Report on Data-intensive Software Management and Mining

Seung-won Hwang

Department of Computer Science and Engineering
POSTECH, Korea
swhwang@postech.ac.kr

1. MOTIVATION

With the increase of publicly available software repositories, we face new challenges related to managing and mining large-scale software repositories. These challenges naturally call for novel research contributions from large-scale data management and mining researchers. Software repositories include multi-version source code, bug reports, and development history, and managing these repositories has been recognized as an important task in software design, development, and maintenance processes.

Due to these emerging challenges, many recent publications in major software engineering conferences and journals, such as ICSE, FSE, or TSE, have been contributed by database and mining researchers. To showcase some such efforts, mining approaches have been used for a corpus of failing traces, automatically collected from users experienced crashes. This corpus can assist developers to diagnose faults, prioritize handling of those faults, and locate faults in the code. Assuming similar faults have similar traces, mining approaches have been applied to mine similar past traces [5]. Once found, the fault location of the past trace can then be used to predict the location of the given trace. Similarly, developers, looking for related code developed in the past for reference, want an instant search of clones, which is essentially a ranked similarity search problem [4]. These efforts have been reported to increase the productivity of software developers, while enhancing the quality of code produced at the same time. Related efforts have also been reported in major human computer interaction conferences, such as SIGCHI. One example of recent research is Blueprint [2], which developed interfaces integrating instant source code and document searches for developers' code development sessions. Another example is Code Bubbles [1], proposing an interface to help developers effectively navigate code fragments spread all over a large corpus.

As these examples show, we observe emerging data management challenges from software problems that are highly relevant to database researchers.

- **O1: Software as a web-scale repository** A recent study [6] shows that an open-source code corpus can be as large as 420 million lines of source code and most code shares a similar structure, which positively suggests that mining for related code can be effective most of the time. However, this study reports that such analysis can take up to four months of CPU time, which invites effective indexing and mining techniques for large-scale repositories.

- **O2: Software as an evolving repository** However, source code repositories are reported to evolve over time. Such evolution has been noticed in software engineering research, and for evolving repositories, versioned index structures [3] studied in database research may enable efficient retrieval of evolutions.

- **O3: Software as a complex data structure** Many analysis schemes represent a piece of software code using different data structures. A parse tree is one example and a call graph is another. Alternatively, it could be high-dimensional vectors [4]. In all cases, an efficient similarity search is non-trivial and database techniques for complex data retrieval can be highly relevant.

- **O4: Software as a social network** Meanwhile, source code is reported to be a small part of a big underlying repository, where code is interconnected with related documents (bug reports or testing documents) and also people in charge of software segments. This forms a large-scale development social network [1].

[1] http://code.google.com/apis/opensocial/

The aim of the DSMM workshop, co-located with CIKM, was to draw the attention of database researchers to emerging challenges in large-scale software repositories. In particular, key topics discussed in this workshop included:

- Software indexing and search techniques.
- Software pattern mining and management techniques
- Management/mining of large-scale multi-version source code corpora
- Case studies of management/mining of large-scale software

2. WORKSHOP OUTLINE

This workshop was chaired by Audris Muckus (Avaya Labs, USA), Seung-won Hwang (POSTECH, Korea), and Sunghun Kim (HKUST, China). The PC team consisted of 15 reviewers from research labs and universities. Geographically, 10 reviewers were from North America, 3 from Europe and 2 from Asia. Half of the reviewers came from the software engineering community and the rest from the database community. The workshop consisted of three sessions, presented by one invited speaker and six paper presenters, as we discuss in detail below. A full list of speakers can be found at: http://ids.postech.ac.kr/dsmm.

2.1 Invited Session

The keynote speech, entitled "Developing accurate risk models requires mathematics, domain knowledge and common sense, although not necessarily in that order?" was delivered by Brendan Murphy from Microsoft Research, Cambridge. He addressed the challenge **O1** and presented on the difficulty of developing a risk model and getting a set of experienced engineers to accept and use your risk model. He then shared his experiences of developing such a model for the Microsoft Windows operating system and how getting engineers to accept the model is only possible by understanding the limitations of the model developed. His speech provided the participants with insights into how industry software is actually developed and how mathematics may not necessarily solve all problems encountered from industry development processes.

2.2 Research Session I: Software Development Process

This session had three papers on how data management techniques can assist software development processes.

The first presentation, entitled "Tree-pattern-based duplicate code detection", discussed the problem of automatically and accurately finding code clones in program files. In particular, this paper addressed the challenge **O3** and abstracted the problem as tree-pattern mining and clustering. However, comparing every pair can be costly and the authors present an efficient algorithm for this problem. Specifically, clustering enables flexible clone detection, covering four known types of clones: exact, parameter-substituted, structure-substituted, and non-contiguous clones.

The second presentation, entitled "Mining software repositories for bug detection requires accurate techniques of identifying bug-fix revisions", discussed repositories of bugs and their fixes. This talk was related to the challenge **O2** of managing evolving repositories of bug fixes. If a bug can be found the repository and annotated with how it was fixed in the past, we can automatically fix the bug guided by the existing fix. As a preliminary to this ambitious goal, the presenter discussed existing noises in the repository and how to remove them automatically. Specifically, the authors inspected bug-fix revisions of three open source projects (Eclipse, Lucene, and Columba) and categorized noises into 11 patterns. With these patterns identified, noise elimination can be automated.

The third presenter could not present her work, entitled "A case study on Model Driven Data Integration (MDDI) for Data Centric Software Development". According to her paper, her work described MDDI, which is a data integration problem over heterogeneous schemas, and presented challenges related to the problem, using actual historical data collected from universities. This paper was related to the challenge **O3**.

2.3 Research Session II: Large-scale Software Corpus

This session had three papers, related to maintaining and mining software-related corpora.

The first presentation, entitled "Transaction synchronization protocol using XML in client-Server environment", discussed versioned-repositories, such as software repositories that are constantly revised (and thus discussed the challenge **O2**). To keep these repositories consistent, transaction synchronization is an important problem: When clients modify their local database, writing this transaction back to the server and to all other clients requires transaction synchronization. To achieve reliable and efficient transaction synchronization, the presenter described an efficient protocol for transactional synchronization, using XML as a transfer medium. This approach can significantly reduce the cost of data transfer between server and clients, thus increasing efficiency.

The second presentation, entitled "A method of workload compression based on characteristics for index selection", discussed a corpus of database workloads for development-related data that are frequently queried by developers. While managing such a workload provides important hints for optimizing the future queries, keeping all the workloads incurs prohibitive storage overheads. The presenter proposed an effective compression algorithm, which can significantly reduce the storage, without seriously compromising optimization quality for future queries. Such compression can be useful for a large-scale repository. This work discussed the challenge **O1**.

The last presentation, entitled "A targeted web crawling for building malicious javascript collection", raised the interesting question of whether the detection of malicious javascript code can be enhanced with a large corpus of malicious scripts. This problem is important, as malicious javascript frequently serves as a starting point for web-based attacks, in particular cross-site scripting. However, collecting up-to-date repositories of scripts is a non-trivial task. The presenter proposed a web crawler focused on more likely locations of malicious scripts for this task, and showed how this targeted web crawler performs. This talk was related to the challenge of large-scale repositories **O1**, but tackles the problem of "collecting" a repository with focus, namely malicious Javascripts.

3. CONCLUSIONS

DSMM was the first CIKM-associated workshop addressing the challenges of large-scale software repositories. Among the experts gathered in three different areas– program analysis, software engineering, and databases– the following future directions were discussed.

First, these challenges call for interdisciplinary efforts. To encourage such efforts, workshops of an interdisciplinary nature are useful. Though such interdisciplinary workshops have been founded, they are usually co-located at conferences related to one specific domain and attendance tends to biased toward that domain. It is thus still challenging to invite multi-disciplinary papers and presentations to one venue. Other efforts that do not require participants to gather in one location can be explored.

Second, for research of this nature to accumulate, releasing standard repositories, tasks, or benchmark data should be encouraged and rewarded. Having these common data, if successful, may address the first challenge as well– different teams can submit their work on common tasks and publish in journals. The participants expressed shared interest in forums for continuing interdisciplinary discussions on software repository management.

4. REFERENCES

[1] Andrew Bragdon et. al. Code bubbles: a working set-based interface for code understanding and maintenance. In *SIGCHI*, 2010.

[2] Joel Brandt et. al. Example-centric programming: Integrating web search into the development environment. In *SIGCHI*, 2010.

[3] Mark Gabel and Zhendong Su. How unique is source code. In *VLDB*, 2010.

[4] Mu-woong Lee, Jongwon Roh, Seung-won Hwang, and Sunghun Kim. Instant code clone search. In *FSE*, 2010.

[5] Chao Liu and Jiawei Han. Failure proximity: A fault localization-based approach. In *FSE*, 2006.

[6] Rui Zhang and Martin Stradling. The hvtree: a memory hierarchy aware version index. In *FSE*, 2010.