11-2017

Guest editor's introduction to the special issue on source code analysis and manipulation (SCAM 2015)

Foutse KHOMH

David LO
Singapore Management University, davidlo@smu.edu.sg

Michael W. GODFREY

DOI: https://doi.org/10.1002/smr.1928

Received: 2 November 2017 | Accepted: 6 November 2017

DOI: 10.1002/smr.1928

WILEY Journal of Software: Evolution and Process

EDITORIAL

Guest editor's introduction to the Special Issue on Source Code Analysis and Manipulation (SCAM 2015)

Abstract

We are happy to introduce you to this special issue that presents selected papers from the 15th IEEE International Working Conference on Source Code Analysis and Manipulation (SCAM 2015). SCAM is a leading conference that brings together researchers and practitioners working on theory, techniques, and applications that concern analysis and/or manipulation of the source code of computer systems. While much attention in the wider software engineering community is properly directed towards other aspects of systems development and evolution, such as specification, design, and requirements engineering, it is the source code that contains the only precise description of the behavior of the system. The analysis and manipulation of source code thus remains a pressing concern. SCAM 2015 was held on September 27 to 28, 2015, in Bremen, Germany, together with 31st International Conference on Software Maintenance and Evolution (ICSME).

The research track of SCAM 2015 received a total of 68 submissions from 27 different countries. After a rigorous reviewing process, where each paper is reviewed by at least 3 PC members, a set of 24 research papers were accepted. An additional 6 tool demo papers were also accepted. Of these 24 research papers, 6 that received the highest reviewer scores were invited to submit an extended version to the *Journal of Software: Evolution and Process*. Four sets of authors agreed to extend their SCAM 2015 papers, and after another rigorous multiround reviewing process, they are accepted for this special issue.

Ghafari et al contributed a paper entitled "Mining unit test cases to synthesize API usage examples." In that paper, motivated by the challenge of finding adequate code examples for a given API, the authors propose an approach that identifies relevant statements in unit test cases of an API to serve as the API's usage examples. Different from many existing approaches, the proposed approach does not rely on the availability of a good set of external client code of the API. The authors have compared the prototype implementation of the proposed approach with a state-of-the-art approach and highlighted that the quality of the extracted code examples is superior and close to manually constructed examples.

Pandita et al contributed a paper entitled "TMAP: Discovering relevant API methods through text mining of API documentation." The authors present an approach, named TMAP, to help in migrating applications across multiple programming languages/platforms. TMAP performs text mining to identify relevant method mappings between a source API and a target API. The authors have demonstrated that TMAP is able to identify mappings for 15 classes across Java and C# API, and Java ME and Android API. Comparisons with results produced by 2 state-of-the-art baselines (ie, Rosetta and StaMiner) demonstrate that TMAP has a superior performance—on average, it is able to identify more than 50% more method mappings than those produced by either baselines.

Nicolay et al contributed a paper entitled "Purity analysis for JavaScript through abstract interpretation." The paper examines the purity of functions in JavaScript programs. A function is considered pure if none of its executions generate or depend upon externally observable side effects. Leveraging flow and intraprocedural analysis, the authors proposed an approach that can determine whether and to what extent a function in a JavaScript program is pure. The proposed approach was evaluated on a benchmark of JavaScript programs, and results showed that it is able to determine function purity in the presence of higher-order functions, dynamic property expressions, and prototypal inheritance. A comparison with existing purity analyses revealed that the proposed approach achieves similar or higher precision.

Aivaloglou et al contributed a paper entitled "Parsing Excel formulas: A grammar and its application on 4 large datasets." The paper presents a grammar for spreadsheet formulas that is compact and produces processable parse trees, suited for further manipulation and analysis, which makes it particularly interesting for spreadsheets research. The authors evaluated their proposed grammar against over 8 million unique formulas extracted from 4 spreadsheet datasets and report that it could successfully parse 99.99% of the formulas. The proposed grammar was also used to examine the characteristics of Excel formulas from the datasets, along 3 different dimensions: complexity, functionality, and data utilization. Results showed that most Excel formulas are simple and almost all formulas use data from other cells, which are often from different worksheets or external files.

We would like to thank all authors and reviewers of this special issue who have done a great job to write and review these 4 high-quality articles. We would also like to thank Gerardo Canfora and other members of *Journal of Software: Evolution and Process* and Wiley who have given us much guidance and help to publish this special issue.

Happy reading!

Foutse Khomh[1]
David Lo[2]
Michael W. Godfrey[3]
[1]*Polytechnique Montreal, SWAT Lab., Montreal, Quebec, Canada*
[2]*Singapore Management University, Singapore, Singapore*
[3]*David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada*

Correspondence Foutse Khomh, Polytechnique Montreal, SWAT Lab., Montreal, Quebec, Canada.
Email: foutse.khomh@polymtl.ca