Industry's Cry for Tools that Support Large-Scale Refactoring

James Ivers
jivers@sei.cmu.edu
CMU Software Engineering Institute
Pittsburgh, PA, USA

Robert L. Nord
rn@sei.cmu.edu
CMU Software Engineering Institute
Pittsburgh, PA, USA

Ipek Ozkaya
ozkaya@sei.cmu.edu
CMU Software Engineering Institute
Pittsburgh, PA, USA

Chris Seifried
cgseifried@sei.cmu.edu
CMU Software Engineering Institute
Pittsburgh, PA, USA

Christopher S. Timperley
ctimperley@cmu.edu
Carnegie Mellon University
Pittsburgh, PA, USA

Marouane Kessentini
kessentini@oakland.edu
Oakland University
Rochester, MI, USA

ABSTRACT

Software refactoring plays an important role in software engineering. Developers often turn to refactoring when they want to restructure software to improve its quality without changing its external behavior. Compared to small-scale (floss) refactoring, many refactoring efforts are much larger, requiring entire teams and months of effort, and the role of tools in these efforts is not as well studied. This short paper introduces an industry survey that we conducted. Results from 107 developers demonstrate that projects commonly go through multiple large-scale refactorings, each of which requires considerable effort. While there is often a desire to refactor, other business concerns such as developing new features often take higher priority. Our study finds that developers use several categories of tools to support large-scale refactoring and rely more heavily on general-purpose tools like IDEs than on tools designed specifically to support refactoring. Tool support varies across the different activities (spanning communication, reasoning, and technical activities), with some particularly challenging activities seeing little use of tools in practice. Our study demonstrates a clear need for better large-scale refactoring tools.

CCS CONCEPTS

• **Software and its engineering → Software evolution**; **Maintaining software**; **Software maintenance tools**; **Development frameworks and environments**.

KEYWORDS

refactoring, refactoring tools, software automation, software evolution

ACM Reference Format:
James Ivers, Robert L. Nord, Ipek Ozkaya, Chris Seifried, Christopher S. Timperley, and Marouane Kessentini. 2022. Industry's Cry for Tools that Support Large-Scale Refactoring. In *44nd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP '22), May 21–29, 2022, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3510457.3513074

ICSE-SEIP '22, May 21–29, 2022, Pittsburgh, PA, USA

ACM ISBN 978-1-4503-9226-6/22/05.
https://doi.org/10.1145/3510457.3513074

1 INTRODUCTION

Refactoring is defined as restructuring software to improve its quality without altering its external behavior [5]. The need to restructure software can come from such diverse goals as improving software quality, migrating to new platforms like cloud, containerizing software for DevOps, incorporating new technologies, or extracting capabilities for strategic reuse. Many of these scenarios involve refactoring at a large scale and imply broad changes to the system that cannot be accomplished through local code changes. This paper focuses on these larger refactoring efforts, which we refer to as **large-scale refactoring** and is the first study of large refactoring efforts from multiple industry organizations.

Large-scale refactoring involves either pervasive changes across a codebase or extensive changes to a substantial element of the system (e.g., greater than 10k LOC) and often involves a substantial commitment of resources, requiring management approval. One example is the need to partition legacy monoliths into smaller pieces to create separately deployable, scalable, and evolvable units. Another is restructuring interfaces and communication patterns to enable replacement of a legacy feature by an improved or less proprietary alternative.

Murphy-Hill and Black [4] introduced two different notions of refactoring: the need to continually tweak code while making other changes (floss refactoring) and infrequent, but focused changes to improve unhealthy code (root-canal refactoring). Developers have common access to tool support in integrated development environments (IDEs) that supports floss refactoring, but studies have shown that many developers do not trust these features [3, 4]. Large-scale refactoring is more closely related to root-canal refactoring in that both focus on structural improvements that are not intermingled with other changes. However, we distinguish it from the common use of root-canal refactoring in scale and motivation. Many examples of root-canal refactorings in the literature focus on system wide code quality improvements and do not represent efforts that require significant commitment of resources.

To understand how developers engage with large-scale refactoring and how they use tools to support different activities involved, we conducted a developer survey. A total of 107 participants took part in the survey, the majority of whom work in industry and have 10+ years of experience. The survey elicited responses on topics that include how common large-scale refactoring is in practice, why it is and is not performed, what challenges developers face in

performing large-scale refactoring, and what tools developers have used in their refactoring efforts [1].

2 IMPLICATIONS FOR INDUSTRY

Industry software goes through periodic structural changes as part of continuous evolution [2]. While intuitively we know that refactorings support these changes, we also know that these efforts are significantly larger in scale than the kinds of changes common in floss refactoring and so may have different implications on desirable tool support.

Large-scale refactoring is a distinct activity for which significant resources need to be allocated, rather than being something that developers can easily weave into their day-to-day work. It includes communication activities like persuading stakeholders of benefits and managing expectations; reasoning activities that span understanding code, requirements, and intent; and technical activities like integrating data from and capabilities of different tools to realize desired changes. In short, it is reasonable to think of large-scale as having a project-like scope that includes building blocks like those of floss refactoring, but that also includes many others.

We captured data for refactoring efforts that were estimated to require a mean of more than 1500 staff days of effort. These refactoring efforts were often motivated by broader business concerns than quality improvement. Moreover, the need for large-scale refactoring is relatively common, with most developers having conducted large-scale refactoring multiple times in addition to having to forego desired large-scale refactoring.

As part of our study, we sought to understand whether the kinds of tools used in large-scale refactoring differ from those used in other refactoring efforts, the different activities involved in refactoring, and how the tools being used support those activities. Our findings confirm existing research on the challenges of smaller-scale (floss) refactoring activities. However, our results demonstrate that when it comes to tools used to perform large-scale refactoring, developers use several categories of tools beyond those that implement refactorings in code.

Tool support varies across the different activities that are involved in large-scale refactoring, with some particularly challenging activities seeing little use of tools in practice. The broad range of tools offered by the respondents go beyond IDEs, including diverse tools such as static code analyzers, issue trackers and wikis, testing tools, and custom scripts. References to custom scripts were significantly more common for large-scale refactoring, suggesting significant tool gaps that are more apparent at scale. While developers broadly agree that better tools are desired, they vary in the activities and degree of intelligence they want in tools.

Responses included several doubts about the feasibility of tool support for large-scale refactoring. A key challenge in developing tools that support large-scale refactoring is improving our understanding of what motivates such activities. Tools for smaller scale refactoring often start with an assumption that the goal is to remove specific code smells or improve specific code quality metrics. While these improvements offer business value in the form of improved software maintainability and developer productivity, these improvements are not always what motivates businesses to invest and hence may be misaligned with project needs. Tools that address different motivations or allow users to express their improvement goals could provide more options for developers.

Our results, similar to those of other studies [6], show that factors such as unclear value, risk of new errors, or resource constraints influence decisions about whether to perform large-scale refactoring. Our respondents reported that prioritizing new features over refactoring and perceiving the cost of refactoring as too high were both the most common and most important reasons that their organizations decided to forgo refactoring. These are familiar rationales to developers in industry.

However, while prioritizing new features over refactoring was the most common reason for forgoing large-scale refactoring, the majority of respondents reported the inability or slowing pace of delivering features as a consequence of forgoing refactoring.

Better tools can help change these business decisions and avoid the consequences that follow. As most of the reasons provided boil down to cost-benefit decisions, tools that reduce cost can shift the balance. A mean of more than 1500 staff days of effort spent on large-scale refactorings suggests many opportunities to reduce the work involved in refactoring activities. While the common wisdom is to focus research on floss refactoring because it is more common (orders of magnitude more so), this perspective neglects the cost difference (large-scale refactorings being orders of magnitude larger).

Our study demonstrates a clear need for better tools and an opportunity for refactoring researchers to make a difference in industry. More detailed discussion of the analysis and results are described at Ivers et. al [1].

ACKNOWLEDGMENTS

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute. DM22-0072

REFERENCES

[1] James Ivers, Robert L. Nord, Ipek Ozkaya, Chris Seifried, Christopher S. Timperley, and Marouane Kessentini. 2022. Industry Experiences with Large-Scale Refactoring. (2022). arXiv:2202.00173

[2] James Ivers, Ipek Ozkaya, Robert L. Nord, and Chris Seifried. 2020. Next Generation Automated Software Evolution Refactoring at Scale. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2020)*. Association for Computing Machinery, New York, NY, USA, 1521–1524.

[3] Miryung Kim, Thomas Zimmermann, and Nachiappan Nagappan. 2014. An empirical study of refactoring challenges and benefits at Microsoft. *IEEE Transactions on Software Engineering* 40, 7 (2014), 633–649.

[4] Emerson Murphy-Hill and Andrew P. Black. 2008. Refactoring Tools: Fitness for Purpose. *IEEE Software* 25, 5 (2008), 38–44.

[5] William F Opdyke. 1992. Refactoring object-oriented frameworks. (1992).

[6] Ewan Tempero, Tony Gorschek, and Lefteris Angelis. 2017. Barriers to Refactoring. *Commun. ACM* 60, 10 (Sept. 2017), 54–61.