ISSN 2186-7437

NII Shonan Meeting Report

No. 2013-12

Software Analytics
Principles and Practices

Ahmed E. Hassan
Katsuro Inoue
Tao Xie
Dongmei Zhang

October 21–25, 2013

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

Software Analytics
Principles and Practices

Organizers:
Ahmed E. Hassan (Queen's University/ BlackBerry)
Katsuro Inoue (Osaka University)
Tao Xie (University of Illinois at Urbana-Champaign)
Dongmei Zhang (Microsoft Research Asia)

October 21–25, 2013

A wealth of various data (e.g., source change history, test cases, and bug reports) exists in the practice of software development. Further modern software and services in operation produce rich data (e.g., operation logs, field crashes, and support calls). Hidden in these unexplored data is rich and valuable information about the quality of software and services and the dynamics of software development. Companies (Microsoft, Google, Facebook, Cisco, Yahoo, IBM, RIM, etc.) are increasingly adding analytics as an important role in their organizations, leveraging the wealth of various data produced around their software or services.

Software analytics is concerned with the use of data-driven approaches to obtain insightful and actionable information for completing various tasks around software systems, software users, and software development process. Insightful information is information that conveys meaningful and useful understanding or knowledge. Actionable information is information upon which software practitioners can come up with concrete solutions (better than existing solutions if any) towards completing tasks. Typically such information cannot be easily obtained by direct investigation on the raw data without the aid of analytic technologies.

Especially recently the area of Big Data has emerged as a critical and strategic focus by the society. Big data is everywhere now but it is still under-utilized in the area of software engineering. However, leveraging big data is very relevant in software engineering as software and services get larger and more interconnected, often being developed by a large number of engineers in distributed fashions and being used by a huge number of users around the world. Software analytics needs to be prepared for the upcoming decade's exciting and yet challenging problem of leveraging big data for software engineering tasks.

This meeting foster collaboration between industry and academia, bringing academic researchers working on the principles and practice of software analytics together with researchers from industry. The aim is not only to act as a forum for the exchange of ideas, but as a vehicle to stimulate, deepen and widen partnership between academia and industry in software analytics internationally. In the age of Big Data, this meeting also serves as the first step to plan

for the next decade of Big Data Analytics in Software Engineering, since it is impossible for individual groups or companies to tackle this challenging problem alone.

Software analytics is an ideal topic for this kind of interaction. It combines challenging research problems with real practical importance for the software industry, and the wider society that it serves. It presents an excellent and wide-ranging set of open research questions to academics concerning, amongst other things, analytic-algorithm design, data analysis, information visualization, scalable computing, software-artifact analysis and mining, social factors, empirical software engineering, measurement, process improvement, and technology transfer and adoption. Software analytics is also of critical practical significance to almost every organization involved in the production and use of software and services. Answers to the currently open research questions in software analytics can have a major impact upon industrial practice, with far-reaching implications for the development of the global economy. This combination of academic challenge and industrial relevance makes software analytics a natural topic for the proposed meeting.

In this meeting, we have brought together software-analytics researchers in academia and industry. Our main focus was to exploit the synergy of these communities and to provide a platform to forge new collaborations. Participants were invited to present a few plenary talks and demos of new tools, beside which the meeting has provided ample opportunities for small working groups on themes suggested by the participants. We expect this meeting has resulted in ample cross-fertilization between the different research areas and shown up exciting directions for improving software-engineering practices via practical software analytics.

Meeting Schedule

- Oct. 20th, Sunday Evening
  - Welcome Reception
- Oct. 21st, Monday Morning
  - Opening
  - Self Introduction
  - Short Presentations
- Oct. 21st, Monday Afternoon
  - Keynote: "MetaMorphosis", Michele Lanza
  - Short Presentations
- Oct. 22nd, Tuesday Morning
  - Short Presentations
  - Software Engineering Researches in Japan, Yasutaka Kamei
  - Software Engineering Researches in Brazil, Marco Gerosa
- Oct. 22nd, Tuesday Afternoon
  - Breakout Sessions
    - Impact to Practice
    - Performance
    - Dependability
    - Analytics for Continuous Delivery
- Oct. 23rd, Wednesday Morning
  - Keynote: "Software Provenance", Michael Godfrey
  - Presentations from Each Breakout Session
- Oct. 23rd, Wednesday Afternoon
  - Excursion and Dinner in Kamakura
- Oct. 24th, Thursday Morning
  - Breakout Sessions
    - Mining App Stores
    - Source Code Analysis
- Oct. 24th, Thursday Afternoon
  - Breakout Sessions (Continued)
  - Presentations from Each Breakout Session
  - Fishbowl Panel
- Oct. 25th, Friday Morning
  - Discussion on Definition and Future of Software Analytics
  - Wrap up

Overview of Talks / Position Statements

On Rapid Releases and Software Testing

Bram Adams, Polytechnique Montréal

Large open and closed source organizations like Google, Facebook and Mozilla are migrating their products towards rapid releases. While this allows faster time-to-market and user feedback, it also implies less time for testing and bug fixing. Since initial research results indeed show that rapid releases fix proportionally less reported bugs than traditional releases, we investigated the changes in software testing effort after moving to rapid releases. We analyzed the results of 312,502 execution runs of the 1,547 mostly manual system-level test cases of Mozilla Firefox from 2006 to 2012 (5 major traditional and 9 major rapid releases), and triangulated our findings with a Mozilla QA engineer. We found that in rapid releases, testing has a narrower scope that enables deeper investigation of the features and regressions with the highest risk, while traditional releases run the whole test suite. Furthermore, rapid releases make it more difficult to build a large testing community, forcing Mozilla to increase contractor resources in order to sustain testing for rapid releases.

Knowledge Engineering for Software Engineering

Yingnong Dang, Microsoft Research Asia

This talk briefed a few projects we conducted at the Software Analytics group of Microsoft Research Asia on software engineering, including code clone analysis, change understanding, and API usage mining. The synergy of these projects is extracting knowledge from large-scale codebase and help boosting software engineering productivity.

Using the Big Sky environment for software analytics (Demo)

Robert DeLine, Microsoft Research

Big Sky is a new integrated environment for large-scale data analysis being developed at Microsoft Research. Big Sky is a collaborative web service that allows data scientists to carry out entire workflows from raw data to final charts and plots. The central metaphor is an indelible research notebook: every action in Big Sky is immediately stored to preserve provenance and to allow repeatable analyses. Big Sky also provides automation and visualization at every step to keep the analyst productive and informed. I look forward to lots of feedback, since the workshop participants are exactly our intended users.

Linux and git: mining how a distributed version control system is used

Daniel M German, University of Victoria, Canada

Distributed Version Control systems (DVCs) are replacing centralized version systems (CVCs). Our research is trying to uncover the way that DVCs

are being used, and how its use differs from CVCs. For the last 22 months we have been mining Linux git repositories in an attempt to understand how they use their DVCs (git). I describe the challenges of this mining, and provide an overview of how linux uses DVCs.

Logical dependencies and others

Marco Aurelio Gerosa, University of São Paulo, Brazil

I have presented some studies that we have been conducting in our group, covering the following topics: a method for the identification of logical dependencies, characteristics of the automated tests x code quality, design degradation, change prediction, and refactoring. I have also presented a short demo of MetricMiner.

Availability of Modification Patterns for Identifying Maintenance Opportunities

Yoshiki Higo, Osaka University

In code repositories, there are multiple commits including the same modification, each of which we call modification pattern. This talk discussed availability of modification patterns for identifying maintenance opportunities such as performing refactorings or finding latent bugs.

What Makes a Green Miner?

Abram Hindle, University of Alberta

This talk discussed recent results and the infrastructure behind Green Mining on the Android platform.

Querying, Transforming, and Synchronizing Software Artifacts

Zhenjiang Hu, National Institute of Informatics

Bidirectional transformations provide a novel mechanism for synchronizing and maintaining the consistency of information between input and output. This talk shows how GRoundTram, a bidirectional graph transformation system, may be useful for querying, transforming, and synchronizing software artifacts in software development.

Automated Analysis of Load Testing Results

ZhenMing (Jack) Jiang, York University, Canada

Many software systems must be load tested to ensure that they can scale up under high load while maintaining functional and non-functional requirements.

Current industrial practices for checking the results of a load test remain ad-hoc, involving high-level manual checks. Few research efforts are devoted to the automated analysis of load testing results, mainly due to the limited access to large scale systems for use as case studies. Approaches for the automated and systematic analysis of load tests are needed, as many services are being offered online to an increasing number of users. I have talked about the general methodology that we have developed over the years to assess the quality of a system under load by mining the system behavior data (performance counters and execution logs).

Making Defects Prediction More Pragmatic

Yasutaka Kamei, Kyushu University, Japan

The majority of quality assurance research focused on defect prediction models that identify defect-prone modules (i.e., files or packages). Although such models can be useful in some contexts, they also have their drawbacks. I have presented some defect prediction studies that we have conducted.

Automated Performance Analysis of Build Systems

Shane McIntosh, Queen's University, Canada

Software developers rely on a fast and correct build system to compile their source code changes to produce modified deliverables for testing and deployment. Unfortunately, the scale and complexity of builds makes build performance analysis necessary, yet difficult due to the absence of build performance analysis tools. In this paper, we propose an approach that analyzes the build dependency graph and the change history of a software system to pinpoint build hotspots, i.e., source files that change frequently and take a long time to rebuild. In conducting a case study on the GLib, PostgreSQL, Qt, and Ruby systems, we observe that: (1) our approach identifies build hotspots that are more costly than the files that: rebuild the slowest, change the most frequently, or have the highest fan-in; (2) logistic regression models built using architectural and code properties of source files can explain 50%-75% of these build hotspots; and (3) build hotspots are more closely related to system architecture than to code properties. Furthermore, we identify build hotspot anti-patterns and offer advice on how to avoid and address them. Our approach helps developers to focus build performance optimization effort (e.g., refactoring) onto the files that will yield the most performance gain.

Disruptive Events on Software Projects

Peter C Rigby, Concordia University, Montréal

I discussed events that disrupt software projects, how we can measure these events, and how different projects mitigate the risk and damage associated with disruption.

Leveraging Performance Counters and Execution Logs to Diagnose Performance Issues

Mark D. Syer, Queen's University, Canada

Load tests ensure that software systems are able to perform under the expected workloads. The current state of load test analysis requires significant manual review of performance counters and execution logs, and a high degree of system-specific expertise. In particular, memory-related issues (e.g., memory leaks or spikes), which may degrade performance and cause crashes, are difficult to diagnose. Performance analysts must correlate hundreds of megabytes or gigabytes of performance counters (to understand resource usage) with execution logs (to understand system behavior). However, little work has been done to combine these two types of information to assist performance analysts in their diagnosis. In this talk, I have presented an approach that combines performance counters and execution logs to diagnose memory-related issues in load tests.

Software Text Analytics: Moving from Correlation Towards Causation

Tao Xie, University of Illinois at Urbana-Champaign, USA

In recent years, using deep natural language process (NLP) techniques to understand semantics of natural language (NL) software artifacts has emerged in the software engineering and security communities. Such movement is beyond what traditional text mining techniques, which typically treat NL sentences as a bag of words and then conduct statistical analysis on these words. This talk presents some recent research efforts that we have conducted in developing/applying NLP techniques for discovering semantic information out of NL software artifacts (https://sites.google.com/site/asergrp/projects/textse). These efforts hold great promises for moving from correlation towards causation, exploring the long-standing issue of "correlation does not imply causation", commonly faced in software analytics.

Active Support for Clone Refactoring

Norihiro Yoshida, Nara Institute of Science and Technology

Clone refactoring (merging duplicate code) is a promising solution to improve the maintainability of source code. This talk discussed research objectives and directions towards the advancement of clone refactoring from the perspective of active support.

Checking App Behavior Against App Descriptions

Andreas Zeller, Saarland University

How do we know a program does what it claims to do? After clustering mined Android apps by their description topics, we identify outliers in each cluster with respect to their API usage. A "weather" app that sends messages

thus becomes an anomaly; likewise, a "messaging" app would not be expected to access the current location. Applied on a set of 22,000+ Android applications, our approach identified several anomalies, and classified known malware accurately with high precision and recall.

145 Questions for Data Scientists in Software Engineering

Thomas Zimmermann, Microsoft Research

I have presented a catalog of 145 questions that software engineers would like to ask data scientists. The catalog was created based on feedback from 810 Microsoft employees. This is joint work with Andrew Begel.

List of Participants

- Bram Adams, Polytechnique Montréal
- Yingnong Dang, Microsoft Research Asia
- Robert Deline, Microsoft Research
- Daniel German, University of Victoria
- Marco Gerosa, University of São Paulo
- Michael Godfrey, University of Waterloo
- Ahmed Hassan, Queen's University
- Yoshiki Higo, Osaka University
- Abram Hindle, University of Alberta
- Zhenjiang Hu, National Institute of Informatics
- Akinori Ihara, Nara Institute of Science and Technology
- Katsuro Inoue, Osaka University
- Zhenming Jiang, York University
- Yasutaka Kamei, Kyushu University
- Sung Kim, Hong Kong University of Science and Technology
- Takashi Kobayashi, Tokyo Institute of Technology
- Michele Lanza, University of Lugano
- Andrian Marcus, Wayne State University
- Shane McIntosh, Queen's University
- Akito Monden, Nara Institute of Science and Technology
- Masao Ohira, Wakayama University
- Peter Rigby, Concordia University
- Mark Syer, Queen's University
- Tao Xie, University of Illinois at Urbana-Champaign
- Norihiro Yoshida, Nara Institute of Science and Technology
- Andreas Zeller, Saarland University
- Dongmei Zhang, Microsoft Research Asia
- Thomas Zimmermann, Microsoft Research