JOURNAL OF SOFTWARE MAINTENANCE AND EVOLUTION: RESEARCH AND PRACTICE
J. Softw. Maint. Evol.: Res. Pract. 2010; **22**:141–142
Published online 8 December 2009 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/smr.420

Editorial

Introduction to the Special Issue on Reverse Engineering (WCRE 2008)

Welcome to the special issue of the 15th Working Conference on Reverse Engineering (WCRE 2008), held in Antwerp, Belgium from 15–18 October, 2008!

Reverse engineering aims at obtaining high-level representations of software systems from existing low-level representations, such as binaries, source code, execution traces or historical development activities. Reverse engineering methods and technologies play an important role in many software engineering tasks. Quite often reverse engineering is the only way to get an understanding of large and complex software systems when commencing important activities, such as bug fixing, software adaptation or maintenance, and system re-engineering or migration.

Each year, researchers gather at WCRE to present the latest advances in reverse engineering, with contributions in the areas of binary reverse engineering, static analysis, dynamic analysis, software repository mining, data reverse engineering, visualizations, program comprehension, code clone detection and software evolution. WCRE provides a platform for not only presenting, but also actively discussing both the issues and the opportunities that our community faces. As our community extends over the boundaries of the academic world, many of the issues come from software engineering professionals. This results in high-quality practical research with strong industrial case studies. While still a relatively young field, the reverse engineering community is thriving and as a result there is now a substantial body of work that has been conducted in the area with important results.

The invited papers for this special issue are revised and expanded versions of the 20 full papers accepted and presented at WCRE 2008. Based on WCRE review reports, presentations and discussions during the conference, we selected five papers as candidates for this special issue. These papers were subsequently extended with new material by the authors and subjected to two additional rounds of reviewing, resulting in four papers that are presented in this special issue.

We hope that readers will gain useful insights into the domain of reverse engineering through the papers in this issue. We would like to extend our gratitude to all the authors who submitted papers to WCRE, to the members of the program committee and their additional reviewers for providing valuable feedback on the papers on time. Also, many thanks to the reviewers who reviewed the papers for this special issue. Their constructive feedback helped shape the papers in this issue. Our thanks also go to Aniello Cimitile, editor of the Journal of Software Maintenance and Evolution, for hosting this special issue.

We briefly introduce the papers in this issue. The papers cover several interesting areas within Reverse Engineering.

Mining Software Repositories

Historical code changes contain valuable information that help developers when performing new changes. In the paper 'Recommending Change Clusters to Support Software Investigation: An Empirical Study', Robillard and Dagenais empirically study the benefits of historical changes using data from seven open source projects spanning 17 years of development. They find that only 13% of these change sets provide useful information for developers. A careful analysis shows that the complex and not well-structured nature of change tasks limits the usefulness of historical change sets.

Detection of Software Clones

Often developers reuse code fragments by copying and pasting them with slight modifications. The detection and tracking of these code clones ensure the successful evolution of long-lived large projects. In the paper 'Near-miss function clones in open source software: an empirical study', Roy and Cordy propose a new hybrid clone detection technique. The technique combines the strength and overcomes the limitations of text-based and AST-based clone detection techniques, while relying on the application of code transformations to better detect clones. Roy and Cordy demonstrate their technique through a large empirical study involving over 20 open-source systems. The authors manually validate their clones, in turn creating a large benchmark for the evaluation of clone detection tools.

Software Visualization

Software visualizations help improve our understanding of complex software systems. However, there is no 'natural' mapping of software to a two-dimensional space as software has no shape–leading to layouts in which position and distance have no meaning. In the paper 'Software Cartography: Thematic Software Visualization with Consistent Layout' Kuhn and colleagues propose an approach for consistent layout in which the position of a software artifact reflects its vocabulary, and distance corresponds to similarity of vocabulary. The authors illustrate the benefits of using their approach to study the evolution of four large and long-lived open-source projects.

Migration of Legacy Systems

The data model plays a central role in legacy information systems. The translation of this data model is one of the most critical decisions when migrating legacy information systems to new platforms or languages (e.g., Java). In the paper 'Migrating legacy data structures based on variable overlay to Java' Ceccato and colleagues document their experience migrating a large banking system with around ten million lines of code. The authors discuss the challenges associated with translating the data model and present several approaches to deal with these challenges.

AHMED E. HASSAN
Queen's University, Ont., Canada

ANDY ZAIDMAN
Delft University of Technology, The Netherlands

MASSIMILIANO DI PENTA
University of Sannio, Italy