SATToSE 2017: The Post-Proceedings Editorial

Haidar Osman[1], Andrei Chis[2], Davide Di Ruscio[3], Vadim Zaytsev[4]

[1] University of Bern, Switzerland osman@inf.unibe.ch
[2] Feenk GmbH, Switzerland chisvasileandrei@gmail.com
[3] University of L'Aquila, Italy, davide.diruscio@univaq.it
[4] Raincode Labs, Belgium, vadim@grammarware.net

Venue

SATToSE is the Seminar Series on Advanced Techniques and Tools for Software Evolution. Its previous editions have happened in Waulsort (Belgium, 2008), Côte d'Opale (France, 2009), Montpellier (France, 2010), Koblenz (Germany, 2011, 2012), Bern (Switzerland, 2013), L'Aquila (Italy, 2014), Mons (Belgium 2015), Bergen (Norway 2016). Its tenth edition took place in Madrid, Spain on 7–9 June 2017. Each edition of SATToSE witnesses presentations on software visualisation techniques, tools for coevolving various software artefacts, their consistency management, runtime adaptability and context-awareness, as well as empirical results about software evolution.

The goal of SATToSE is to gather both undergraduate and graduate students to showcase their research, exchange ideas, improve their communication skills, attend and contribute technology showdown and hackathons.

The highlights of the programme included two invited talks given by Serge Demeyer and Joost Visser, an interactive tutorial by Felipe Ortega Soto, and a hands-on hackathon by Felienne Hermans. The detailed programme, as well as the pre-proceedings drafts can be found on our website:
http://sattose.org/2017.

Selection process

Each pre-proceedings submission was reviewed by at least three different peers. All submissions with a conflict of interest with one of the editors (co-authored by them or their colleagues) were handled by the other editor. We would like to express our gratitude to the program committee (listed in lexicographic order) who provided the reviews.

- Anya Helene Bagge
- Alexandre Bergel
- Andrea Caracciolo
- Tommaso Dal Sasso
- Serge Demeyer
- Coen De Roover
- Davide Di Ruscio
- Anne Etien
- Mohammad Ghafari
- Michael W. Godfrey
- André Hora
- Mircea Lungu

- Kim Mens
- Nevena Milojković
- Sebastiano Panichella
- Luca Ponzanelli
- Alexander Serebrenik
- Vadim Zaytsev

The call for post-proceedings contributions was communicated to all participants after the event. Only some decided to pursue the finalisation of their contribution for the post-proceedings where they might have solicited more co-authors, changed the title, and included more results. As a result, we have received 5 submissions of the extended versions of pre-proceedings abstracts.

Each submitted report for the post-proceedings has been assigned a shepherd to ensure that the authors took the reviews from the pre-proceedings phase into account. The emphasis was put on clear problem definitions and descriptions of advanced aspects of the techniques contemplated in the solution, as opposed to the finality of the obtained results. Thus, most submissions are intermediate reports on ongoing work or summaries of previously developed tools and papers.

Organisation

- **General Chair:** Gregorio Robles (Universidad Rey Juan Carlos)
- **Program Co-Chairs:**
  - Haidar Osman (University of Bern)
  - Andrei Chis (Feenk GmbH)
- **Hackathon Chair:** Felienne Hermans (Delft University of Technology)
- **Social Media Chair:** Vadim Zaytsev (Raincode Labs)
- **Steering Committee Chair** Kim Mens (Université catholique de Louvain)
- **Steering Committee:**
  - Gregorio Robles (Universidad Rey Juan Carlos, Spain)
  - Anya Helene Bagge (University of Bergen, Norway)
  - Mircea Lungu (University of Groningen, The Netherlands)
  - Davide Di Ruscio (University of L'Aquila, Italy)
  - Vadim Zaytsev (Raincode Labs, Belgium)
  - Coen De Roover (Vrije Universiteit Brussel, Belgium)
  - Oscar Nierstrasz (University of Bern, Switzerland)
- **Post-proceedings Editors:** Haidar Osman (University of Bern)

Contents of the volume

- ◇ *Bringing Incremental Builds to Continuous Integration*
  Incremental builds can considerably speed up the edit-compile-test loop during program development. While this technique is commonly used for local builds, it is seldom enabled during continuous integration. Correctness of continuous integration builds is usually preferred to compilation speed. With current tools, it is not trivial to get both properties, but we show that it is theoretically achievable with a carefully designed system. We first assess the potential benefits of incremental builds in continuous integration environments. We then identify different reasons that prevent that optimisation in practice. From these, we derive requirements to be met by future build systems to support incremental continuous integration. These steps are illustrated with existing tools, research insight and sample cases from industry. Ultimately, this paper defines a new research direction at the intersection of build systems and continuous integration.
- ◇ *Test Refactoring: a Research Agenda*
  Research on software testing generally fo- cusses on the effectiveness of test suites to detect bugs. The quality of the test code in terms of maintainability remains mostly ignored. However, just like production code, test code can suffer from code smells that imply refactoring opportunities. In this paper, we will summerize the state-of-the-art in the field of test refactoring. We will show that there is a gap in the tool support, and pro- pose future work which will aim to fill this gap.
- ◇ *Assessing Test Suite Effectiveness Using Static Metrics*
  With the increasing amount of automated tests, we need ways to measure the test effectiveness. The state-of-the-art technique for assessing test effectiveness, mutation testing, is too slow and cumber- some to be used in large scale evolution studies or code audits by external companies. In this paper we investigated two alternatives, namely code coverage and assertion count. We discovered that code coverage outperforms assertion count by showing a relation with test suite effectiveness for all analysed project. Assertion count only displays such a relation in only one of the analysed projects. Further analysing this relationship between assertion count coverage and test effectiveness would allow to circumvent some of the problems of mutation testing.
- ◇ *The Impact of Automated Code Quality Feedback in Programming Education*
  While some university-level programming courses focus on software quality, often in introductory courses code quality is little touched upon due to time constraints. Students usually get feedback on code quality after the grading of their assignment, feed- back that cannot be used on that same assignment. Our aim is to improve students? skills for code quality during the evolution of a students? programming assignment, while keeping the overhead low for teaching staff as well as for students. Better Code Hub is a service that checks code quality according to ten guidelines. We employ Better Code Hub as a formative assessment and feedback tool enabling students to monitor their

progress on code quality. Our findings indicate that there is an improvement in the code quality of the students? assignments over the period the tool is used. Our experiments show that students benefited the most from feedback on unit length, unit complexity, and code duplication.

◇ *Analysis of a Clone-and-Own Industrial Automation System: An Exploratory Study*
In industry, the development of similar products is often addressed by cloning and modifying existing artifacts. This so-called clone-and-own approach is often considered to be a bad practice but is perceived as a favorable and natural software reuse approach by many practitioners. Unfortunately, current literature lacks quantitative information about the positive and negative effects of clone-and-own. In this paper, we present the results of our exploratory analysis of an industry system developed using the clone-and-own approach. We found that products from the same product family can vary significantly in change activity over time, divergence from their origin and synchronization activity. We will further investigate these factors to develop quantitative measures for the assessment of clone-and-own benefits and drawbacks.