Report on the 6th International Workshop on Quantitative Approaches to Software Quality (QuASoQ 2018)

Horst Lichter
RWTH Aachen University
Germany
lichter@swc.rwth-aachen.de

Thanwadee Sunetnanta
Mahidol University
Thailand
thanwadee.sun@mahidol.ac.th

Toni Anwar
Universiti Teknologi PETRONAS
Malaysia
toni.anwar@utp.edu.my

Taratip Suwannasart
Chulalongkorn University
Thailand
taratip.s@chula.ac.th

I. Introduction

Based on the feedback and experience we got from the 5th workshop we slightly adjusted the list of topics for the workshop and planned to invite a keynote speaker again. The topics of interest included

- New approaches to measurement, evaluation, comparison and improvement of software quality
- Metrics and quantitative approaches in agile projects
- Case studies and industrial experience reports on successful or failed application of quantitative approaches to software quality
- Tools, infrastructure and environments supporting quantitative approaches
- Empirical studies, evaluation and comparison of measurement techniques and models
- Quantitative approaches to test process improvement, test strategies or testability
- Empirical evaluations or comparisons of testing techniques in industrial settings

Overall, the workshop aimed at gathering together researchers and practitioners to discuss experiences in the application of state of the art approaches to measure, assess and evaluate the quality of both software systems as well as software development processes in general and software test processes in particular.

As software development organizations are always forced to develop software in the "right" quality, the quality specification and quality assurance are crucial. Although there are lots of approaches to deal with quantitative quality aspects, it is still challenging to choose a suitable set of techniques that best fit to the specific project and organizational constraints.

Even though approaches, methods, and techniques are known for quite some time now, little effort has been spent on the exchange on the real-world problems with quantitative approaches. For example, only limited research has been devoted to empirically evaluate risks, efficiency or limitations of different testing techniques in industrial settings.

Hence, one main goal of the workshop was to exchange experience, present new promising approaches and to discuss how to set up, organize, and maintain quantitative approaches to software quality.

II. Workshop Format

Based on our former experience we wanted the workshop to be highly interactive. In order to have an interesting and interactive event sharing lots of experience, we organized the workshop presentations applying the author-discussant model.

Based on this workshop model, papers are presented by one of the authors. After the presentation, a discussant starts the discussion based on his or her pre-formulated questions. Therefore, the discussant had to prepare a set of questions and had to know the details of the presented paper. The general structure of each talk was as follows:

- The author of a paper presented the paper (20 minutes).
- After that, the discussant of the paper opened the discussion using his or her questions (5 minutes).
- Finally, we moderated the discussion among the whole audience (5 minutes).

III. Invited Talk

This year we were happy to have Prof. Hongyu Zhang as our invited speaker. Hongyu Zhang is currently an Associate Professor at The University of Newcastle, Australia. Previously, he was a Lead Researcher at Microsoft Research Asia and an Associate Professor at Tsinghua University, China. He received his PhD degree from National University of Singapore in 2003. His research is in the area of Software Engineering, in particular, software analytics, testing, maintenance, and reuse. The main theme of his research is to improve software quality and productivity by mining software data. He has published more than 120 research papers in international journals and conferences, including TSE, TOSEM, ICSE, FSE, POPL, AAAI, KDD, IJCAI, ASE, ISSTA, ICSM, ICDM, and USENIX. He received two ACM Distinguished Paper awards.

He also served as a program chair and committee member for many software engineering conferences. He is on the Editorial Board of Journal of Systems and Software, and is a Senior Member of IEEE.

Prof. Hironori Washizaki presented in his talk entitled "*Intelligent Fault Diagnosis and Prediction through Data Analytics*" important insights how the analysis of big data can support the prediction of faults in systems to support managers taking the right decisions before releasing a software system.

IV. Workshop Contributions

Altogether ten papers were submitted. Finally, nine papers h accepted by the program committee for presentation and publication covering very different topics. We grouped the papers into three sessions and added a final round-up slot to present and discuss the major findings of our workshop. In the following we want to give a short overview of the accepted papers.

A. Yeongjun Cho, Jung-Hyun Kwon, In-Young Ko*: Cross-Sub-Project Just-in-Time Defect Prediction on Multi-Repo Projects*

Just-in-time (JIT) defect prediction, which predicts defect-inducing code changes, can provide faster and more precise feedback to developers than traditional module-level defect prediction methods. We find that large-scale projects such as Google Android and Apache Maven divide their projects into multiple sub-projects, in which relevant source code is managed separately in different repositories. Although sub-projects tend to suffer from a lack of the historical data required to build a defect prediction model, the feasibility of applying cross-subproject JIT defect prediction has not yet been studied. A cross-sub- project model to predict bug-inducing commits in the target sub-project could be built with data from all other sub-projects within the project of the target sub-project, or data from the subprojects of other projects, as traditional project-level JIT defect prediction methods. Alternatively, we can rank sub-projects and select high-ranked sub-projects within the project to build a filtered-within-project model. In this work, we define a subproject similarity measure based on the number of developers who have contributed to both sub-projects to rank sub-projects. We extract the commit data from 232 sub-projects across five different projects and evaluate the cost effectiveness of various cross-sub-project JIT defect prediction models. Based on the results of the experiments, we conclude that 1) cross-sub-project JIT defect prediction generally has better cost effectiveness than within-sub-project JIT defect prediction, especially when the sub-projects from the same project are used as training data; 2) in filtered-within-project JIT defect-prediction models, the developer similarity-based ranking can achieve higher cost effectiveness than the other ranking methods; and 3) although a developer similarity-based filtered-within-project model achieves lower cost effectiveness than a within-project model in general, we find that there is room for further improvement to the filtered-within-project model that may outperform the within-project model..

B. Chao Zhang, Weiliang Yin and Zhiqiang Lin*: Boost Symbolic Execution Using Dynamic State Merging and Forking*

Symbolic execution has achieved wide application in software testing and analysis. However, path explosion remains the bottleneck limiting scalability of most symbolic execution engines in practice. One of the promising solutions to address this issue is to merge explored states and decrease number of paths. Nevertheless, state merging leads to increase in complexity of path predicates at the same time, especially in the situation where variables with concrete values are turned symbolic and chances of concretely executing some statements are dissipated. As a result, calculating expressions and constraints becomes much more time consuming and thus, the performance of symbolic execution is weakened in contrast. To resolve the problem, we propose a merge-fork framework enabling states under exploration to switch automatically between merging mode and forking mode. First, active state forking is introduced to enable forking a state into multiple ones as if a certain merging action taken before were eliminated. Second, we perform dynamic merge fork analysis to cut source code into pieces and continuously evaluate efficiency of different merging strategies for each piece. Our approach dynamically combines paths under exploration to maximize opportunities for concrete execution and ease the burden on underlying solvers. We implement the framework on the foundation of the symbolic execution engine KLEE, and conduct experiments on GNU Core utils code using our prototype to present the effect of our proposition. Experiments show up to 30% speedup and 80% decrease in queries compared to existing works.

C. Konrad Fögen and Horst Lichter*: A Case Study on Robustness Fault Characteristics for Combinatorial Testing - Results and Challenges*

Combinatorial testing is a well-known black-box testing approach. Empirical studies suggest the effectiveness of combinatorial coverage criteria. So far, the research focuses on positive test scenarios. But, robustness is an important characteristic of software systems and testing negative scenarios is crucial. Combinatorial strategies are extended to generate invalid test inputs but the effectiveness of negative test scenarios is yet unclear. Therefore, we conduct a case study and analyze 434 failures reported as bugs of an financial enterprise application. As a result, 51 robustness failures are identified including failures triggered by invalid value combinations and failures triggered by interactions of valid and invalid values. Based on the findings, four challenges for combinatorial robustness testing are derived.

D. Séverine Sentilles, Efi Papatheocharous and Federico Ciccozzi*: What do we know about software security evaluation? A preliminary study*

In software development, software quality is nowadays acknowledged to be as important as software functionality and there exists an extensive body-of-knowledge on the topic. Yet, software quality is still marginalized in practice: there is no consensus on what software quality exactly is, how it is achieved and evaluated. This work investigates the state-of-the-art of software quality by focusing on the description of evaluation methods for a subset of software qualities, namely those related to software security. The main finding of this paper is the lack of information regarding fundamental aspects that ought to be specified in an evaluation method description. This work follows up the authors' previous work on the Property Model Ontology by carrying out a systematic investigation of the state-of-the-art on evaluation methods for software security. Results show that only 25% of the papers studied provide enough information on the security evaluation methods they use in their validation processes, whereas the rest of the papers lack important information about various aspects of the methods (e.g., benchmarking and comparison to other properties, parameters, applicability criteria, assumptions and available implementations). This is a major hinder to their further use.

E. Maohua Gan, Kentaro Sasaki, Akito Monden and Zeynep Yucel*: Generation of Mimic Software Project Data Setsfor Software Engineering Research*

To conduct empirical research on industry software development, it is necessary to obtain data of real software projects from industry. However, only few such industry data sets are publicly available; and unfortunately, most of them are very old. In addition, most of today's software companies cannot make their data open, because software development involves many stakeholders, and thus, its data confidentiality must be strongly preserved. This paper proposes a method to artificially generate a "mimic" software project data set whose characteristics (such as average, standard deviation and correlation coefficients) are very similar to a given confidential data set. The proposed method uses the Box–Muller method for generating normally distributed random numbers, then, exponential transformation and number reordering are used for data mimicry. Instead of using the original (confidential) data set, researchers are expected to use the mimic data set to produce similar results as the original data set. To evaluate the usefulness of the proposed method, effort estimation models were built from an industry data set and its mimic data set. We confirmed that two models are very similar to each other, which suggests the usefulness of our proposal.

F. Nayla Nasir and Nasir Mehmood Minhas*: Implementing Value Stream Mapping in a Scrum-based project - An Experience Report*

The value stream mapping is one of the lean practices, that helps to visualize the whole process and identifies any bottlenecks affecting the flow. Proper management of the value stream can significantly contribute towards waste elimination by categorizing process activities to be either value adding or non value-adding. Lean development focuses on the value through the elimination of waste. Adding value through embracing change and customer satisfaction are also the benefits of Scrum. This study reports our experience regarding the implementation of VSM with Scrum. We followed the action research method, with an objective to see if VSM can contribute to the identification and reduction of wastes in a Scrum-based project. We identified a noticeable amount of waste even with strict compliance to the Scrum practices. On the basis of identified waste, their root causes, and possible mitigation strategy we have proposed a future state map, that could help improve the productivity of the process. The results of our study are encouraging, and we suggest that adoption of VSM with Scrum could add more value to the Scrum-based projects.

G. Ankush Dadwal, Hironori Washizaki, Yoshiaki Fukazawa, Takahiro Iida, Masashi Mizoguchi and Kentaro Yoshimura*: Prioritization in Automotive Software Testing: Systematic Literature Review*

Automotive Software Testing is a vital part of the automotive systems development process. Not identifying the critical safety issues and failures of such systems can have serious or even fatal consequences. As the number of embedded systems and technologies increases, testing all components becomes more challenging. Although testing is expensive, it is important to reduce bugs in an early stage to maintain safety and to avoid recalls. Hence, the testing time should be reduced without impacting the reliability. Several studies and surveys have prioritized Automotive Software Testing to increase its effectiveness. The main goals of this study are to identify: (i) the publication trends of prioritization in Automotive Software Testing, (ii) which methods are used to prioritize Automotive Software Testing, (iii) the distribution of studies based on the quality evaluation, and (iv) how existing research on prioritization helps optimize Automotive Software Testing.

H. Reishi Yokomori, Norihiro Yoshida, Masami Noro and Katsuro Inoue: Use-Relationship Based Classification for Software Components

In recent years, the maintenance period of the software system is increasing. The size of the software system has grown, and the number of classes and the relationship between classes are also increasingly complicated. If we can categorize software components based on information such as functions and roles, we believe that these classified components can be understood together, and are useful for understanding the system. In this paper, we proposed a classification method for software components based on similarity of use relation. For each component, a set of components used by the component was analyzed. And then, for each pair of components, the distance was calculated from the coincidence of the two sets. A distance matrix was created and components were classified by hierarchical cluster analysis. We applied this method to jlGui consisting of 70 components. 8 clusters of 36 components were extracted from the 70 components. Characteristics of the extracted clusters were evaluated, and the content of each cluster was introduced as a case study. In 7 clusters out of the 8 clusters, components of the cluster were strongly similar with each other from the viewpoint of their functions. Through these experiments, we confirmed that our method is effective for classifying components of the target software, and is useful for understanding them.

V. Summary of the Discussions

About 30 researchers attended the workshop and participated in the discussions. The author-discussant model was well received by the participants and led to intensive discussions among them.

For instance, the discussion of paper D *(Séverine Sentilles et al)* focused on issues regarding the categorization of existing literatures on their evaluation methods related to software security into three groups based on their main purpose. The first group focuses on defining a new property or metric. The second group bases their work on already defined properties. Finally, the last group of the literatures are not explicitly referring to any property, method or metric.

Another example, the discussion on paper G *(Ankush Dadwal et al.)* focused on issues why automotive software testing was chosen rather than software testing on other applications. The testing methods presented were so debatable among the discussants and the attendants.

The last discussion of the workshop was about classification methods for software components based on similarity of use relation *(Reishi Yokomori et al.)*. The attendants had some issues on similarity of use relation. The similarity of software components can be calculated using silhouette coefficient. However, this led to interesting discussions how similar software components are clustered.

To conclude, in the course of this workshop the participants proposed and discussed different approaches to quantify relevant aspects of software development. Especially the discussions led to new ideas, insights, and take-aways for all participants.

VI. Acknowledgments

Many people contributed to the success of this workshop. First, we want to give thanks to our invited speaker and the authors and presenters of the accepted papers. Furthermore, we want to express our gratitude to the APSEC 2018 organizers; they did a perfect job. Finally, we are glad that these people served on the program committee (some of them for many years) and supported the workshop by soliciting papers and by writing peer reviews:

- Matthias Vianden, Aspera GmbH, Aachen, Germany
- Wan M.N. Wan Kadir, UTM Johor Bahru, Malaysia
- Maria Spichkova, RMIT University, Melbourne, Australia
- Tachanun Kangwantrakool, ISEM, Thailand
- Jinhua Li, Qingdao University, China
- Apinporn Methawachananont, NECTEC, Thailand
- Nasir Mehmood Minhas, BTH Karlskrona, Sweden
- Chayakorn Piyabunditkul, NSTDA, Thailand
- Sansiri Tanachutiwat, Thai German Graduate School of Engineering, TGGS, Thailand
- Hironori Washizaki, Waseda University, Japan
- Hongyu Zhang, University of Newcastle, Australia
- Minxue Pan, Nanjing University, China

Finally, the QuASoQ organizers would like to express their deepest gratitude to our colleague Ashish Sureka, who passed away early this year, for his continuous support and valuable contributions.