Report on the 5th International Workshop on Quantitative Approaches to Software Quality (QuASoQ 2017)

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

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

Toni Anwar
UTM Johor Bahru
Malaysia
tonianwar@utm.my

I. Introduction

After the successful workshop QuASoQ 2016, which was held in Hamiltion, New Zealand, the organizers of the 4th workshop wanted to widen the scope of quantitative approaches to software quality. Therefore, the call for papers and the list of topics of the workshop were adjusted in the direction of quantitative approaches in software testing. 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 (15 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 (10 minutes).

III. Invited Talk

This year we were happy to have Prof. Hironori Washizaki as an invited speaker. Hironori Washizaki is the Director and a Professor with the Global Software Engineering Laboratory, Waseda University, Japan. He is also a Visiting Professor with the National Institute of Informatics, and an Outside Director with the SYSTEM INFORMATION CO., LTD. He was a Visiting Professor with the Ecole Polytechnique de Montreal, in 2015. He has long-term experience of researching and practicing software design, reuse, quality assurance, and education.

Prof. Hironori Washizaki presented in his talk entitled “Pitfalls and Countermeasures in Software Quality Measurements and Evaluations” important aspects that are influencing the application of quality measurements in the context of software development. He identified a set of pitfalls and presented respective countermeasures. Essentially, appropriate goals and strategies have to be defined and linked together to make a measurement program successful.

IV. Workshop Contributions

Altogether twelve papers were submitted. Finally, ten papers were 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. Lov Kumar, Santanu Rath and Ashish Sureka: *Estimating Web Service Quality of Service Parameters using Source Code Metrics and LSSVM*

We conduct an empirical analysis to investigate the relationship between thirty-seven different source code metrics with fifteen different Web Service QoS (Quality of Service) parameters. The source code metrics used in our experiments consists of nineteen Object-Oriented metrics, six Baski and Misra metrics, and twelve Harry M. Sneed metrics. We apply Principal Component Analysis (PCA) and Rough Set Analysis for feature extraction and selection. The different sets of metrics are provided as input to the predictive model generated using Least Square Support Vector Machine (LSSVM) with three different types of kernel functions: RBF, Polynomial, and Linear. Our experimental results reveal that the prediction model developed using LSSVM method with RBF kernel function is more effective and accurate for prediction of QoS parameters than the LSSVM method with linear and polynomial kernel functions. Furthermore, we also observe that the predictive model created using object-oriented metrics achieves better results in comparison to other sets of source code metrics.

B. Sandhya Tarwani and Ashish Sureka: *Investigating the Effectiveness of Greedy Algorithm on Open Source Software Systems for Determining Refactoring Sequence*

The deeper problem in the source code are the bad smells that indicates something is wrong and if they are not detected timely, then they lead towards the complete deterioration of the working software causing major financial and productivity loss. Refactoring helps in removing these bad smells by improving internal quality attributes of the software without affecting its external behaviour. However refactoring needs to be applied in a controlled manner. In this study an approach has been propose for determining an optimal refactoring sequence that will maximize the source-code maintainability using greedy algorithm. The proposed approach selects the most optimum sequence at every step-in hope of finding the global optimum solution. We conduct an empirical analysis on four open-source software and select those classes that have bad smells greater than or equal to four. Further filtration is done by selecting those classes from the group that have high value of source code lines. We demonstrate the effectiveness of our approach using concrete examples of the experimental dataset and presenting summary results.

C. Jan Thomas, Ana Nicolaescu and Horst Lichter*: Static and Dynamic Architecture Conformance Checking: A Systematic, Case Study-Based Analysis on Tradeoffs and Synergies*

In order to uncover architectural drift, a plethora of architecture conformance checking tools has been proposed that mainly leverage two approaches: they extract architectural knowledge based on either source code artifacts (static approach) or run-time behavior (dynamic approach). Although both approaches have been evaluated separately, no up-to-date analysis of their relative strengths and weaknesses, nor real-world comparative case studies of the two were published. In this paper we address this issue by presenting the results of a direct comparison of both approaches. We first identify and compare their strengths and weaknesses on a theoretical level. We then evaluate these results against our experiences gained in a large-scale industrial case study. As a result, we argue that the approaches cannot substitute each other as they differ in many key aspects. Hence, we crystallize guidelines regarding how to combine these such that their strengths are emphasized while weaknesses mitigated.

D. Abdus Satter, Nadia Nahar and Kazi Sakib*: Automatically Identifying Dead Fields in Test Code by Resolving Method Call and Field Dependency*

Dead fields are the unused setup fields in the test code which reduce the comprehensibility and maintainability of a software system. A test class contains dead fields when developers initialize setup fields without analyzing the usage of fields properly. Manually identifying dead fields to remove from the code is a time consuming and error-prone task. In this paper, a technique named Dead Field Detector (DFD) has been proposed to detect dead fields automatically. The technique constructs Call Graph (CG) and Data Dependence Graph (DDG) from test code to find method invocation and field dependency relationships, respectively. It identifies the fields initialized in the setup method and its invoked methods from CG. It finds setup fields by collecting the initialized fields and their dependent fields from DG. To determine the usage of the setup fields, it checks the bodies of the test methods and their invoked methods obtained from CG. All the unused setup fields are separated from the used fields and considered as dead fields. In order to evaluate DFD, an open source project named eGit was used. The result analysis shows that DFD has identified 14.03% more setup fields and 60.98% more dead fields than an existing technique named TestHound for eGit.

E. Yuichiro Senzaki, Siyuan Liu, Hironori Washizaki, Yoshiaki Fukazawa, Hiroshi Kobayashi and Masaharu Adachi*: A Web Application to Manage and Improve Software Development Projects by SEMAT Essence*

As part of the rapid advances in software engineering, each year a vast amount of new knowledge and ideas are proposed. However, a gap often arises between new ideas and current methods due to a lack of fundamental theory. To bridge this gap, SEMAT (Software Engineering Methods and Theory) Essence has been proposed as the common ground in software engineering. Using SEMAT Essence, developers can track the progress and health of a project more efficiently from various viewpoints. However, SEMAT Essence has some limitations. In practice, only a few tools implement SEMAT Essence. Most of these tools are problematic and do not sufficiently satisfy the requirements for practical developments. Therefore, we develop a tool called OCMS (Online Checklist Management System), which improves existing tools. An experiment where students manage an ET robot contest project using OCMS confirms its effectiveness and demonstrates that OCMS can help developers improve efficiency.

F. Felix Timm, Simon Hacks, Felix Thiede and Daniel Hintzpeter*: Towards a Quality Framework for Enterprise Architecture Models*

While Enterprise Architecture Management is an established and widely discussed field of interest in the context of information systems research, we identify a lack of work regarding quality assessment of enterprise architecture models in general and frameworks or methods on that account in particular. By analyzing related work by dint of a literature review in a design science research setting, we provide twofold contributions. We (i) suggest an Enterprise Architecture Model Quality Framework (EAQF) and (ii) apply it to a real world scenario.

G. Ke Dai and Philippe Kruchten*: Detecting Technical Debt through Issue Trackers*

Managing technical debt effectively to prevent it from accumulating too quickly is of great concern to software stakeholders. To pay off technical debt regularly, software developers must be conscious of the existence of technical debt items. The first step is to make technical debt explicit; that is the identification of technical debt. Although there exist many kinds of static source code analysis tools to identify code-level technical debt, identifying non-code-level technical debt is very challenging and needs deep exploration. This paper proposed an approach to identifying non-code-level technical debt through issue tracking data sets using natural language processing and machine learning techniques and validated the feasibility and performance of this approach using an issue tracking data set recorded in Chinese from a commercial software project. We found that there are actually some common words that can be used as indicators of technical debt. Based on these key words, we achieved the precision of 0.72 and the recall of 0.81 for identifying technical debt items using machine learning techniques respectively.

H. Suppasit Roongsangjan, Thanwadee Sunetnanta and Pattanasak Mongkolwat*: Multi-Level Compliance Measurements for Software Process Appraisal*

Software process appraisal is to assess whether an implemented software process complies with a process reference model. To conduct the appraisal, the appraisal team will request an organization to provide objective evidence reflecting practice implementation. Then such evidence will be examined, verified, and validated to generate appraisal results. This evidence collection process is done after a process is implemented. To better prepare for software process appraisal, we argued that the compliance of a process can be measured prior to its implementation. In the light of that, we proposed multi-level compliance measurements to determine process reference model compliance, in terms of Process Model Readiness Score, Process Enactment Score, and Process Implementation Readiness Score. These measurements help provide an insight analysis of where the problems of practice implementation lie, i.e. at process modeling, at process enactment, or at process implementation.

I. Tachanun Kangwantrakool and Thanaruk Theeramunkong: *Towards the Re-engineering of Readiness Review Process with R2P2 Lifecycle Model*

As a lesson learned, the readiness review process of SCAMPI appraisal is very complicated and effort and cost consuming for novice organizations who need to know their status of CMMI practices classification. In SCAMPI appraisal, the readiness review process is a single process that runs from start to the end. During the readiness review process, nobody collected database to improve the process's performance. Our research towards the re-engineering of readiness review process aims to enhance the process performance and reduce effort/cost of the readiness review process implementation. We design our R2P2 Lifecycle Model to provide the benchmark of readiness review process relative to Capability Maturity Model Integration (CMMI) practices implementation. The R2P2 Lifecycle Model describes the requirements, activities, and practices associated with the readiness review process that composes the model. The total of 49 appraisals data are used to establish, evaluate, and enhance the performance and efficiency of the R2P2 Lifecycle Model. This paper presents our conceptual view of the R2P2 Lifecycle Model and a lesson learned from the preliminary development of the model. We are at the first stage of the R2P2 Lifecycle Model development by using 30 historical appraisals cases for root causes analyzing of the weakness of readiness review process. Therefore, we will use this lesson learned to enhance the R2P2 Lifecycle Model in next stage.

V. Summary of the Discussions

About 25 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.

To conclude, in the course of this workshop the participants proposed and discussed different approaches to measure and quantify relevant aspects of software development. For Especially the discussions led to constructive feedback, deeper insights, and hopefully some 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 2017 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
- Taratip Suwannasart, Chulalongkorn Univiversity, Thailand
- Tachanun Kangwantrakool, ISEM, Thailand