Empir Software Eng (2009) 14:257–261
DOI 10.1007/s10664-009-9110-3

EDITORIAL

Guest editors introduction: special issue on mining software repositories

Stephan Diehl · Harald C. Gall · Ahmed E. Hassan

Published online: 5 May 2009

The Mining Software Repositories (MSR) field analyzes and cross-links the rich data available in software repositories to uncover interesting and actionable information about software systems and projects. Examples of software repositories are:

- **Historical repositories** such as source control repositories, bug repositories, and archived communications record information about the evolution and progress of a project.
- **Run-time repositories** such as deployment logs contain information about the execution and the usage of an application at a single or multiple deployment sites.
- **Code repositories** such as Sourceforge.net and Google code contain the source code of various applications developed by several developers.

Software repositories are commonly used in practice as record-keeping repositories and are rarely used to support decision-making processes. For instance, historical repositories are used to track the history of a bug or a feature, but are not commonly used to determine the expected resolution time of an open bug based on the resolution time of previously-closed bugs. Mining these historical, run-time and code repositories, we can uncover useful and important patterns and information. Developers can use historical information to propagate changes to related artifacts, instead of only using static or dynamic code dependencies which may fail to capture important dependencies. For example, a change to the code which writes data to a file may require changes to the code which reads data from the file, although there exists no traditional (e.g., data and control flow) dependencies

S. Diehl (✉)
Department for Software Engineering, University of Trier, Trier, Germany
e-mail: diehl@uni-trier.de

H. C. Gall
Department of Informatics, University of Zurich, Zurich, Switzerland
e-mail: gall@ifi.uzh.ch

A. E. Hassan
School of Computing, Queen's University, Kingston, ON, Canada
e-mail: ahmed@cs.queensu.ca

between both pieces of code. As for run-time repositories, they could be used to pinpoint execution anomalies by identifying dominant execution or usage patterns across deployments, and flagging deviations from these patterns. Code repositories could be used to identify dominant and correct library usage patterns by mining the usage of a library across many projects.

The MSR field is rapidly taking a central and important role in supporting software development practice and software engineering research. Research is now proceeding to uncover the ways in which mining these repositories can help *support the maintenance of software systems*, *improve software design/reuse*, and *empirically validate novel ideas and techniques*. By transforming these repositories from static record-keeping ones into active ones, we can guide decision processes in modern software projects and support software engineering research activities.

In an effort to bring together the practitioners and researchers working in this important and emerging field, the first International Workshop on Mining Software Repositories (MSR) was held at the International Conference on Software Engineering (ICSE), the flagship conference for Software Engineering in 2004. After four successful years as ICSE's largest workshop, MSR became a Working Conference in 2008. As a Working Conference, MSR recognizes the maturity and breadth of the work in the MSR field, while still encouraging free-form open discussions about the MSR field. An MSR challenge is also held yearly so researchers can compare their techniques towards a common problem or project.

MSR is a young and emerging field which continues to attract a large amount of interest within software engineering. For the latest information about MSR, please refer to http://www.msrconf.org.

The invited papers for this special issue are revised and expanded versions of papers accepted and presented at MSR 2005 and 2006. The expanded papers have undergone two rounds of full reviewing. They present several novel and interesting approaches to derive useful information from software repositories. We briefly introduce the papers.

1 Software Evolution in the Large

Most software evolution studies focus on the evolution of a single product developed by a coordinated team. However, software systems are nowadays commonly composed of a large set of applications and libraries. Many of these libraries and applications are developed by unrelated parties with their own goals and practices. Studying the evolution of these complex ecosystems of applications sheds light on the complexity of modern software development and project management practices. In this issue, Gonzalez-Barahona and his colleagues take a different approach to studying software evolution. They study software evolution in the large, the evolution of compilations of software, composed of many different individual software applications that are combined together to form a system, in contrast to the traditional view of software evolution in the small, the evolution of a single application. Through a case study which uses nine years of history for a large open source (libre) software distribution, the Debian GNU/Linux, Gonzalez-Barahona and his colleagues show remarkable findings about the evolution of software ecosystems and

the interdependencies between their different parts. For instance, one of the findings that stands out is that the size of Debian, measured as the number of applications (packages) or as the lines of code, doubles approximately every two years. Considering the size of Debian of about 300 MLOC spread over 10,000 packages in 2007, this rapid growth poses significant challenges for the current Debian team and signals many problems for the future evolution of Debian.

2 Bug-Fix Patterns

Practitioners and researchers strive to create bug-free software. However, little is known about the most common kinds of software bugs, and whether the rate of occurrence of these bug kinds is consistent across systems. Using this information, research efforts can focus on reducing the most common kinds of bugs. Of course, this only makes sense if the most common bug types are similar across a broad range of systems. In this issue, Pan and his colleagues mine the source control repositories of seven open source Java projects (Eclipse, Columba, JEdit, Scarab, ArgoUML, Lucene, and MegaMek) to identify bug-fix patterns. They identify 27 bug-fix patterns, which are amenable to automated detection. These patterns cover 45.7% to 63.3% of all fixed bugs throughout the lifetime of these projects. An example pattern is the *change-in-if-conditional* which appears in 5-18% of all bug fixes across the studied projects. Pan and his colleagues note that these patterns are consistent across projects independent of the type of software being produced. Such findings reinforce earlier research findings on single systems and help direct research efforts for bug prevention and detection.

3 Visual Analysis of Repositories

Most source control systems are primarily designed to support the task of archiving software and maintaining code consistency during development. These systems provide limited functionality for navigating and exploring the raw data stored in their repositories. Voinea and Telea present a visual and customizable framework for the rapid analysis of the large amount of complex raw data in software repositories. The framework architecture enables the rapid customization for different repository standards and formats (e.g., SVN versus CSV) through the use of a network of scripted components connected by a dataflow engine to a central fact database. Voinea and Telea demonstrate the usefulness of their visualizations by studying repository data for five projects (three industry-size Open Source projects (ArgoUML, PostgreSQL, KDE Koffice), one academic system (mCRL2) and one commercial system (MagnaView)), and by verifying many of their visual findings using developers form these projects.

Acknowledgments We are grateful to the continuous support and encouragement offered by the Editorial board for the Journal of Empirical Software Engineering and by the Editor-in-Chief Lionel Briand. This issue is the result of a great deal of effort by the reviewers, authors, and attendees of MSR 2005 and 2006. We thank the authors for keeping up with the review schedule and the reviewers for their detailed and constructive comments which helped shape the papers.

Stephan Diehl is a full professor for computer science at the University of Trier, Germany. His research interests include software engineering and information visualization, and in particular the intersection of both areas, namely software visualization.

He was the program chair of the ACM Symposium on Software Visualization in 2005 and 2006 and the International Workshop on Mining Software Repositories (MSR) in 2006 and 2007.

Contact him at diehl@uni-trier.de; http://www.st.uni-trier.de/~diehl/.

Harald C. Gall is a professor of software engineering in the Department of Informatics at the University of Zurich, Switzerland. His research interests include software engineering, focusing on software evolution, software quality analysis, software architecture, reengineering, collaborative software engineering, and service centric software systems.

He was the program chair of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC-FSE) in 2005, MSR in 2006 and 2007. He is the program co-chair of the International Conference on Software Engineering (ICSE) 2011.

Contact him at gall@ifi.uzh.ch; http://seal.ifi.uzh.ch/gall.

Ahmed E. Hassan is the NSERC/Research In Motion (RIM) Chair in Software Engineering at the School of Computing, Queen's University, Canada. His research interests include Mining Software Repositories (MSR) and Ultra Large Scale (ULS) systems. He spent the early part of his career helping architect the Blackberry wireless platform.

He spearheaded the organization and creation of the MSR workshop series and its associated research community. He co-edited a special issue of the IEEE Transaction on Software Engineering (TSE) on MSR in 2005. He is the program chair for the Working Conference for Reverse Engineering (WCRE) 2008.

Contact him at ahmed@cs.queensu.ca; http://www.cs.queensu.ca/~ahmed.