Methodological Aspects of Software Engineering, Assurance, Quality, and Reliability Engineering (MAS/AQR) 2019 Welcome from the Minitrack Co-Chairs

Bastian Tenbergen
Department of Computer Science
State University of New York at Oswego, USA
bastian.tenbergen@oswego.edu

Benoît Ries
University of Luxembourg, Esch-sur-Alzette, Luxembourg
benoit.ries@uni.lu

1. Description of the Minitrack

It is increasingly often the case that in contemporary products, innovation and value-added benefit is achieved due to software features, rather than novel ways to implement hardware. For example, while years ago, engine throttle in a car was achieved by a physical cord running from the gas pedal to the engine, in contemporary automobiles, throttle is achieved digitally, by reading the gas pedal position and transmitting it to a control unit. The significance of software is increasing, not only with regard to modern products, but also with regard to the skills and required knowledge of software engineers. Software engineers must be concerned with more than just DevOps. They must also be concerned with the related disciplines (e.g., electrical engineering), product quality, system reliability, and safety. Moreover, software engineers must be able to successfully communicate their findings, work products, engineering choices, and produce suitable assurance documents.

Instructing the knowledge and skills necessary to successfully do so requires carefully designed instructional methods, novel approaches, as well as established best practices. This is the aim of the MAS/AQR 2019 minitrack. We sought thought-provoking and highly constructive discussions among a broad audience and presenters to jointly identify promising educational approaches, explore challenges, share experiences, ideas, and new impulses regarding methodological aspects of software engineering. In particular, we placed emphasis on Software Assurance (e.g., safety, security, or privacy assurance), Software Quality (e.g., user testing, formal verification, and code refactoring), as well as Reliability Engineering. This minitrack therefore shall serve as a platform to facilitate collaboration between researchers and educators, both in industry as well as in academia.

2. Program Committee and Review Process

Each paper submitted to the MAS/AQR underwent thorough review, by at least four experts in the field. To ensure comparable high-quality reviews each paper has been reviewed by experts from conceptual modeling as well as experts in the field of software engineering education. Furthermore, each paper was reviewed in a double-blind fashion, strictly controlling for conflicts of interest (see Principle 1.3 in https://www.acm.org/code-of-ethics). The following individuals served as the program committee:

Mark Ardis
Dan Bagert — *Benedictine College (USA)*
Fabio Binder
Pontifícia Univ. Católica do Paraná (Brazil)
Jennifer Brings
Univ. of Duisburg-Essen (Germany)
Christopher Bull — *Brown Univ. (USA)*
Y C Cheng — *Taipei Tech (Taiwan)*
Steve Chenoweth
Rose-Hulman Institute of Tech (USA)
Tayana Conte
Univ. Federal do Amazonas (Brazil)
Marian Daun
Univ. of Duisburg-Essen (Germany)
Supannika Koolmanojwong
Univ. of Southern California (USA)
Dieter Landes — *HS Coburg (Germany)*
Patrick Letouze
Univ. Federal do Tocantins (Brazil)
Yihao Li — *Univ. of Texas at Dallas (USA)*
Maíra Marques — *Univ. de Chile*
Nancy Mead — *Carnegie Mellon Univ. (USA)*
Jürgen Mottok — *OTH Regensburg (Germany)*
Rory O'Connor — *Dublin City Univ. (Ireland)*
Mark Paulk — *Univ. of Texas at Dallas (USA)*

Y. Raghu Reddy *IIIT Hyderabad (India)*
Benoît Ries
Univ. of Luxembourg (Luxembourg)
Daniel Schlegel
State Univ. of New York at Oswego (USA)
Marcelo Schots
Rio de Janeiro State Univ. (Brazil)
Yvonne Sedelmaier *HS Coburg (Germany)*
Mauricio Souza
Federal Univ. of Minas Gerais (Brazil)
Bastian Tenbergen
State Univ. of New York at Oswego (USA)
Naoyasu Ubayashi *Kyushu University (Japan)*
Norha M. Villegas *Univ. ICESI (Colombia)*
Charles Wallace *Michigan Tech Univ. (USA)*
Daniela Zehetmeier *FH München (Germany)*

3. Minitrack Program

This year, Methodological Aspects of Software Engineering, Assurance, Quality, and Reliability Engineering (MAS/AQR) emerged from joining two minitracks co-located at the 52nd Hawaii International Conference on System Sciences 2019 (HICSS-52) as part of the Invited Track "Software Engineering Education and Training." This track has a long, successful history as a standalone conference known as CSEE&T and took place for the first time as part of the HICSS conference. From more than 50 contributions submitted to all Software Engineering Education and Training minitracks, five submissions were accepted to MAS/AQR.

In [1], Roberto Flores describes the design of and experiences with a software engineering fundamentals course for 2nd year baccalaureate level students. In particular, the course design emphasized UML patterns as well as usability concerns and overall project quality. It is particularly interesting that the author made a strong and conscientious choice to provide open educational resources in their course design, which invites others to adopt the course design and share their findings.

In [2], Petri Inhatola and Andrew Petersen investigate more than 800 student-produced software solutions to introductory python assignments in CS1-level programming courses. Their aim was to ascertain if traditional complexity metrics (such as cyclomatic complexity) correlates with student learning outcomes. Results show that no such correlation could be found, suggesting that for novice programmers, established complexity metrics may not be a useful way to ascertain source code evolution.

Marian Daun and Bastian Tenbergen follow up on their previous research on teaching requirements engineering to undergraduate students using industry-realistic case examples in [3]. Specifically, the authors improved a course design for graduate and undergraduate courses in Germany to an undergraduate setting in the US. They summarize quantifiable results and experiences with both application in the US as well as in Germany.

A particularly daunting task for many software engineering educators is to teach computational thinking to non-engineering students. In [4], Keeheon Lee and Youn Ah Khang tackle this issue by making clever use of embedded systems-type consumer electronics and encourage students to think from a business perspective about the solutions to be engineered into the product.

Thorsten Haendel proposes a combined approach using design patterns and gamification to instruct software refactoring needs, strategies, and techniques in [5]. By competing against other student teams and/or benchmark scores, students can learn to anticipate the need for refactoring, avoid bad design patterns proactively, and incrementally build experience in such matters over time.

We hope that these minitrack contributions find wide-spread use in the software engineering education community.

References

[1] Flores, R., "The Quest for a Practical Sophomore-Level Software Engineering Course."

[2] Inhatola, P., Petersen, A., "Code Complexity in Introductory Programming Courses."

[3] Tenbergen, B., Daun, M., "Industry Projects in Requirements Engineering Education: Application in a University Course in the US and Comparison with Germany."

[4] Lee, K., Kang, Y. A., "Bringing Computational Thinking to Nonengineering Students through a Capstone Course."

[5] Haendler, T., Neumann, G., "Serious Refactoring Games."