SATToSE 2014: The Post-proceedings Editorial

Davide Di Ruscio[†] and Vadim Zaytsev[‡]

[†]University of L'Aquila, Italy, davide.diruscio@univaq.it
[‡]Universiteit van Amsterdam, The Netherlands, vadim@grammarware.net

Venue

SATToSE is the Seminar Series on Advanced Techniques and Tools for Software Evolution. Its previous editions has happened in Waulsort (Belgium, 2008), Côte d'Opale (France, 2009), Montpellier (France, 2010), Koblenz (Germany, 2011, 2012) and Bern (Switzerland, 2013). Its seventh edition took place in L'Aquila (Italy) on 9–11 July 2014. 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 five invited lectures (given by Marianne Huchard, Leon Moonen, Alfonso Pierantonio, Alexander Serebrenik, Tanja Vos), an interactive tutorial (by Anya Helene Bagge) and a hands-on hackathon (by Alexander Serebrenik). The detailed programme, as well as the pre-proceedings volume with moderated but not yet reviewed abstracts and drafts can be found on our website: http://sattose.org/2014.

Selection process

We would like to express our gratitude to the following people (listed in lexicographic order) who provided the reviews for the submissions of this volume:

- Gabriela Arévalo
- Çiğdem Aytekin
- Anya Helene Bagge
- Antonio Cicchetti
- Juri Di Rocco
- Davide Di Ruscio
- Mike Godfrey
- Mathieu Goeminne
- Ammar Hamid
- Michiel Helvensteijn
- Ludovico Iovino
- Andy Kellens
- Xavier Le Pallec
- Angela Lozano
- Mircea Lungu
- Tom Mens
- Oscar Nierstrasz
- Bogdan Vasilescu
- Sylvain Vauttier
- Tanja Vos

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, others often solicited more co-authors, changed the title, included more results. As a result, we have received 12 submissions: one of them was a keynote add-on, one was a tutorial write-up, the rest were being extended versions of pre-proceedings abstracts.

Each submitted report has been 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. 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. The authors received enough time after the review to take the feedback into account and finalise their submissions.

Organisation

- ◇ **General Chair:** Davide Di Ruscio (University of L'Aquila)
- ◇ **Program Chair:** Vadim Zaytsev (Universiteit van Amsterdam)
- ◇ **Hackathon Chair:** Alexander Serebrenik (TU Eindhoven)
- ◇ **Steering Committee Chair** Ralf Lämmel (Universität Koblenz)
- ◇ **Local Committee:**
  - Romina Eramo (University of L'Aquila)
  - Ludovico Iovino (University of L'Aquila)
- ◇ **Steering Committee:**
  - Michael W. Godfrey (University of Waterloo)
  - Marianne Huchard (Université Montpellier 2)
  - Tom Mens (University of Mons)
  - Oscar Nierstrasz (University of Bern)
  - Coen De Roever (Free University Brussels)
  - Vadim Zaytsev (Universiteit van Amsterdam)
- ◇ **Post-proceedings Editors:**
  - Davide Di Ruscio (University of L'Aquila)
  - Vadim Zaytsev (Universiteit van Amsterdam)

Contents of the volume

- ◇ *A Critique on Code Critics*
  Pharo (a Smalltalk IDE) has a recommendation facility called "code critics" which is used for signalling code controversial and suspicious implementation choices and for promoting good style language idioms. For instance, they report code smells and encourage correct use of object orientation. In this paper, code critics are re-evaluated based on a case study of the code of a big system with 51 packages (Moose). It was observed that some code critics tend to occur together and therefore should be grouped in issues of a higher level of abstraction; others occur only in presence of others up to the point of being overshadowed by them; some are even triggered by the same patterns and offer competing solutions. The approach is promising and can be applied broadly to any quality-driven program analysis.
- ◇ *User Interface Level Testing with TESTAR*
  Software testing is exercised at many levels, and if done at the Graphical User Interface (GUI) level, allows to create very realistic test cases due to close emulation of user behaviour. This report offers an extension of the TESTAR tool by the same authors and their collaborators. TESTAR implements a model-driven approach to test automation which generates test cases based on a model which is automatically derived from the GUI through the accessibility API. The extension revolves around action ranking and using machine learning techniques in order to execute sensible and sophisticated sequences of GUI actions instead of blindly making a random selection of clicks and keystrokes that are visible and unblocked in each state.
- ◇ *Qualifying Chains of Transformation with Coverage-based Evaluation Criteria*
  Development of complex and large model transformations can be optimised by composition of reusable smaller ones. Yet, composing them is a complex problem: typically smaller transformations are discovered and selected by developers from different and heterogeneous sources, and then chained by composition processes that are semi-automated at best. Without considering the semantic properties of a transformation, it is difficult for the user to make the right choice between two possible proposed chains. This report contains a proposal to classify such chains with respect to the coverage of the metamodels involved in the transformation.
- ◇ *Describing the Correlations between Metamodels and Transformations Aspects*
  The paper considers two main concepts in MDE: a metamodel as a model that all regular models in the ecosystem conform to; and a model transformation as a description of changes in models. Surfacing, formalising and maintaining relations between the artefacts of these two kinds is a big subdomain of modelware. This report presents an approach to understand structural characteristics of metamodels by looking at how model transformations depend on metamodels of their source and target models.

- ◇ *A Three-level Formal Model for Software Architecture Evolution*
  Architecture Description Languages are the focus of this paper: a brief overview of them is given, and one (Dedal) is explain in full detail. It formally defines the architecture of a software system on three levels: the abstract specification, the concrete configuration and the instantiated architecture assembly. Software evolution is expressed in Dedal in static invariants that bind descriptions together and in evolution rules that trigger changes at each level. The running example in the paper is a home automation software that is used to guide the reader through explanations of evolution in Dedal. The authors consider integrating the methods presented here, in an IDE platform to help developers embrace software evolution further.
- ◇ *Representing Uncertainty in Bidirectional Transformations*
  Bidirectional transformations (BX) are a largely recognised quickly growing field of MDE. In BX, we usually deal with both ways of propagating information between two (or more) software artefacts — such changes are typically non-univocal in the sense that more than one correct solutions could be admitted and tolerated. However, most existing BX frameworks do not represent uncertainty explicitly. In this report, the authors insist that they should and show that it is quite possible in their framework, by re-explaining the non-deterministic nature of bidirectionality and presenting a case study with a family of cohesive models with uncertainty.
- ◇ *Towards a Taxonomy for Bidirectional Transformation*
  Software model consistency and synchronisation are often achieved by designing and implementing bidirectional transformations. This report demonstrates an early attempt to take a step towards a taxonomy of BX by illustrating a set of relevant properties of such mappings. The contribution is of an analytical nature: existing literature is analysed and characteristics of existing approaches are put into perspective.
- ◇ *Languages, Models and Megamodels*
  Software modelling is considered to be a somewhat difficult or even obscure subdomain of software engineering, with some of its topics such as megamodelling leaning towards the obscure part. However, it can be explained in relatively simple terms and given to software evolution researchers, can prove to be a useful advanced technique of expressing relations between software artefacts. During SATToSE 2014, we have held an interactive tutorial on this topic, which this report tries to summarise and complement.