Autom Softw Eng (2013) 20:137–139
DOI 10.1007/s10515-013-0121-3

Guest editors introduction: special issue on innovative automated software engineering tools

John Grundy · John Hosking

Received: 28 January 2013 / Accepted: 31 January 2013 / Published online: 9 February 2013
© Springer Science+Business Media New York 2013

1 Introduction

By definition, "automated" software engineering needs the support of automation tools, in order to be effective (or even possible) (Grundy and Hosking 2001). Many tools have been developed to support automation, in both narrow and broad domains. These range across AI toolkits, theorem provers and model checkers; requirements, design, coding and testing support tools; various configuration management, process enactment and project management support tools; and code generators, code analysis, visualisation, refactoring and reverse engineering tools.

To continue to advance the field of Automated Software Engineering, good automation-support tools need to be developed and deployed alongside, and in combination with, new and improved ASE techniques. Such tools are usually themselves extremely complex engineered software artifacts. ASE tools are challenging to design, to build, to scale, to make robust, and to integrate and evolve. To engineer such increasingly complex tools, we must investigate new directions in tool engineering and deployment. We need new approaches to building, scaling, and deploying tools, new domains or ways in which to apply tools, and new techniques for synthesizing tools.

J. Grundy (✉)
Centre for Computing & Engineering Software Systems, Swinburne University of Technology, Melbourne, Australia
e-mail: jgrundy@swin.edu.au

J. Hosking
College of Engineering & Computer Science, Australian National University, Canberra, Australia
e-mail: john.hosking@anu.edu.au

2 Background

Traditionally Automated Software Engineering (ASE) has been supported by a variety of tools. These particularly include tools for theorem proving, model checking and other complex model analysis all being techniques which are extremely difficult if not impossible to do without tool support (Holzmann 1997). Other early ASE tools developed included tools to assist in requirements capture and analysis, particularly for very formal requirements modeling; tools to support software testing, particularly test case generation and test result analysis; and tools to support complex software development processes. These included CASE (Computer Aided Software Engineering) tools with model analysis features, process-centered environments with enactable software processes, project management tools, and version control and configuration management tools.

Wider applications of automation to Software Engineering have become popular in more recent times. These have included ASE techniques for model construction including those for requirements, design, coding, or combinations of these. These techniques have been embodied in various Model-Driven Engineering processes and associated tool support (Scmidt 2006). Other popularized application areas have been ASE tools for reverse engineering, refactoring, and visualisation of models and/or code. While such tools have existed for some time, the complexity of modern software applications has meant they have become critical for many development and maintenance tasks. Even more recent areas of application include various data mining, search-based software engineering applications and other knowledge-intensive software tasks, driven again by the huge increase in size of software and which in turn require sophisticated ASE tools to support them (Harman and Jones 2001).

3 Special issue focus

We sought substantial, archival contributions to the ASE literature that included either new application areas of ASE tools, new innovations in applying ASE tools to traditional areas, or new ways of realizing innovative ASE tools. The latter includes architecting of ASE tools and addressing challenging issues of scaling, robustness, reliability and integration. We asked authors to focus on the *tool* aspect, not the *technique* aspect of their work. We wanted journal readers to be able to learn important lessons about tool innovation in the target tool domain(s) so that other researchers could benefit from the work presented. We expected evaluation to be holistic. Some tools can be clearly evaluated and compared to other tools by their performance, scaling to large models, and the range of support features offered. Others might have to be evaluated on their support for software engineers including tool usability, expressiveness, effectiveness, differentiation from other tools, and integration with other tools.

Overall we received 32 submissions to the special issue, a very gratifying number. All papers were refereed by at least three experts in the Automated Software Engineering community. After re-revision of nearly a dozen papers, we accepted 9 papers for the special issue, the first four of which appear in this issue of the ASE journal. The second set of papers will appear in a later issue of *Automated Software Engineering*.

4 Papers in part 1 of the special issue on innovative ASE tools

Arendt and Taentzer describe a framework and Eclipse-based support tool for model quality assurance. Their approach supports the definition and evolution of complex models necessary to support a range of complex, model-based software engineering tasks. Their innovative tool supports not only model capture and management, but analysis of models using a variety of metrics and then the application of a range of model refactorings to address model short-comings, or "bad smells". They evaluate their tool on several large model analysis and refactoring problems investigating tool scalability, performance, and suitability.

Walderhaug describes a novel ASE toolchain for the health domain. This toolset supports developers in engineering complex services for this domain using health concepts to ensure standardized, integrated services result. A model-driven development to services engineering is employed whereby complex health services are modeled abstractly then successively refined down to implementations. Key benefits include support for ensuring adherence to standardized concepts and interfaces, documentation of services, integration into complex service-oriented architectures, and traceability from requirements to service implementations. A detailed experiment with developers was performed to assess various aspects of the toolchain suitability.

O'Halloran describes an approach to automated verification of code using Simulink®. The CLawZ toolset provides a highly automated approach to verifying correctness of complex, dynamic code generated from the Simulink® tool. An auto-coder generates Ada code from the Simulink® specification. The CLawZ tool uses a formal model derived from the source model and a set of refinement script generators. A refinement checker and a theorem prover are then used to determine if errors exist in the source model and code. This ASE tool was evaluated by comparing effort used in a traditional testing-based approach to the effort used to deploy CLawZ on the same problem.

Nöhrer and Egyed describe a tool to guided decision-making in software engineering tasks. Their innovative ASE tool supports software engineers in capturing and reasoning about complex decision paths and dependencies in a range of contexts, including product line engineering. Their tool allows users to answer a set of questions in an arbitrary order and to have complex inter-dependencies analysed and users guided in terms of ordering, conflict avoidance, and conflict resolution. They evaluated their tool using six complex decision scenarios including architectural product line engineering and product configuration.

We hope that you enjoy this first installment on Innovative ASE tools!

References

Grundy, J.C., Hosking, J.G.: Software Tools, 2nd edn. Wiley Encyclopaedia of Software Engineering. Wiley, New York (2001)

Harman, M., Jones, B.F.: Search-based Software Engineering, Inf. Softw. Technol. **43**(14), 833–839 (2001)

Holzmann, C.J.: The model checker SPIN. IEEE Trans. Softw. Eng. **23**(5), 279–295 (1997)

Scmidt, D.C.: Model-driven Engineering, Computer **39**(2), 25–31 (2006)