Dagstuhl Seminar "Mining Programs and Processes" – Executive Summary

Abraham Bernstein • Harald Gall • Andreas Zeller

The main goal of the seminar "Mining Programs and Processes" was to create a synergy between researchers of three communities, namely mining software repositories, data mining and machine learning, and empirical software engineering. This goal was only partially met; while we had good response rates from the mining software archives community, we only had little response from the machine learning community. This was due to an unfortunate timing: The Dagstuhl seminar ran at the same time as NIPS 2007, the major machine learning event, and thus had great trouble attracting machine learning researchers. At the time the seminar date was finalized, the NIPS date was not set yet, and when the conflict was discovered, it was too late to change the date of the seminar. We thus lost a number of opportunities for interchange with machine learning ressearchers. Furthermore, one of our organizers (Tao Xie) was unable to come to due Visa issues. Despite these drawbacks, we are convinced that the seminar did generate a deeper understanding of the three communities' research challenges and state-of-the-art works. We thus focused on the interaction between mining software archives and empirical software engineering, with a small influx from the machine learning side.

We had invited a small number of participants to give keynotes that would reach out to different communities; in particular, Katharina Morik showed off the machine learning perspective, and Lionel Briand bridged the gap to empirical software engineering. During the seminar, the participants broke into five working groups, each dedicated to a specific topic. The results of these groups reflect the state of the art, as well as challenges for the future:

Issues of Mining Software Repositories

The main challenge of mining software repositories is to provide solid empirical evidence. The field thus has to adopt established methods and techniques from empirical software engineering, including discussing outcomes that may contradict or confirm existing studies, and also involve into theory building – that is, not only report facts and correlations, but also possible hypotheses for these findings.

What do Developers (really) Need?

This session was one of the most creative ones, in terms of identifying the needs of real developers and which information repositories could provide that would adress these needs. Based on reports from developer studies (most notably Microsoft), the group identified three central needs: (1) code rationales (“why does the code look as it does?”), (2) help in debugging (“where did this value come from?”), and (3) team awareness, i.e. a notion about what their coworkers are doing. (1) and (3) were found to be particularly well addressable by mining team information.

Empirical Studies and Long-Term Objectives

This group discussed what the long-term objectives of mining software repositories could be, and again, this future was to be found in a solid empirical foundation. While the focus of mining software repositories would be in automatic information extraction, empirical software engineering would focus on demonstrating that this information would actually be useful. In the long term, mining software repositories would provide important findings for empirical studies, but the standards in terms of empirical evidence are not to be ignored.

An Infrastructure for Mining Software Repositories

In this session, we discussed what an infrastructure for mining software repositories would have to provide. Desired (or required) features include better composability of individual tools, better engineering and documentation of tools, as well as common formats and models for data exchange.

Dagstuhl Seminar Proceedings 07491
Mining Programs and Processes
http://drops.dagstuhl.de/opus/volltexte/2009/2246

Important recommendations included to make tools widely available (with a citation mechanism to give credit), being able to emit results in simple formats, as well as the usage of Wiki pages such that users could add additional information or tutorials on the tools. Running Mining Challenges in using same/different tools on a common set of published data can foster important comparisons of tools and approaches and thereby enable a deeper foundation for mining.

Understanding Code Changes

This group discussed recurring technical problems in mining version histories, namely parsing and processing individual changes. Challenges include parsing source code that is possibly incomplete or comes in languages where appropriate tools are not available. Also, it is not always straight-forward to isolate the purpose of individual changes, as changes frequently do multiple things at once, or are hard to discriminate into fixes, features, and maintenance improvements. One possible technique is to identify the *dynamic impact* of a change – renamings and refactoring have a local impact at best, whereas bug fixes should show non-local changes.

Summary

While there clearly is a need for further interaction between machine learning and mining software repositories, the seminar made clear progress into the interaction between mining software archives and empirical software engineering; this interaction is also reflected in the respective venues, which more and more adopt mining and empiricism as standard techniques. Also, the seminar showed inspiring directions in mining itself, as outlined above.