Mining the Mind, Minding the Mine

Grand Challenges in Comprehension and Mining

Andrew J. Ko
University of Washington
Seattle, Washington
ajko@uw.edu

ABSTRACT

The program comprehension and mining software repository communities are, in practice, two separate research endeavors. One is concerned with what's happening in a developer's mind, while the other is concerned with what's happening in a team. And yet, implicit in these fields is a common goal to make better software and the common approach of influencing developer decisions. In this keynote, I provide several examples of this overlap, suggesting several grand challenges in comprehension and mining.

CCS CONCEPTS

• **Human-centered computing**; • **Software and its engineering → Software configuration management and version control systems**; **Software maintenance tools**; **Programming teams**;

KEYWORDS

Mining software repositories, program comprehension

ACM Reference Format:
Andrew J. Ko. 2018. Mining the Mind, Minding the Mine: Grand Challenges in Comprehension and Mining. In *ICPC '18: ICPC '18: 26th IEEE/ACM International Confernece on Program Comprehension , May 27–28, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, Article 4, 1 page. https://doi.org/10.1145/3196321.3196324

1 OVERVIEW

Software engineering research is broadly concerned with inventing ways to make better software, faster. In this pursuit, however, we often take wildly different approaches to discovery. Some of us make tools, some of us advance scientific theories, and other build robust theories of verification. This diversity of approaches is a great strength of our field: by exploring multiple ways of advancing software engineering in parallel, we increase the speed of discovery.

However, this diversity of approaches also creates artificial rifts between ideas. Take, for example, two co-located ICSE conferences, the International Conference on Program Comprehension and the International Conference on Mining Software Repositories. On the surface, these two communities appear to study different things. Program comprehension researchers, for example, are interested in what happens in developers' minds, trying to build theories about how developers reason about code, and build tools that leverage these theories to make comprehension faster and more robust. This "micro" view is powerful because it has direct implications for the programming languages and tools we invent to engineer software. The mining software repository community, however, is interested in what can be known by studying repositories, which are traces of entire teams working together to repair, enhance, and maintain systems. This "macro" view is powerful because it promises teams and communities of developers a global, temporal view of quality. These are two distinct phenomena, studied at two different levels, with few shared visions, methods, or tools.

There is, however, common ground between these two fields. Both have the same *goal* of improving software quality. Both also have the same approach to *intervention* by changing what developers believe about their code, and thus changing developer behavior to result in better systems. By focusing on the problem of changing developer behavior, we can see many opportunities for each community to leverage the other's work.

Consider the problem of a developer understanding the architecture of a large software system. While comprehension researchers have long studied how to apply program analysis to this problem, they have generally not applied the methods of software repository mining. For example, in addition to understanding the *current* architecture of the system, mining could help reveal the history of changes, who made the changes, why they made the changes, and even what changes might be made in the future, and by whom.

Repository mining can also leverage the ideas from comprehension. For example, consider the problem of improving a defect prediction tool. Defects ultimately arise from developers having an inadequate understanding of an algorithm, a component, a dependency, or some other fact about an architecture. Measuring, modeling, and controlling for the *comprehensibility* of these different elements—something program comprehension researchers study explicitly—may explain much of the variance which components contain defects.

These are just a few examples of the opportunities for collaboration between these two fields. They also illustrate how seeking the common ground between communities can lead innovation. In this keynote, I discuss these ideas and more, sharing a range of opportunities and grant challenges.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grants No. 1314399 and 1703304. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
ICPC '18, May 27–28, 2018, Gothenburg, Sweden
© 2018 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-5714-2/18/05…$15.00
https://doi.org/10.1145/3196321.3196324