Issue Overview

Analyzing the Evolution of Large-Scale Software

This special issue reports on approaches that address the issue of analyzing and visualizing the evolution of large-scale software. These approaches were discussed during the IEEE Workshop on Evolution of Large-Scale Industrial Software Evolution (ELISA), which was co-located with the International Conference on Software Maintenance in Amsterdam in September 2003. After a thorough review process, two articles have been selected for this special issue. The techniques proposed and illustrated in these articles allow us to gain a deeper insight into the nature of the evolution of large-scale software.

THE WORKSHOP

The IEEE ELISA workshop was co-located with the International Conference on Software Maintenance in Amsterdam in September 2003. It was organized by Tom Mens, Juan F. Ramil, Michael W. Godfrey and Brian Down. The main theme of the workshop was planned to be the evolution of large-scale industrial software. Wider themes, such as the evolution of open source software, were also well represented in the workshop submissions. The goal was to achieve a deeper and wider insight into the problems posed by the evolution of software and the possible technological and managerial solutions.

This goal is particularly relevant in current-day software practice, since size issues compound the challenges to achieving disciplined software evolution. Industrial and open-source software generally consist of numerous software artifacts that need to be evolved in a harmonious fashion, change requests that arrive more quickly than can be reasonably implemented, and the involvement of several teams implementing the evolution. Typical examples of large-scale software include air traffic control, popular PC operating systems, and telephone switch software. The challenges posed by the continual evolution of these and similar software clearly challenge the current state-of-the-art.

In total, the ELISA workshop received 19 workshop submissions, nine of which were short papers and 10 were full papers; 15 of the submissions were classified as research papers, the remaining four as experience papers. One submission was rejected because it was considered as being of insufficient quality.
The workshop itself was very well attended. There were more than 30 participants, coming from more than 10 different countries. The workshop was subdivided into four sessions, each focusing on a particular subarea in the research domain of software evolution.

- The first session, chaired by Tom Mens, was dedicated to software evolution benchmarks, which is one of the research themes of the ESF Scientific Network RELEASE. Susan Elliott Sim gave a keynote lecture sharing her expertise in the wider 'benchmark' topic and suggesting how it could be applied to software evolution. Next, Serge Demeyer presented his view on how to develop a benchmark for software refactoring.
- The second session was chaired by Brian Down of Sun Microsystems Canada. It was devoted to experience reports on industrial case studies. He and two other industrial representatives presented their experience on how to migrate or evolve legacy software. The two selected papers that appear in this special issue were presented during this session.
- The third session, chaired by Juan F. Ramil, focused on techniques and empirical studies to understand the evolution of large software.
- The fourth session, chaired by Michael W. Godfrey, focused on taxonomies and classifications. The first presentation proposed a taxonomy of clone detection techniques, and the second proposed a taxonomy of model and metamodel co-evolution.

Some conclusions from the workshop include the following. Open source offers much freer access to the historical records than any commercial or industrial organization. This is a tremendous advantage that must be exploited. Also, studies in the area of software evolution need to be more rigorous for results to be more easily compared and contrasted. This is where benchmarks, taxonomies, and metamodels can help, but there is a clear need for standards in this area.

SELECTED PAPERS FOR THIS SPECIAL ISSUE

After the workshop, the authors of eight of the papers were invited to submit revised versions for consideration for publication in this special issue. After several rounds of thorough independent reviewing, two of the papers were ultimately selected. These two papers were substantially revised with respect to the original workshop submission.

The first selected article ‘Using software trails to rebuild the evolution of software’ by Daniel M. German presents a method to recover and analyze the evolution of a software system using its software trails: information left behind by the contributors to the development process of the product, such as mailing lists, Web sites, version control logs, software releases, documentation, and the source code. The method is validated by recovering a detailed view of the evolution history of Ximian Evolution, a 0.5 million SLOC open source e-mail application.

The second article, ‘Visualizing feature evolution of large-scale software based on problem and modification report data’ by Michael Fischer and Harald Gall, presents an approach to analyzing feature evolution in software systems. In particular, this article examines hidden dependencies between structurally unrelated, but over time logically coupled, features. A visualization of the detected dependencies between features can indicate locations of so-called design erosion in the architectural evolution of a software system. The approach is illustrated by its application to Mozilla, a 3.7 million SLOC open source Web browser.
Although the detailed goals and approaches of the two articles are different, they share many commonalities. Both articles use a large-scale open-source software system as a case study. Both approaches use and correlate evolution history information in a variety of forms and formats: version control logs, release histories, source code information, problem reports, modification reports, bug reports, mailing lists, and so on. Both papers use quantitative data to visualize the evolution of software.

Finally, the techniques proposed in both articles bring us one step closer to the initial workshop goals, since they allow us to gain a deeper insight into the nature of the evolution of large-scale software. Of course, much more remains to be done in this field, if the software evolution phenomenon is one day to be satisfactorily understood and managed.

ACKNOWLEDGEMENTS

We wish to thank all the workshop participants for their contributions, and the authors of the selected papers for their efforts in making substantial revisions. We also express our gratitude to all the reviewers that took the time to provide detailed comments on the papers submitted to the workshop and this special issue: Stephen Cook, Serge Demeyer, Massimiliano Di Penta, Jean-Marie Favre, Guenter Kniesel, Kostas Kontogiannis, Robin Laney, Michele Lanza, Manny Lehman, Kim Mens, Vaclav Rajlich, Harvey Siy, Arie Van Deursen, Jilles van Gurp and Michel Wermelinger.

TOM MENS
JUAN F. RAMIL
MICHAEL W. GODFREY