College of Engineering

Guest Editors' Introduction to the Special Section from the International Conference on Software Maintenance and Evolution

David Binkley, Rainer Koschke, and Spiros Mancoridis, *Senior Member*, *IEEE*

SOFTWARE maintenance and evolution are relevant to users, engineers, and researchers who come into contact with software beyond Version 1. The International Conference on Software Maintenance and Evolution (ICSM) is the premiere forum for software maintenance researchers and practitioners to examine, discuss, and exchange ideas regarding the key issues facing the software maintenance community. During the conference, participants from academia, government, and industry share ideas and experiences solving critical software maintenance problems.

ICSM 2006 was held in Philadelphia on 24-27 September 2006 in cooperation with several colocated workshops. These included the Eighth IEEE International Symposium on Web Site Evolution (WSE), the Sixth IEEE International Workshop on Source Code Analysis and Manipulation (SCAM), the Second International IEEE Workshop on Software Evolvability, and the Second International Workshop on Predictive Models of Modern Industrial Software Engineering.

ICSM 2006's technical program was anchored by 45 papers selected from 147 submissions. The program also included keynote addresses from three distinguished speakers: Patrick Lardieri, David Notkin, and Richard Stallman. Of the 45 papers, seven were invited to this special issue with five passing the rigorous *TSE* review process.

These five selected papers include two that discuss frameworks and three that present new techniques. The frameworks support the creation of language-independent program analyses and evolvable test suites. Two of the techniques consider source-code reengineering and the final one the challenging problem of making live updates to a software system while it is running.

The first of the two framework papers, "An Extensible Metamodel for Program Analysis" by D. Strein, R. Lincke, J. Lundberg, and W. Löwe, describes a language-independent framework for building new analyses. The resulting architecture supports the construction of specific analyses (e.g., refactorings) and is easily extensible through the addition of new front ends to support new languages. This flexibility and the use of loose coupling between components of the existing framework support the easy integration of new components. The paper uses the implementation of the tools VIZZANALYZER and XDEVELOP as a proof of concept. Looking forward, further work includes empirical study (e.g., considering running time and memory consumption) and the incorporation of dynamic analysis into the framework (e.g., to support debuggers and profilers). This paper was published in the September 2007 issue of *TSE*; however, the abstract of this paper is included in this special section. The paper can be found in the Computer Society Digital Library at http://computer.org/tse/archives.htm.

In the second framework paper, "On the Detection of Test Smells: A Metrics-Based Approach for General Fixture and Eager Test" by B. Van Rompaey, B. Du Bois, S. Demeyer, and M. Rieger, a framework for evaluating the evolvability of white box tests suites alongside the program is presented. The goal of this approach is to avoid the (often significant) cost incurred when test cases must be coevolved with the program. The framework accomplishes this by describing a collection of *test smells*. Akin to code smells, test smells allow the concrete expression of what a good evolvable test is by exploiting the principles that underlie white box testing. The paper proposes a formal description of test smells by means of metric predictors, empirically evaluated for two example test smells. Looking forward, future work will consider the interplay between test smells and frequently changing test cases. Understanding how test smells emerge and grow should help in proactively detecting current and future smells.

The third and fourth papers address reengineering, a problem into which significant software evolution energy is invested. The first of these two papers, "API-Evolution Support with Diff-CatchUp" by Z. Xing and E. Stroulia, addresses the API evolution problem. In short, reusable components, and thus their APIs, must evolve in response to client needs for improved functionality, quality, and generality. Applications using an API which evolve independently can thus "break." To tackle the API-evolution problem, the paper presents a technique and a tool that automatically recognizes API changes in a reused component and proposes plausible fixes based on working examples of the framework code base. Looking forward, planned extensions to the work

- *D. Binkley is with the Computer Science Department, Loyola College in Maryland, 4501 North Charles Street, Baltimore, MD 21210-2699. E-mail: binkley@cs.loyola.edu.*
- *R. Koschke is with the Universität Bremen, Fachbereich 03, Postfach 33 04 40, 28334 Bremen, Germany. E-mail: koschke@informatik.uni-bremen.de.*
- *S. Manicordis is with the Department of Computer Science, College of Engineering, Drexel University, University Crossings Room 139, 3141 Chestnut Street, Philadelphia, PA 19104. E-mail: spiros@drexel.edu.*

For information on obtaining reprints of this article, please send e-mail to: tse@computer.org.

include automatically bringing about some types of changes (e.g., simple identifier renamings) and a user study to evaluate the usability of the tool.

Another look at reengineering is presented in "Improving the Usability of e-Commerce Applications Using Business Processes" by Y. Zou, Q. Zhang, and X. Zhao. This work aims to improve the experience of those interacting with e-commerce applications via a Web page. Users are often overwhelmed by the enormous functionality made available to them by these applications. Knowledge embedded in business process definitions is used to reengineer the interface of existing e-commerce applications. The improved application provides contextual information for fulfilling each business task and also guides users through the various tasks of a business process in a step-by-step fashion. A usability study demonstrates the effectiveness of the reengineered applications. Looking forward, improvement would come from replacing the current semi-automated approach with one that can automatically establish key bindings between tasks and user-interface components.

The final technique, considered in "Tranquillity: A Low Disruptive Alternative to Quiescence for Ensuring Safe Dynamic Updates" by Y. Vandewoude, P. Ebraert, Y. Berbers, T. D'Hondt, addresses the challenging problem of *online* software update. The crux of the problem is placing a system in a consistent state before and after a runtime change to the code. The paper illustrates how the new notion of tranquillity can replace the stronger notion of quiescence as a necessary and sufficient condition for safe runtime changes. Tranquillity is easier to obtain and less disruptive for the running application. An implementation on a component middleware platform is used to experimentally verify the practical applicability of the approach. Looking forward, further empirical study of tranquillity's effectiveness might consider, for example, how often tranquillity must fall back on quiescence.

This issue is a result of the tremendous effort of many people. The guest editors would like to thank Hausi Muller for his guidance in the reviewing processes, the authors, and the reviewers for their hard work. We also appreciate the excellent support and cooperation provided by the IEEE Computer Society staff. We hope that you find the enclosed papers useful and interesting and that they will inspire you to attend future ICSMs.

David Binkley
Rainer Koschke
Spiros Mancoridis
Guest Editors

David Binkley received the doctorate degree from the University of Wisconsin in 1991, at which time he joined the faculty at Loyola College in Maryland, where he is a professor of computer science at Loyola College in Maryland. From 1993 to 2000, Dr. Binkley worked as a faculty researcher at the National Institute of Standards and Technology (NIST). During this time, he was part of the team that worked on the C program slicer Unravel and also considered the problems of creating high integrity (safety critical) software systems and software engineering technology transfer. Dr. Binkley's present US National Science Foundation funded research focuses on improving semantics-base software engineering tools. Recently, he has also been involved in a seven school collaborative project aimed at increasing the representation of undergraduate women and minorities in computer science. In 2006, he cochaired the International Conference on Software maintenance and, in 2002, he was the general chair of the International Workshop on Source Code Analysis and Manipulation and a year later served as the program cochair for that conference.

Rainer Koschke received the degree in computer science from the University of Stuttgart and received the doctoral degree in computer science from the University of Stuttgart, Germany. He is a professor of software engineering at the University of Bremen, Germany. His research interests are primarily in the fields of software engineering and program analyses. His current research includes architecture recovery, feature location, program analyses, clone detection, and reverse engineering.

Spiros Mancoridis received the PhD dgree in computer science from the University of Toronto in 1996 and then joined Drexel University as an assistant professor. He was promoted to associate professor with tenure in 2001 and to full professor in 2007. His research area is software engineering with an emphasis on reverse engineering, code analysis, software security, and software forensics. In 1998, he was received a CAREER Award from the US National Science Foundation. Since then he has received numerous grants from DARPA, NSF, Department of Justice, Department of Defense, Sun Microsystems, AT&T Labs Research, and Lockheed Martin. Professor Mancoridis was the general chair of the IEEE International Conference on Software Maintenance in 2006, the program committee cochair for the Genetic and Evolutionary Computing Conference in 2005, and was on the program committees of several IEEE and AAAI conferences, including WCRE, CSMR, ICSM, ICPC, SCAM, and GECCO. He is a senior member of the IEEE. He is the founder and director of the Drexel University Software Engineering Research Group and has been the undergraduate and graduate director of the Software Engineering degree programs since 2001. He has authored more than 55 scientific publications and has graduated two PhD students and numerous MSc students.