obfuscation. Specically, our selected machine-learning features leverage categorized Android-API usage, which represent semantics of both benign and malicious Android apps. We assess RevealDroids accuracy and obfuscation resilience on an updated dataset of malware from a diverse set of families, including malware obfuscated using various transformations. We further compare RevealDroid against other state-of-the-art and state-of-the-practice approaches for malware detection and family identication.

Release Management in Mobile App Stores

Maleknaz Nayebi, University of Calgary

Large software organizations such as Facebook have had to invest heavily into a customized release strategy for their mobile apps as the vetting process of app stores introduces lag and uncertainty into the release process. To understand the common release strategies used for mobile apps, the results of two surveys with users and developers were discussed in conjunction with the release patterns being mined from mobile app stores. The results of our study suggests that an apps release strategy is a factor that affects the ongoing success of mobile apps. Moreover, the intensive availability of user feedback and involvement could be used for making better release decisions.

How Can I Improve My App? Classifying User Reviews for Software Maintenance and Evolution

Sebastiano Panichella, University of Zurich

App Stores, such as Google Play or the Apple Store, allow users to provide feedback on apps by posting review comments and giving star ratings. These platforms constitute a useful electronic mean in which application developers and users can productively exchange information about apps. Previous research showed that users feedback contains usage scenarios, bug reports and feature requests, that can help app developers to accomplish software maintenance and evolution tasks. However, in the case of the most popular apps, the large amount of received feedback, its unstructured nature and varying

quality can make the identification of useful user feedback a very challenging task. In this paper we present a taxonomy to classify app reviews into categories relevant to software maintenance and evolution, as well as an approach that merges three techniques: (1) Natural Language Processing, (2) Text Analysis and (3) Sentiment Analysis to automatically classify app reviews into the proposed categories. We show that the combined use of these techniques allows to achieve better results (a precision of 85% and a recall of 85%) than results obtained using each technique individually (precision of 70% and a recall of 67%). We implemented the proposed approach as a Java tool (available at http://www.ifi.uzh.ch/seal/people/panichella/tools/ARdoc.html), called ARdoc (App Reviews Development Oriented Classifier), that automatically recognize natural language fragments in user reviews that are relevant for developers to evolve their applications.

Challenges at Sony Mobile, Expectations on MSR

Junji Shimagaki; David Pursehouse, Sony Mobile Communications Inc.

(Disclaimer: This report is not representation of Sony Mobile Communications Inc. but of those presenters who participated in the Shonan meeting.)

As a mobile handset vendor which sells products globally, increasing number of localizations, product types and supported Android OS's enforced us to establish the "scalable" software development. Strategic branch planning reduced the cost of developing common software features. Automated software testing handles hundreds of thousands of tests for every commit created for our products. Delivery from vendors, i.e., Google and Qualcomm, is dealt with in a sophisticated manner so that it is integrated to our internal branches within a few days.

Sony Mobile's top-prio challenges are to identify (1) factors or people's behavior really matter to software quality (2) keys to conduct even more efficient software development with this multiple-system integration workflow and (3) the best practices of code review, which plays the significant role to improve the quality. Sony Mobile is currently giving significant time and effor to analyze our development data qualitatively and quantitatively and collaborating with external researchers to tackle with the challenges.

Improving Mobile Security with Static and Dynamic Code Analysis Techniques - Part 1

Siegfried Rasthofer, TU-Darmstadt

Android is one of the most popular operating system for smartphones and tablets. This popularity is also very attractive for attackers who try to gain money from victims by stealing sensitive data, sending premium messages or blackmailing users with ransomware. Android malware is very common which results in the need for (semi-) automatic code-analysis tools that try to identify the malicious behavior. The challenge hereby is the design of precise and efficient tools that are able to analyze applications with different complexity. In this talk I will give a brief overview about the Android security research in the Secure Software Engineering group at TU Darmstadt (Germany). In particular, I will

introduce different (semi-) automatic code analysis tools for inspecting Android applications. For instance, I will introduce Harvester which can be used as a fully-automatic de-obfuscator, and the Android Bytecode IDE for debugging compiled Android Applications on an intermediate representation.

Supporting Developers in Making Successful Apps

Federica Sarro, University College London

Recent work showed that the information available in the existing app stores can be mined and analysed to support developers in making successful apps [1]. In this talk I illustrate how we automatically mined key features from app descriptions and analysed their migration through product categories [2], and how we discovered impactful app releases relying only on the information available in existing app stores [3]. I also discuss open challenges such as making it more reliable [4] and actionable for developers.

[1] Afnan A. Al-Subaihin, Anthony Finkelstein, Mark Harman, Yue Jia, William Martin, Federica Sarro, Yuanyuan Zhang: App store mining and analysis. DeMobile@SIGSOFT FSE 2015: 1-2.

App stores are not merely disrupting traditional software deployment practice, but also offer considerable potential benefit to scientific research. Software engineering researchers have never had available, a more rich, wide and varied source of information about software products. There is some source code availability, supporting scientific investigation as it does with more traditional open source systems. However, what is important and different about app stores, is the other data available. Researchers can access user perceptions, expressed in rating and review data. Information is also available on app popularity (typically expressed as the number or rank of downloads). For more traditional applications, this data would simply be too commercially sensitive for public release. Pricing information is also partially available, though at the time of writing, this is sadly submerging beneath a more opaque layer of in-app purchasing. This talk will review research trends in the nascent field of App Store Analysis, presenting results from the UCL app Analysis Group (UCLappA) and others, and will give some directions for future work.

[2] Federica Sarro, Afnan A. Al-Subaihin, Mark Harman, Yue Jia, William Martin, Yuanyuan Zhang: Feature lifecycles as they spread, migrate, remain, and die in App Stores. RE 2015: 76-85.

We introduce a theoretical characterisation of fea- ture lifecycles in app stores, to help app developers to identify trends and to find undiscovered requirements. To illustrate and motivate app feature lifecycle analysis, we use our theory to empirically analyse the migratory and non-migratory behaviours of 4,053 non-free features from two App Stores (Samsung and BlackBerry). The results reveal that, in both stores, intransitive features (those that neither migrate nor die out) exhibit signifi- cantly different behaviours with regard to important properties, such as their price. Further correlation analysis also highlights differences between trends relating price, rating, and popularity. Our results indicate that feature lifecycle analysis can yield insights that may also

help developers to understand feature behaviours and attribute relationships.

[3] William Martin, Federica Sarro, Mark Harman, Causal Impact Analysis Applied to App Releases in Google Play and Windows Phone Store. Technical Report UCL - RN/15/07 (available at http://www.cs.ucl.ac.uk/fileadmin/UCL-CS/research/Research_Notes/RN_15_07.pdf).

App developers would like to know the characteristics of app releases that achieve high impact. To address this, we mined the most consistently popular Google Play and Windows Phone apps, once per week, over a period of 12 months. In total we collected 3,187 releases, from which we identified 1,547 for which there was adequate prior and posterior time series data to facilitate causal impact assessment, analysing the properties that distinguish impactful and non-impactful releases. We find that 40% of target releases impacted performance in the Google store and 55% of target releases impacted performance in the Windows store. We find evidence that more mentions of features and fewer mentions of bug fixing can increase the chance for a release to be impactful, and to improve rating.

[4] William Martin, Mark Harman, Yue Jia, Federica Sarro, Yuanyuan Zhang: The App Sampling Problem for App Store Mining. MSR 2015: 123-133.

Many papers on App Store Mining are susceptible to the App Sampling Problem, which exists when only a subset of apps are studied, resulting in potential sampling bias. We introduce the App Sampling Problem, and study its effects on sets of user review data. We investigate the effects of sampling bias, and techniques for its amelioration in App Store Mining and Analysis, where sampling bias is often unavoidable. We mine 106,891 requests from 2,729,103 user reviews and investigate the properties of apps and reviews from 3 different partitions: the sets with fully complete review data, partially complete review data, and no review data at all. We find that app metrics such as price, rating, and download rank are significantly different between the three completeness levels. We show that correlation analysis can find trends in the data that prevail across the partitions, offering one possible approach to App Store Analysis in the presence of sampling bias.

Commit Guru: Analytics and Risk Prediction of Software Commits

Emad Shihab, Concordia University

In this talk, we present Commit Guru, a language agnostic analytics and prediction tool that identifies and predicts risky software commits. Commit Guru is publicly available and is able to mine any GIT SCM repository. Analytics are generated at both, the project and commit levels. In addition, Commit Guru automatically identifies risky (i.e., bug-inducing) commits and builds a prediction model that assess the likelihood of a recent commit introducing a bug in the future. Finally, to facilitate future research in the area, users of Commit Guru can download the data for any project that is processed by Commit Guru with a single click. Several large open source projects have been successfully processed using Commit Guru. Commit Guru is available online at commit.guru.

Mobile App Store Feature Analysis

Yuanyuan Zhang, Research Associate, University College London

This talk introduces app store mining and analysis as a form of software repository mining. Unlike other software repositories traditionally used, app stores usually do not provide source code. However, they do provide a wealth of other information in the form of pricing and customer reviews. Therefore, we use data mining to extract feature information, which we then combine with more readily available information to analyse apps technical, customer and business aspects. We applied our approach to the apps available in the Blackberry app store. Our results show that there is a strong correlation between customer rating and the rank of app downloads, though perhaps surprisingly, there is no correlation between price and downloads, nor between price and rating. More importantly, we show that these correlation findings carry over to (and are even occasionally enhanced within) the space of data mined app features, providing evidence that our App store MSR approach can be valuable to app developers.

List of Participants

- Bram Adams, Ecole Polytechnique de Montreal
- Steven Arzt, TU-Darmstadt
- Christian Bird, Microsoft Research
- Daniel German, Univ. of Victoria
- Alessandra Gorla, IMDEA Software Institute
- Safwat Hassan, Queen's University
- Abram Hindle, University of Alberta
- Andrew Hoefel, Google
- Takashi Ishio, Osaka University
- Yue Jia, University College London
- Yasutaka Kamei, Kyushu University
- Raula Kula, Osaka University
- Konstantin Kuznetsov, Saarland University
- Michele Lanza, University of Lugano
- Soo Ling Lim, University College of London
- Yepang Liu, Hong Kong Univ. of Science and Technology
- Sam Malek, George Mason University
- Audris Mockus, the University of Tennessee
- Laura Moreno, University of Texas-Dallas
- Meiyappan Nagappan, Rochester Institute of Technology
- Maleknaz Nayebi, University of Calgary
- Sebastiano Panichella, University of Zurich
- Siegfried Rasthofer, TU-Darmstadt
- Federica Sarro, University College London
- Emad Shihab, Concordia University
- Yuanyuan Zhang, Research Associate, University College London
- Thomas Zimmermann, Microsoft Research

Visitors

- David Pursehouse, Sony Mobile Communications Inc.
- Junji Shimagaki, Sony Mobile Communications Inc.
- Toshiaki Tanaka, Sony Mobile Communications Inc.