Special issue on Source code analysis and manipulation

Abstract

The source code is the reference repository of knowledge, used during program comprehension and modification. SCAM, the IEEE Workshop on Source Code Analysis and Manipulation, brings together researchers and practitioners whose work revolves around the source code, its automatic or semi-automatic analysis, and its transformation.

This special issue features extended versions of the best papers presented during SCAM 2005 in Budapest. The two main topics of the selected papers are slicing and program transformations. These are quite representative of the mixture of subjects that are dealt with in this workshop and that range from theoretical properties of code analysis techniques, such as slicing, to practical applications of transformation and restructuring methods.

Keywords: Code analysis; Program transformation; Slicing; Verification and validation

1. Introduction

The aim of SCAM is to bring together researchers and practitioners working on theory, techniques and applications which 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 system development and evolution, such as specification, design and requirement 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.

For the purpose of clarity source code is taken to mean any fully executable description of a software system. It is therefore so construed as to include machine code, very high level languages and executable graphical representations of systems. The term analysis is taken to mean any automated or semi-automated procedure which takes source code and yields insight into its meaning. The term manipulation is taken to mean any automated or semi-automated procedure which takes and returns source code.

Started in 2001 from the seminal ideas of its initiator, Mark Harman, SCAM grew in submissions and attendance over the years, reaching its current format, consisting of a two-day event, structured around technical sessions with plenty of time allocated to discussion. Each technical session is structured as short presentations (15 min) followed by open discussion, based on controversial questions and issues raised by the authors of the papers presented. All attendees are encouraged to write ideas on plastic slides, rather than merely contributing verbally. These slides are collected and scanned for publication on the SCAM Web site and the materials are still available for most previous SCAM events.

The workshop maintained its initial aims and focus. The high quality of the submissions is witnessed by the regular publication of special issues in top-ranked journals of the field, featuring selected papers presented at the workshop.

2. History of SCAM

The first SCAM workshop (SCAM 2001) was held in Florence, on November 10th 2001, co-located with the International Conference on Software Maintenance (ICSM 2001), the Workshop on Empirical Studies of Software
Maintenance (WESS 2001) and the Workshop on Web Site Evolution (WSE 2001). Mark Harman was the initiator of this workshop series. He gave it a strong connotation in terms of covering the range of topics going from code analysis, to program transformation, understanding and maintenance, thus bridging the gap between communities such as compiler construction, static and dynamic analysis, abstract interpretation, slicing, reverse engineering and reengineering, or maintenance. SCAM 2001 was a one-day event with the emphasis on discussion; 40 people attended this first workshop edition. Since then SCAM participation steadily increased to more than 50 participants in recent SCAM workshops. There was a special issue of the Journal Information and Software Technology featuring extended versions of selected papers.

The second SCAM workshop (SCAM 2002) was held in Montréal, on October 1st 2002, once again co-located with ICSM and WSE. The workshop maintained the discussion-based emphasis. Selected papers were published in a special issue of the Journal of Automated Software Engineering.

The third SCAM workshop, held in Amsterdam, was co-located with ICSM and also with Software Technology Experience and Practice (STEP 2003), the ELIZA workshop, the workshop on Visualization of Software (VISSOFT 2003), and WSE 2003. Chris Verhoef gave the keynote presentation “Managing Multi-billion Dollar IT Budgets using Source Code Analysis”. Due to the large number of paper submissions in 2003, the organizers moved from the one-day format of previous years to a one-and-a-half-day format.

The fourth SCAM workshop was held in Chicago, in co-location with the 20th International Conference on Software Maintenance (ICSM 2004), the 10th International Symposium on Software Metrics (Metrics 2004) and WSE 2004. The large number of submissions and attendees have been accommodated into a two-day event, including regular paper presentations and discussion. The keynote talk, given by Gregg Rothermel, was entitled “Using Source-Code Analysis to Help End-user Programmers Create Dependable Software”. There was a special issue of the Journal of Systems and Software devoted to extended versions of the best papers.

The fifth SCAM workshop was a two-day event located in Budapest, in co-location with the 21st International Conference on Software Maintenance (ICSM 2005), STEP 2005, WSE 2005 and VISSOFT 2005. Eelco Visser from the Universiteit Utrecht gave the keynote speech “Transformations for abstraction”, based on his work on Stratego/XT. The program chairs received 48 regular submissions and were able to select 18 excellent papers, covering a broad range of activities in source code analysis and manipulation. From those papers, two have been selected for publication in this special issue.

The sixth and seventh SCAM workshops will be held respectively in Philadelphia and Paris, still co-located with the IEEE International Conference on Software Maintenance (ICSM 2005). We hope that you will appreciate the paper selection in this issue and we also hope to see you in Philadelphia and Paris.

3. Selected papers

The two papers included in this special issue are representative of the works that fall into the area of interest of SCAM, ranging from theoretical code analysis to practical code manipulation techniques. The paper by Binkley et al., entitled “A Formalisation of the Relationship between Forms of Program Slicing”,

The problem of giving a unifying formal framework for the classification and comparison of variants of program slicing. On the other hand, the paper by Zhang et al., “Using Source Transformation to Test and Model Check Implicit-Invocation Systems”, represents a practical application of program transformations, aimed at supporting the verification and validation of programs developed according to the publish–subscribe architecture.

Program slicing has been the subject of research for several decades and a lot of variants, alternative definitions and different algorithms for computing program slices have been investigated. Relating the various forms of slicing with each other has thus become of crucial importance. The paper by Binkley et al. deals with such meta-level, by providing a formal framework for the comparison of slicing techniques, in terms of two inter-technique relationships, subsumption and ranking. A slicing technique subsumes another technique if it produces slices that are also slices according to the second one. On the other hand, ranking characterizes the size of slices, by determining whether the slices produced by a technique are necessarily smaller than those produced by another technique. Clearly, the goal is to define techniques that can produce minimal slices. During the workshop, the discussion focused on the relevance of minimal slices for practical purposes. It was noticed that the ability to recognize minimal slices is important for assessing the level of precision reached, for conducting impact analysis with no false positives and for knowing that
everything included in a slice is really needed. This might be computable for proper subsets of existing programming languages.

Implicit-invocation (or publish–subscribe) systems are particularly challenging for verification and validation methods, since they intrinsically involve concurrent, non-deterministic execution of components. The complexity of such systems demands an integrated approach for assessing their correctness. Testing can exercise them and validate selected executions. Model checking can verify that they obey given properties. Zhang et al. propose an approach that supports both verification and validation of these systems. At the core of their method is a special purpose language, designed for the specification of implicit-invocation systems. Transformation of this language into Turing Plus produces a fully executable system that can be tested. Transformation into an SMV model allows model checking. Program transformation and code manipulation were substantially represented at SCAM 2005, with two technical sessions, as well as the keynote presentation, devoted to these topics. Among the others, one of the subjects of discussion was the difficulty of reusing and composing existing transformation tools for different purposes and in different contexts. On one hand, the domain and the tools are not mature enough, and there are no stable schemas, interfaces and domain boundaries. On the other hand, there are many opportunities for useful combinations, especially if based on serialized intermediate representations.

We are grateful to the authors of the papers for their time and skill in preparing these extended versions of their papers and to the reviewers for their time and expertise in providing timely reviews of all the papers submitted. We hope that you will enjoy and benefit from reading the two papers in this issue, finding stimulating new ideas for the analysis and manipulation of the source code.

Giuliano Antoniol received his degree in electronic engineering from the Universita’ di Padova in 1982. From 1983 to 1987 he worked at the SIP-Italian Telecommunication firm. In 1987 he joined the Speech Recognition and Synthesis Group as Senior Researcher in the Artificial Intelligence Division. From 1994 to 1999 he led the PURE Project team. Since February 2000 he has been Associate Professor at the University of Sannio – Faculty of Engineering – Sabbatical at the École Polytechnique de Montréal, Montréal, Canada. His current research interests include software evolution, software quality, empirical studies and software testing.

Jens Krinke is assistant professor for software technology at FernUniversität/University Hagen, Germany. He received his Ph.D. in 2003 from the University of Passau; his Ph.D. advisors were Gregor Snelting and Tim Teitelbaum (Cornell/Grammatech). His research focuses on the fields of program slicing, clone detection, aspect mining and distant learning. He has developed the VALSOFT Slicing System for validation of software systems and was one of the main contributors to the Praktomat system, a unique system for providing web-based programming courses.

Paolo Tonella is a senior researcher at ITC-irst, Trento, Italy. He received his Ph.D. degree in software engineering from the University of Padova in 1999, with the thesis “Code Analysis in Support to Software Maintenance”. Since 1994 he has been a researcher of the Software Engineering group at ITC-irst. He is the author of “Reverse Engineering of Object Oriented Code”, Springer, 2005. His current research interests include reverse engineering, aspect oriented programming, empirical studies, Web applications and testing.

Giuliano Antoniol
École Polytechnique de Montréal, Montréal, Canada
E-mail address: antoniol@ieee.org.

Jens Krinke
FernUniversität in Hagen, Germany
E-mail address: jens.krinke@fernuni-hagen.de.