Report on the 4th International Workshop on Quantitative Approaches to Software Quality (QuASoQ 2016)

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

Konrad Fögen
RWTH Aachen University
Germany
foegen@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 2015, which was held in New Delhi, India, 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).

Again, this format was very successful as it led to more intensive discussions among the participants.

III. Workshop Contributions

Altogether nine papers were submitted. Finally, seven 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. Hirohisa Aman, Sousuke Amasaki, Tomoyuki Yokogawa and Minoru Kawahara: *Local Variables with Compound Names and Comments as Signs of Fault-Prone Java Methods*

This paper focuses on local variables and comments in methods of Java applications. Both of them are usually used at the programmer's discretion. Thus, naming local variables and commenting code can vary among individuals, and such an individual difference may cause a dispersion in quality.

The authors conducted an empirical analysis on the fault-proneness of Java methods which are collected from nine popular open source products. The results report the following three findings: (1) Methods having local variables with compound names are more likely to be faulty than the others; (2) Methods having local variables with simple and short names are unlikely to be faulty, but their positive effects tend to be decayed as their scopes get wider; (3) The presence of comments within a method body can also be useful sign of fault-prone method.

B. Ahmed Alharthi, Maria Spichkova and Margaret Hamilton: *Sustainability Profiling of Long-living Software Systems*

In this paper the authors introduce a framework for software sustainability profiling. The goal of the framework is to analyse sustainability requirements for long-living software systems, focusing on usability and readability of the sustainability profiles. To achieve this goal, the authors applied a quantitate approach such as fuzzy rating scale-based questionnaires to rank the sustainability requirements, and the Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) to analyse the results of questionnaires and to provide a basis for system profiling.

The core profiling elements provided by our framework are (1) a sustainability five-star rating, (2) visualisation of the five sustainability dimensions as a pentagon graph detailing combination for individual, social, technical, economic and environmental dimensions, and (3) a bar graph of overall sustainability level for each requirement. To ensure sustainability, the proposed profiling framework covers the five dimensions of sustainability to quantify the sustainability of any software system not only during the requirement gathering phase but also during maintenance phase of software system lifecycle.

C. Richa Awasthy, Shayne Flint and Ramesh Sankaranarayana: *Towards improved Adoption: Effectiveness of Research Tools in Real World*

One of the challenges in the area of software engineering research has been the low rate of adoption by industry of the tools and methods produced by university researchers. In this paper the authors present a model to improve the situation by providing tangible evidence that demonstrates the real-world effectiveness of such tools and methods. A survey of practising software engineers indicates that the approach in the model is valid and applicable. The authors applied and tested the model for providing such evidence and demonstrated its effectiveness in the context of static analysis using FindBugs. This model can be used to analyse the effectiveness of academic research contributions to industry and contribute towards improving their adoption.

D. Lov Kumar, Santanu Rath and Ashish Sureka: *Predicting Quality of Service (QoS) Parameters using Extreme Learning Machines with Various Kernel Methods*

Web services which are language and platform independent self-contained web-based distributed application components represented by their interfaces can have different Quality of Service (QoS) characteristics such as performance, reliability and scalability. One of the major objectives of a web service provider and implementer is to be able to estimate and improve the QoS parameters of their web service as its clients application are dependent on the overall quality of the service.

In this paper the authors hypothesized that the QoS parameters have a correlation with several source code metrics and hence can be estimated by analyzing the source code. They investigated the predictive power of 37 different software metrics to estimate 15 QoS attributes. Furthermore, they developed QoS prediction models using Extreme Learning Machines (ELM) with various kernel methods. Since the performance of the classifiers depends on the software metrics that are used to build the prediction model, the authors also examined two different feature selection techniques i.e., Principal Component Analysis (PCA), and Rough Set Analysis (RSA) for dimensionality reduction and removing irrelevant features. The performance of QoS prediction models are compared using three different types of performance parameters i.e., MAE, MMRE, RMSE. The obtained experimental results demonstrate that the model developed by extreme learning machine with RBF kernel achieves better results as compared to the other models in terms of the predictive accuracy.

E. Abdus Satter and Kazi Sakib: *Improving Recall in Code Search by Indexing Similar Codes under Proper Terms*

The recall of a code search engine is reduced, if feature-wise similar code fragments are not indexed under common terms. In this paper, a technique named Similarity Based Method Finder (SBMF) is proposed to alleviate this problem. The technique extracts all the methods from a source code corpus and converts these into reusable methods (i.e., program slice) through resolving data dependency. Later, it finds similar methods by checking signature (i.e., input and output types) and executing methods for a randomly generated set of input values. Methods are considered as feature-wise similar if these produce the same output set. In order to index these methods against common and proper terms, SBMF selects the terms that are found in most of the methods. Finally, query expansion is performed before searching the index to solve the vocabulary mismatch problem.

In order to evaluate SBMF, fifty open source projects implementing nine different functionalities or features were used. The results were compared with two types of techniques - Keyword BasedCode Search (KBCS) and Interface Driven Code Search (IDCS). On an average, SBMF retrieves 38% and 58% more relevant methods than KBCS and IDCS, respectively. Moreover, it is successful for all the features by retrieving at least one relevant method representing each feature whereas IDCS and KBCS are successful for 3 and 7 features out of 9 respectively.

F. Eun-Hye Choi, Osamu Mizuno and Yifan Hu: *Code Coverage Analysis of Combinatorial Testing*

Combinatorial t-way testing with small t is known as an efficient black-box testing technique to detect parameter interaction failures. So far, several empirical studies have reported the effectiveness of t-way testing on fault detection abilities. However, few studies have investigated the effectiveness of t-way testing on code coverage, which is one of the most important coverage criteria widely used for software testing.

This paper presents a quantitative analysis to evaluate the code-coverage effectiveness of t-way testing. Using three open source utility programs, the authors compared t-way testing with exhaustive (all combination) testing w. r. t. code coverage and test suite sizes.

G. Lucas Gren and Alfredo Goldman*: Trying to Increase the Mature Use of Agile Practices by Group Development Psychology Training - An Experiment*

There has been some evidence that agility is connected to the group maturity of software development teams. This study aims at conducting group development psychology training with student teams, participating in a project course at university, and compare their group effectiveness score to their agility usage over time in a longitudinal design. Seven XP student teams were measured twice (43+40), which means 83 data points divided into two groups (an experimental group and one control group).

The results showed that the agility measurement was not possible to increase by giving a 1.5-hour of group psychology lecture and discussion over a two-month period. The non-significant result was probably due to the fact that 1.5 hours of training were not enough to change the work methods of these student teams, or, a causal relationship does not exist between the two concepts. A third option could be that the experiential setting of real teams, even at a university, has many more variables not taken into account in this experiment that affect the two concepts. The authors therefore had no conclusions to draw based on the expected effects. However, they believed these concepts have to be connected since agile software development is based on teamwork to a large extent, but there are probable many more confounding or mediating factors.

IV. Summary of the Discussions

In total 10 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 A *(Aman et al.)* about compound names and comments as signs for faults has encountered great interest among the audience as many of them reported similar experiences. The discussion also led to new ideas as it was revealed that there may also be cultural aspects worth to be considered in future work. As an example, styleguides of a company which enforce certain conventions for variable names and comments or even the tongue in which the source code is written as some tongues may have an impact on the use of compound naming or comments.

As another example, the discussion of paper C *(Awasthy et al.)* focused on issues regarding the adoption of research tools in practice. It was pointed out that there is a mismatch between the researcher's focus when developing a tool and the practitioner's expectations when actually considering to use that tool. As a result, the proposed model could be adjusted to obtain feedback by the practitioner's earlier and more frequently.

The last discussion of the workshop was about an empirical study *(Gren & Goldman)* regarding a connection between agility and the maturity of a group of software developer's. The findings of this research were negative and thus no connection could be approved or disproved. However, this led to interesting discussions about the reasons and about any other yet unconsidered factors which might be involved.

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.

V. Acknowledgments

Many people contributed to the success of this workshop. First of all, we want to give thanks to the authors and presenters of the accepted papers. Furthermore, we want to express our gratitude to the APSEC 2016 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
- Jinhua Li, Qingdao University, China
- Apinporn Methawachananont, NECTEC, Thailand
- Jarernsri L. Mitrpanont, Mahidol University, Thailand
- Nasir Mehmood Minhas, PMAS - AAUR Rawalpindi Pakistan
- Chayakorn Piyabunditkul, NSTDA, Thailand
- Sansiri Tanachutiwat, Thai German Graduate School of Engineering, TGGS, Thailand
- Hironori Washizaki, Waseda University, Japan
- Hongyu Zhang, Microsoft Research, China