Report on MSR 2005: International Workshop on Mining Software Repositories

Ahmed E. Hassan and Richard C. Holt
Software Architecture Group (SWAG)
School of Computer Science
University of Waterloo
Waterloo, Canada
{aeehassa, holt}@plg.uwaterloo.ca

Stephan Diehl
Computer Science
Catholic University Eichstätt
Eichstätt, Germany
diehl@acm.org

Abstract

A one-day workshop on the topic of mining software repositories was held at ICSE 2005 in St. Louis, Missouri. Researchers and practitioners in this field try to transform static record keeping software repositories to active ones. These repositories enable researchers to gain empirically based understanding of software development, while software practitioners use them to predict and plan various aspects of their project.

After the surprisingly high number of submissions and attendance of the first iteration of the MSR workshop in 2004, it turned out that MSR 2005 even attracted more people. We finally had $\boxed{\text{XX}}$ registered participants for the workshop. Due to the high quality of submitted papers, we decided to squeeze 22 papers into one day by having 11 of the papers be presented as Lightning talks – an euphemism for 5 minutes presentations. There was a very vivid combined discussions and demo session after these talks. When asking the audience at the end of the workshop, they indicated that they actually liked the Lightning talks session.

This report includes an overview of the presentations made during these sessions and a summary of the issues raised throughout the workshop.

Introduction

Software repositories such as source control systems, archived communications between project personnel, and defect tracking systems are used to help manage the progress of software projects. Software practitioners and researchers are beginning to recognize the potential benefit of mining this information to support the maintenance of software systems, improve software design/reuse, and empirically validate novel ideas and techniques. Research is now proceeding to uncover the ways in which mining these repositories can help to understand software development, to support predictions about software development, and to plan various aspects of software projects.

Scope and Topics of Interest

We sought position papers that address issues along the general themes, including but not limited to the following:

- Approaches to study the quality of the mined data along with guidelines to ensure the quality of the recovered data
- Proposals for exchange formats, meta-models, and infrastructure tools to facilitate the sharing of extracted data and to encourage reuse and repeatability
- Models for social and development processes that occur in large software development projects
- Search techniques to assist developers in finding suitable components for reuse
- Techniques to model reliability and defect occurrences
- Analysis of change patterns to assist in future development
- Case studies on extracting data from repositories of large long lived projects
- Suggestions for benchmarks, consisting of large software repositories, to be shared among the community

Workshop Format

We received 38 papers from 14 countries. Papers were reviewed by the workshop's program committee in terms of their relevance to the aims of the workshop and their technical content. Accepted papers were posted on the workshop's web site prior to the workshop at:

http://msr.uwaterloo.ca

The workshop program was broken into five sessions: 4 with regular talks and one combined lightning talks and demo session.

Regular talks were 15 minutes with one clarification question. At the end of each session we had an open discussion of all papers in that session. In contrast the lightning talks were only 5 minutes long with no clarification questions. The lightning talks were followed by a one hour walkaround demos and discussion session.

Workshop Sessions

Session 1: Evolution and Change Patterns

The papers in this session investigated how changes occur in evolving software systems and how to deal with the huge amounts of data. Neamtiu et al. used abstract syntax trees to identify changes and found that change increases with depth. Williams et al. recovered system specific function usage patterns from the change history. In a case study Fischer et al. looked at the operating system BSD and its offspring to see how the evolution of product families differs from that of single programs. Finally, by looking at the evolution of code clones Kim and Notkin characterized different kinds of clones and found that there are actually good clones.

Session 2: Defect Analysis

In general defect analysis is concerned where bugs come from and where they go and how well we can predict them. Sliwerski and Zimmermann tried to identify changes that induce later fixes, i.e. a change to correct a previous buggy change. In particular they found, that such changes more frequently occur on Fridays. Görg and Weißgerber developed a method to automatically detect incomplete refactorings. Some of the incomplete refactorings that they found lead to compilable programs, but do not preserve the semantics.

Session 3: Education

While most of the other papers were looking at archives of medium to large open source programs, the papers in this session applied mining to student programs to allow teachers to shadow students' progress. Spacco et al. investigated how warnings of different static analyzes predict exceptions raised when testing the program. Mierle et al. extracted various quantitative measures from 200 CVS archives of student projects and tried to relate these with grades. Surprisingly, no predictors stronger than simple lines-of-code were found.

Session 4: Lightning Talks

The talks in this session were divided into four themes:

4a) Text Mining Ohba and Gondow suggested to mine for concept keywords in identifiers to relate bug reports and source code. In a case study Ying et al. found different kinds of Eclipse comments that start with `TODO` and argued that these and other source code comments provide important information: "Someone left a note for you in the code". Hayes et al. undertook a pilot study to examine the impact of analyst decisions on the final outcome of the text mining process.

4b Software Changes and Evolution Kim et al. developed a taxonomy of function signature change kinds and analyzed 8 open source software systems to see how often each of these change kinds occurred.

Ratzinger et al. identified two bad change smells, i.e. bad practices of how to change code, and showed in a case study that these can be used to find bad smells in the source code.

In their case study Antoniol et al. apply two techniques from signal processing, namely Linear Predictive Coding and Cepstral analysis, to identify files whose sizes similarly evolve over time.

4c) Process and Collaboration VanHilst et al. argue that mining software repositories provides useful process metrics without adding overhead to the process ifself.

Huang and Liu applied social network analysis to divide modules into conceptual kernel and non-kernel modules, as well as developers into core and none-core teams.

Huang and Liu received the Lightning award for the best presentation in this session.

4d) Taxonomies and Formal Representations Two different taxonomies were proposed: Kagdi et al. proposed a taxonomy based on technical aspects, e.g. the kinds and granularity of mining, whereas German et al. also considered the context, i.e. who applies the mining and why is it used. Hindle and German designed a query language to extract information from software repositories. Their language is based on a formal model of repositories consisting of four entities: authors, modification request, revisions, and files.

Session 5: Integration and Collaboration

Robles and González-Barahona combined various sources of data to map the different identities used by a developer in one or more open source projects to a single person.

Ohira et al. developed a graph-based tool to analyze and visualize the relationship among projects and developers at SourceForge. They found that about 66 percent of all projects had only one developer.

Conklin et al. reported about a repository for researchers to store and share meta-data (developer names, platforms, licence types, etc.) extracted from general repositories like SourceForge, GNU Savannah and the like.

1 Conclusions

Tools and approaches were presented that help to identify potential bugs, who are the core-team members of a project, or how to reuse a specific part of program code. The techniques applied ranged from classical data mining to signal processing. In the discussions the question of what are the underlying heuristics and are they valid, was raised several times. Also privacy issues were discussed a lot: although the archives are publicly available, should we make our mining results also publicly available as they often provide condensed information about individual developers.

By looking at the data stored in software archives researchers found that there are good code clones, that students who put spaces after commas get better grades and that programmers should not work on Fridays. As more of these tools become available, they will enable us to find out whether these findings are true for our own projects or students.

Finally, there are plans to turn MSR into a two days workshop and include a posters session and a challenge task to enable comparison of the various approaches.