SATToSE 2016: The Post-Proceedings Editorial

Haidar Osman[1], Davide Di Ruscio[2], Vadim Zaytsev[3],
Mircea Lungu[4], Anya Helene Bagge[5]

[1] University of Bern, Switzerland osman@inf.unibe.ch
[2] University of L'Aquila, Italy, davide.diruscio@univaq.it
[3] Raincode, Belgium, vadim@grammarware.net
[4] University of Groningen, The Netherlands m.f.lungu@rug.nl
[5] University of Bergen, Norway anya@ii.uib.no

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), and Mons (Belgium 2015). Its ninth edition took place in Bergen, Norway on 11–13 July 2016. 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 four invited talks (given by Oscar Nierstrasz, Venera Arnaoudova, Aiko Yamashita, and Romain Robbes), an interactive tutorial (by Venera Arnaoudova), and a hands-on hackathon (by Gregorio Robles). The detailed programme, as well as the pre-proceedings drafts can be found on our website: http://sattose.org/2016.

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.

- ◇ Bram Adams
- ◇ Alexander Bergel
- ◇ Serge Demeyer
- ◇ Coen De Roover
- ◇ Michael W. Godfrey
- ◇ Georgios Gousios
- ◇ Kim Mens
- ◇ Haidar Osman
- ◇ Romain Robbes
- ◇ 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 6 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:** Anya Helene Bagge (University of Bergen)
◇ **Program Chair:** Mircea Lungu (University of Groningen)
◇ **Hackathon Chair:** Gregorio Robles (Universidad Rey Juan Carlos)
◇ **Proceedings Chair:** Haidar Osman (University of Bern)
◇ **Social Media Chair:** Vadim Zaytsev (Raincode)
◇ **Local Organization Chair:** Anna Maria Eilertsen (University of Bergen)
◇ **Local Staff:**
- Katerina Ivanova (University of Bergen)
- Eivind Jahren (University of Bergen)
- Håkon Lerring (University of Bergen)
- Niklas Trippler (University of Bergen)
- Patrick Monslaup (University of Bergen)

◇ **Steering Committee Chair** Kim Mens (Université catholique de Louvain)
◇ **Steering Committee:**
- Anya Helene Bagge (University of Bergen, Norway)
- Coen De Roever (Free University Brussels)
- Davide Di Ruscio (University of L'Aquila, Italy)
- Michael W. Godfrey (University of Waterloo)
- Mircea Lungu (University of Groningen)
- Oscar Nierstrasz (University of Bern)
- Vadim Zaytsev (Universiteit van Amsterdam)

◇ **Post-proceedings Editors:**
- Haidar Osman (University of Bern)
- Mircea Lungu (University of Groningen)

Contents of the volume

- ◇ *Beyond Context-Oriented Software*
  The last two decades have seen a lot of research on context-aware and context-oriented software development technologies, across subfields of computer science. This research area is slowly starting to mature and researchers currently explore how to unify different solutions proposed in these subfields. We envision that within another decade some of these solutions will make it into mainstream software development approaches, tools and environments. Most end-user software built by that time will be context-aware and able to adapt seamlessly to its context of use (devices, surrounding environment, and users? preferences). This transition from traditional to context-oriented software also requires a mindset shift in users. If users are to accept adaptive systems, they need to be in control. Context-orientation should evolve to become less technology- and more user-centric, putting the user back in control. A first step is to provide good feedback to the user about when and what adaptations take place, and mechanisms to allow users to partly control certain adaptations, followed by easily usable and understandable personalisation mechanisms dedicated to each end user. Eventually, when adaptive systems become completely natural and adopted by end users, this will culminate in our vision where users are in full control of relevant features or adaptations of applications of their interest, selected on-demand from online feature clouds, and integrated automatically into the running system.
- ◇ *Building Ecosystem-Aware Tools Using the Ecosystem Monitoring Framework*
  Integrating ecosystem data into developer tools can be very beneficial but is usually complicated. By automating the routine parts of this task we can reduce the amount of work needed to develop these tools. We have developed a framework that allows developers to quickly develop new tools that use ecosystem data. This framework automates the execution of user-defined analyses on ecosystem projects, allowing the developer to focus only on what ecosystem data is needed for her tool and how to present it.
- ◇ *On the Non-Generalizability in Bug Prediction*
  Bug prediction is a technique used to estimate the most bug-prone entities in software systems. Bug prediction approaches vary in many design options, such as dependent variables, independent variables, and machine learning models. Choosing the right combination of design options to build an effective bug predictor is hard. Previous studies do not consider this complexity and draw conclusions based on fewer-than-necessary experiments. We argue that each software project is unique from the perspective of its development process. Consequently, metrics and machine learning models perform differently on different projects, in the context of bug prediction. We confirm our hypothesis empirically by running different bug predictors on different systems. We show there are no universal bug prediction configurations that work on all projects.

◇ *Towards Efficient Object-Centric Debugging with Declarative Breakpoints*
Debuggers are central tools in IDEs for inspecting and repairing software systems. However, they are often generic tools that operate at a low level of abstraction. Developers need to use simple breakpoint capabilities and interpret the raw data presented by the debugger. They are confronted with a large abstraction gap between application domain and debugger presentations. We propose an approach for debugging object-oriented programs, using expressive and flexible breakpoints that operate on sets of objects instead of a particular line of source code. This allows developers to adapt the debugger to their domain and work at a higher level of abstraction, which enables them to be more productive. We give an overview of the approach and demonstrate the idea with a simple use case, and we discuss how our approach differs from existing work.

◇ *Comparing The Accumulation Of Technical Debt Between Two Applications Developed With Spring Web MVC And Apache Struts 2*
This paper presents the results of an observational study that investigates the differences between two widely used software development frameworks for Java EE applications. Also, it presents the accumulation of Technical Debt and the evolution of code quality metrics of software developed using these frameworks. Considering that web applications hold the lion?s share of today?s IT industry, this study focuses on two widely popular Java EE frameworks, namely Spring Web MVC Framework and Apache Struts 2. In particular, we have developed one system over four versions in both frameworks while monitoring Technical Debt and code quality metrics. The findings indicate that software developed based on these frameworks is relatively free of Technical Debt. Moreover, we have not noticed any significant differences between the two frameworks in terms of Technical Debt, from the perspective of source code metrics. Finally, conducting this study, we realized that if the framework is properly used it can potentiality lead to high quality and maintainable systems.

◇ *CSS Corpus for Reproducible Analysis*
Reproducibility of research heavily depends on the availability of the datasets from the experiments in the context of metaprogramming, the corpus of the code that was used to run the analyses and transformations. In the case of CSS, the problem is even more acute since the web is a constantly changing environment where the same address can refer to a frequently changing artefact. In this report, we explain how we created a corpus of CSS files as a part of our project of building a framework for analysing style sheets. We also include two case studies of explanatory nature showing how style sheets from various websites go about coding conventions and about code duplication. We believe this work will be useful for other CSS researchers to compare techniques they develop, on a uniform yet realistic dataset.