Science of Computer Programming 134 (2017) 1–3

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Editorial

Guest editors' introduction to the 6th issue of Experimental Software and Toolkits (EST-6)

1. Introduction

Welcome to the sixth special issue on Experimental Software and Toolkits (EST) of Elsevier's Science of Computer Programming journal. The EST series aims at allowing academic software developers to publish the software systems they developed with similar rigor as academic papers. Peer reviewers evaluated not only a scientific paper about the developed system but also the system itself. The goal is facilitating the academic software development community to find a wider audience for their work. Moreover, by enhancing the accessibility and reproducibility of the submitted software, a wider application and adoption of experimental software is enabled.

2. WASDeTT

This sixth special issue was associated with WASDeTT 2013, the fourth International Workshop on Academic Software Development Tools and Techniques. This workshop was held on July 1, 2013, colocated with ECOOP / ECSA / ECMFA at Montpellier, France. The motivation for the WASDeTT workshop series was to create an opportunity for academic software developers to present and demonstrate novel tools, and to provide a forum for discussing with their peers on the issues and challenges they encountered in building such tools. The academic software developer has some idea, algorithmic or functional, and in order to prove or validate his or her idea, software is adapted or newly developed. Some of these systems are distributed to different researchers and occasionally become very popular in the community. But the development of these software systems is almost always in conflict with papers that are to be written. When the user community grows and requests for enhancements and improvements increase, the effort to maintain such systems may become overwhelming. WASDeTT and EST are our modest contributions to provide a way to tool builders to get more credit for their development work and more visibility for their tools, by providing a venue where both the tool and the corresponding system get published in a peer-reviewed journal.

The papers invited to submit to this special issue were selected from the submissions (tool demonstration papers), that where originally presented at WASDeTT 2013. The submission for this special issue consisted of a short paper describing the submitted tool and some of the tool building issues encountered, together with the tool itself. We did not limit submissions to a particular domain but accepted submissions about academic software systems in a broad sense. With the workshop and special issue we wanted to go one step beyond the traditional tool demonstrations at various conferences, because the reviewers were required to install and test the submitted systems. The software systems were reviewed on various aspects:

- ease of installation,
- quality of (user) documentation,
- ease of usage, and
- applicability and relevance to the intended domain.

The major improvement of this format over more traditional conference and workshop tool demonstrations is that it allows a broader audience to download, install, and use the system. Indeed, with this special issue we decided to publish the submitted software systems via SHARE [1] and Elsevier.

3. SHARE

To this extent, we invited Pieter Van Gorp to give a keynote talk at WASDeTT 2013, where he presented SHARE,[1] a web portal for creating and sharing executable research tools. SHARE provides an excellent infrastructure to host prototypes of

[1] https://is.ieis.tue.nl/staff/pvgorp/share/.

http://dx.doi.org/10.1016/j.scico.2016.11.001

academic tools. It alleviates both developers and users (in our case reviewers) from the installation hurdle. A SHARE virtual machine contains all the required software and documentation that is required for reproducing research contributions. In order to facilitate an easy testing of the submitted tools, we encouraged the tool builders to install their tools on the SHARE platform. We owe our gratitude to Pieter, who was very helpful during the entire process to help us make this work.

4. Submissions

We received nine high-quality submissions for this special issue. To ensure quality reviews worthy of this journal, for the review process every single paper was assigned to three different reviewers, as well as one of the special issue editors for follow-up during the reviewing process, to ensure consistency among how all papers were handled. To avoid conflicts of interest, submissions where one of the special issue editors was involved as a co-author, were handled separately as regular submissions to Science of Computer Programming, but following the same format including a review of the submitted system. Eventually, the following six papers and systems made it for inclusion in the special issue on Experimental Software and Toolkits of Science of Computer Programming.

- **Implementing record and refinement for debugging timing-dependent communication** by T. Felgentreff, M. Perscheid and R. Hirschfeld. This paper presents the *Peek-At-Talk* debugger for investigating non-deterministic failures with low overhead in a systematic, top-down method, with a particular focus on tool-building issues. First, they show how their debugging framework *Path Tools* guides developers from failures to their root causes and gathers run-time data with low overhead. Second, they present *Peek-At-Talk*, an extension to their *Path Tools* framework to record non-deterministic communication and refine behavioral data that connects source code with network events. Finally, they scope changes to the core library to record network communication without impacting other network applications.
- **Continuous quality assessment with inCode** by G. Ganea, I. Verebi and R. Marinescu. This paper introduces *inCode*, an Eclipse plugin aimed at transforming quality assessment and code inspections from a standalone activity, into a continuous process, fully integrated in the development life-cycle. But *inCode* not only assesses continuously the quality of Java systems; it also assists developers in taking restructuring decisions, and even supports them in triggering non-standard, complex refactorings. This paper introduces *inCode*'s differentiating features, presents the design goals that shaped construction decisions, and describes a controlled experiment designed to validate the usability of the tool.
- **Enabling PHP software engineering research in Rascal** by M. Hills, P. Klint and J.J. Vinju. This paper discusses the ongoing work on *PHP AiR*, a framework for PHP Analysis in Rascal. *PHP AiR* is focused especially on program analysis and empirical software engineering, and is being used actively and effectively in work on evaluating PHP feature usage and system evolution, on program analysis for refactoring and security validation, and on source code metrics.
- **Cost-effective evolution of research prototypes into end-user tools: The MACH case study** by H. Störrle. This paper presents the *MACH* Model Analyzer/Checker, a stand-alone tool with a command-line interpreter. *MACH* integrates a set of research prototypes for analyzing UML models. By choosing a simple command line interpreter rather than a (costly) graphical user interface, it achieves its core goal of quickly deploying research results to a broader audience while keeping the required effort to an absolute minimum. The paper analyzes *MACH* as a case study of how requirements and constraints in an academic environment influence design decisions in software tool development. The paper argues that the chosen approach while perhaps unconventional, serves its purpose with a remarkable cost-benefit ratio.
- **Mercury: Using the QuPreSS reference model to evaluate predictive services** by S. Martínez-Fernández, X. Franch and J. Bisbal. This paper presents *Mercury*, a tool based on the QuPreSS reference model and customized to the weather forecast domain. *Mercury* measures the quality of weather predictive services, and automates the context-dependent selection of the most accurate predictive service to satisfy a customer query. To do so, candidate predictive services are monitored so that their predictions can be eventually compared to real observations obtained from a trusted source. *Mercury* is a proof-of-concept of QuPreSS that aims to show that the selection of predictive services can be driven by the quality of their predictions. The paper shows how *Mercury* was built from the QuPreSS reference model and how it can be installed and used.
- **JPC: A library for categorising and applying inter-language conversions between Java and Prolog** by S. Castro, K. Mens and P. Moura. This paper presents an approach to enable a smooth interaction between Java and Prolog programs. Existing approaches provide limited support to allow programmers to customize how Prolog artefacts should be reified in the Java world, or how to reason about Java objects on the Prolog side. Appropriate mappings may depend on the particular context in which a conversion is accomplished. Although some libraries alleviate this problem by providing higher-level abstractions to deal with the complexity of custom conversions between artefacts of the two languages, such libraries are difficult to implement and evolve, because of a lack of appropriate underlying building blocks for encapsulating, categorizing and applying Java–Prolog conversion routines. This paper instead introduces a new library, *JPC*, serving as a development tool for both programmers willing to categorize context-dependent conversion constructs in their Java–Prolog systems, and for architects implementing frameworks providing higher-level abstractions for better interoperability between these two languages.
- **Managing facts and resources with the pica IDE infrastructure library** by A.H. Bagge. Whereas integrated development environments are becoming increasingly responsive and reactive to their users' editing activities, existing compilers cannot easily be adapted to this increased responsiveness. Instead, IDE developers typically have to develop their own

language frontends particularly targeted at IDE usage. To counter this situation, this paper describes the design of a library, *Pica*, for managing resources and information about the editor and user's workspace, and the facts produced by various analyses and compiler phases, to ease IDE integration. Although the paper illustrates how *Pica* is applied on the Eclipse IDE for the experimental Magnolia programming language, it is designed to be independent of the underlying IDE, and to provide support for multiple programming languages.

Acknowledgements

To conclude this introduction we would like to thank everyone who helped making this special issue possible. In particular, we thank the referees for performing their thorough and thoughtful reviews and re-reviews of the tools and papers, Pieter Van Gorp for his help with setting up and using the SHARE system, all the authors for their hard work on tools and submissions, and the SCP Editorial Office for supporting this special issue and making sure it had a soft landing, in spite of the many inevitable delays we encountered along the way.

References

[1] Pieter Van Gorp, Steffen Mazanek, SHARE: a web portal for creating and sharing executable research papers, Proc. Comput. Sci. 4 (January 2011) 589–597.

Mark G.J. van den Brand [a]
Jurgen J. Vinju [b,a]
Kim Mens [c]
[a] *Eindhoven University of Technology, Groene Loper 5, NL-5612 AZ Eindhoven, The Netherlands*
[b] *Centrum voor Wiskunde en Informatica, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands*
[c] *Université catholique de Louvain, Place Sainte Barbe 2, B-1348 Louvain-la-Neuve, Belgium*