MALTESQUE 2019 Workshop Summary

Arcelli Fontana, Francesca; Perrouin, Gilles; Ampatzoglou, Apostolos; Archer, Mathieu; Walter, Bartosz; Cordy, Maxime; Palomba, Fabio; Devroey, Xavier

DOI
10.1145/3375572.3375582

Publication date
2020

Document Version
Submitted manuscript

Published in
Software Engineering Notes

Citation (APA)

Important note
To cite this publication, please use the final published version (if applicable). Please check the document version above.
MALUTESQUE 2019 Workshop Summary

Francesca Arcelli Fontana  
University of Milano-Bicocca  
Milan, Italy  
francesca.arcelli@unimib.it

Gilles Perrouin*  
University of Namur  
Namur, Belgium  
gilles.perrouin@unamur.be

Apostolos Ampatzoglou  
University of Macedonia  
Thessaloniki, Greece  
apostolos.ampatzoglou@gmail.com

Mathieu Acher  
University of Rennes 1, IRISA/INRIA Rennes, France  
mathieu.acher@irisa.fr

Bartosz Walter  
Poznań University of Technology  
Poznań, Poland  
Bartosz.Walter@cs.put.poznan.pl

Maxime Cordy  
University of Luxembourg  
Luxembourg  
maxime.cordy@uni.lu

Fabio Palomba  
University of Zurich  
Zurich, Switzerland  
apalomba@ifi.uzh.ch

Xavier Devroey  
Delft University of Technology  
Delft, The Netherlands  
X.D.M.Devroey@tudelft.nl

ABSTRACT
Welcome to the third edition of the workshop on Machine Learning Techniques for Software Quality Evaluation (MaLTeSQuE 2019), held in Tallinn, Estonia, August 27th, 2019, co-located with ESEC / FSE 2019. This year MALTESQUE merged with the MASES (Machine Learning and Software Engineering in Symbiosis) workshop, co-located with the ASE 2018 conference. Ten papers from all over the world were submitted, seven of them were accepted. The program also featured a keynote by Lionel Briand on the use of machine learning to improve software testing.

1. INTRODUCTION
The assessment of software quality is one of the most multifaceted (e.g., structural quality, product quality, process quality, etc.) and subjective aspects of software engineering (since in many cases it is substantially based on expert judgment). Such assessments can be performed at almost all phases of software development (from project inception to maintenance) and at different levels of granularity (from source code to architecture). However, human judgment is: (a) inherently biased by implicit, subjective criteria applied in the evaluation process, and (b) its economical effectiveness is limited compared to automated or semi-automated approaches. To this end, researchers are still looking for new, more effective methods of assessing various qualitative characteristics of software systems and the related processes.

In the recent years, we observed a rising interest in adopting various approaches to exploit machine learning (ML) and automated decision-making processes in several areas of software engineering. These models and algorithms help to alleviate human subjectivity in order to make informed decisions based on available data and evaluated with objective criteria. Thus, the adoption of ML techniques is a promising way to improve software quality evaluation. Conversely, learning capabilities are increasingly embedded within software, including in critical domains such as automotive and health. This calls for the application of quality assurance techniques to ensure the reliable engineering of ML-based software systems.

The aim of MaLTeSQuE is to provide a forum for researchers and practitioners to present and discuss new ideas, trends and results concerning the application of ML methods to software quality evaluation and the application of software engineering techniques to self-learning systems. We expect that the workshop will help in (1) validating existing ML methods for software quality evaluation as well as their application to novel contexts, (2) evaluating the effectiveness of ML methods, both compared to other automated approaches and the human judgment, (3) adapting ML approaches being already used in other areas of science in the context of software quality, (4) designing new techniques to validate ML-based software, inspired by traditional software engineering techniques.

2. KEYNOTE TAKEAWAY
Lionel Briand from the University of Ottawa, Canada and from the Interdisciplinary Centre of Security and Trust at the University of Luxembourg delivered the keynote entitled “Effective Use of AI in Automated Software Testing: Practicality and Scalability Benefits”.

Testing is the main mechanism used in industry to assess and improve the dependability of software systems. Various techniques from Artificial Intelligence (AI), e.g., evolutionary computing, machine learning, and natural language processing, can lead, if integrated properly, to scalable and practical test automation solutions. The talk covered recent and representative examples of novel applications of AI to software test automation, done in collaboration with industry partners in the satellite and automotive domains. It was complemented with lessons learned and future research directions. Lionel Briand stressed the fact that effective solutions often involve multiple techniques, inviting us to combine them and to mix different expertises. Additionally, Lionel Briand mentioned that to fill the industry-academia gap, academia should focus on ignored industrial problems, consider the hypotheses and contextual factors that may hinder the deployment of software engineering solutions.

3. INSIGHTS FROM PAPER SESSIONS
The workshop also had three sessions: “Testing and debugging”, “On the role of data” and “Quality attributes”. Each session was followed by a panel to engage the discussion between speakers and the audience.

*FNRS Research Associate
3.1 Testing and debugging
Aravind Nair introduced the session by explaining how machine learning can be used for predicting metamorphic relations [5], relying on mutation testing as a data augmentation technique. This approach reduces the risks of overfitting for small datasets.

Markus Borg followed by exposing an open implementation of the SZZ algorithm, complaining that existing implementations are not publicly available [1]. The SZZ algorithm is used to identify bug-introducing commits. He then illustrated this implementation to extract data from the open source Jenkins repository. This data is finally used to train machine learning classifiers to predict bug-introducing commits. The results were not so encouraging, but deserve further evaluation.

3.2 On the role of data
The second session questioned the importance of data in the use of machine learning algorithms. As Markus Borg rightfully pointed in the first session, “ML is data-hungry”, and one would add that the quality of this data does matter.

Michael Felderer reinforced this statement and presented a risk assessment approach for the use of ML applications [2]. In particular, he presented a method to evaluate the risk of having poor data for some of the features (data smells, pipeline, etc.) and their potential impact on the ML model, into a risk factor being a combination of the two.

Dario di Nucci explained the role of balancing in the process of analysis and prediction of code smells [6]. The study covers five balancing techniques and found that SMOTE has the best performance. Yet, it was not always possible to use SMOTE when a small number of smells are present in the dataset and the performance improvement is not very important.

3.3 Quality attributes
The last session focused on using the machine learning to assess quality attributes. Nickolay Viuginov presented a technique to autofold code of lower importance in an IDE [9], allowing developers to focus on more relevant parts and facilitating comparisons amongst large files.

Valentina Lenarduzzi aimed at surgically precise technical debt estimation [4]. Starting from the observation that existing metrics used by SonarQube are too coarse-grained and are not taking into account development efforts and historical data, she therefore offered to use them in ML-based technical debt estimation techniques.

Valentina Lenarduzzi introduced the session by explaining how machine learning can be used for predicting metamorphic relations [5], relying on mutation testing as a data augmentation technique. This approach reduces the risks of overfitting for small datasets.

Markus Borg followed by exposing an open implementation of the SZZ algorithm, complaining that existing implementations are not publicly available [1]. The SZZ algorithm is used to identify bug-introducing commits. He then illustrated this implementation to extract data from the open source Jenkins repository. This data is finally used to train machine learning classifiers to predict bug-introducing commits. The results were not so encouraging, but deserve further evaluation.

To help increase the visibility of the workshop, the 3rd edition of the MaLTEsQueE workshop will be summarized by a special issue of the the Journal of Systems and Software whose initial submissions were received in November 2019. We invite extended and revised versions of the papers presented at the workshop, as well as newly submitted, original works in this area.

5. REFERENCES