Journal of Software: Evolution and Process

JOURNAL OF SOFTWARE: EVOLUTION AND PROCESS
J. Softw. Evol. and Proc. 2014; **26**:531–532
Published online 5 June 2014 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/smr.1655

Editorial

Guest editorial for the special issue on source code analysis and manipulation, SCAM 2012

This issue of the Journal of Software: Evolution and Practice concentrates on the bottom line in computer software engineering—the source code itself. While much attention in the wider software engineering community is directed toward other aspects of systems development and evolution, such as specification, design, and requirements engineering, it is the source code that contains the only complete and precise description of the behavior of the system. The analysis and manipulation of source code thus remains a pressing concern.

The three papers in this issue were selected and extended from the best papers presented at the 12th IEEE International Conference on Source Code Analysis and Manipulation (SCAM 2012) held in Riva del Garda, Italy, in September 2012. The aim of the SCAM conference is to bring together researchers and practitioners working on the theory, techniques, and applications that concern analysis and/or manipulation of the source code of computer systems. It is the source code that contains the only truly precise description of the behavior of such systems. Many other conferences and workshops address the applications of source code analysis and manipulation. The aim of SCAM is to focus on the algorithms, tools, and techniques of source code analysis and manipulation themselves—what they can achieve and how they can be improved, refined, and combined.

The SCAM 2012 conference received 40 regular paper submissions of outstanding quality, of which 16 papers were selected for presentation after review by at least three referees for relevance, soundness, and originality. For this special issue, seven of the very best of these were invited to submit extended and revised papers.

However, this special issue was not meant to be closed only to invited papers. A public open call was published to invite outstanding papers by other authors on source code analysis and manipulation. In total, seven papers were submitted to this special issue. Each of the submissions was reviewed by a minimum of three expert referees.

Following the first round of review, five papers were selected for final consideration in this special issue. The authors were asked to revise their papers in response to the referees' comments, and the revised drafts were then reviewed for a second time for conformance to the referees' comments. Following this second review, three papers were selected for publication in this special issue.

The selected papers represent some of the very best work that has appeared at SCAM, and cover all of its main areas of interest, namely program transformation, represented by Chris Wilcox, Michelle Mills Strout, and James M. Bieman; feature location, represented by André L. Santos; and impact analysis, represented by Lajos Schrettner, Judit Jász, Tamás Gergely, Árpád Beszédes, and Tibor Gyimóthy.

In the first paper, 'An optimization-based approach to LUT program transformations', Wilcox *et al.* present an approach to cope with the trade-off between performance and accuracy. In computational software, programmers may replace calls to mathematical primitives (especially trigonometric formulas) with accesses to tables that use precomputed data (look-up tables (LUT)). However, when this alternative design is adopted, it involves substantial manual effort that is prone to programming error. This paper presents an automatic approach to refactor a scientific-computation program to a program that uses LUT. The trade-off between speed and accuracy in using LUT is addressed as a multi-objective optimization problem.

The paper by André L. Santos, 'GUI Code Tracing Through Direct Program Interaction', presents a novel approach to feature location in Graphical User Interface (GUI) code, that is, the identification of the portions of the program that are executed when a graphical widget is triggered. The approach relies on a lightweight instrumentation based on aspect-oriented programming. The benefit of this tool is studied by involving human participants in a user study.

In the third paper, 'Impact Analysis in the Presence of Dependence Clusters Using Static Execute After in WebKit', Schrettner *et al*. present an approach based on the Static Execute After (SEA) for impact analysis. With the objective of identifying the extent of the program, potentially affected by a maintenance change, SEA is meant to be a faster alternative to forward program slicing. However, SEA trades accuracy for speed, to deliver a faster analysis with less precise results, thus returning larger impact sets. Prediction capabilities and size of impact sets are assessed empirically on a relevant case study, the WebKit.

We hope that you will find all of these papers to be of interest, and we encourage those who find them so to join us at the SCAM conferences to come.

Mariano Ceccato
Fondazione Bruno Kessler, Trento, Italy

Zheng Li
College of Information Science and Technology, Beijing University of Chemical Technology, Beijing, China

James R. Cordy
School of Computing, Queen's University, Kingston, Canada