Analysis of a Code Review Tool Evolution: A Case Study of Rietveld to Gerrit

Osamu Mizuno, Graduate School of Science and Technology, Kyoto Institute of Technology, Kyoto, Japan
Junwei Liang, Graduate School of Science and Technology, Kyoto Institute of Technology, Kyoto, Japan

ABSTRACT

In this study, the authors intend to assess the improvements of Gerrit. The central concern is “Does Rietveld evolve into Gerrit as the developers intended?” To answer this question, the authors first compare qualitative features of two code review tools. The authors then conducted an interview with a developer of Gerrit and obtained the developer’s original intention of improvements in Gerrit. By analyzing mined data from code review logs, the authors try to explain the effects of improvements quantitatively. The result of analysis showed the authors that the improvements of Gerrit that the developer is expected are not observed explicitly.

Keywords: Code Review, Empirical Study, Software Development, Software Evolution, Tool Evolution

1. INTRODUCTION

Mining software repositories has become a current trend of software engineering. Much research using mining technique has been done to discover various issues in software engineering (Zimmermann, Weißgerber, Diehl, & Zeller, 2005). Repositories mining research often focuses on the quality of software. For example, detection of fault-prone software modules is one of good field of repositories mining (Catal & Diri, 2009; Hata, 2012). In fact, the number of fault-prone module detection approaches using mining technique has rapidly increases in last five years.

In order to assure the quality of software, early detection of defects is recommended. The code review is one of effective ways for such early detection of defects in software (Rigby & Storey, 2011). The code review activities include various useful insights for software quality. However, especially in open source software (OSS) developments, records of code review merely remained in a systematic way.

DOI: 10.4018/ijisi.2015010102
Logs of reviews were mostly on the mailing-list, and researchers needed so many efforts to obtain data from mailing-lists or unstructured data (Thomas, 2012).

Recently, for the effectiveness of the code review process in OSS development, tools for code review have been developed. One common weakness of mail based review was the lack of linking between patches and the version control system (Rigby, 2004). The review with tools provides this linkage. For example, a diff made against an older version of a file can be updated by selecting the most reviewed version within a tool, or an approved review can be immediately committed into the version control system.

Not so many open source projects, however, adopt review with tools. Chromium is one of them, which is a successful open source browser project used the code review tool called Rietveld. Rietveld is an open source code review tool based on Mondrian, the internal code review tool used by Google. Both of Rietveld and Mondrian was created by Guido van Rossum who is best known as the author of the Python programming language.

Rietveld was a good tool for code review. The developers, however, appended patches to Rietveld to improve features they want. As a result, they built a new tool for code review from a scratch. The documents of Gerrit says the background as follows (Gerrit, 2012c):

“Gerrit Code Review started as a simple set of patches to Rietveld, and was originally built to service AOSP. This quickly turned into a fork as we added access control features that Guido van Rossum did not want to see complicating the Rietveld code base. As the functionality and code were starting to become drastically different, a different name was needed. Gerrit calls back to the original namesake of Rietveld, Gerrit Rietveld, a Dutch architect.” For this reason, Gerrit is a successor of Rietveld, and thus it is natural that Gerrit has improved features from Rietveld. Like Rietveld and Gerrit, software is sometimes improved and evolved into another new software. Such improvement is done by various reasons, for example, user’s requirements, developer’s intuition, performance issues, and so on. Although the effect of improvement should be evaluated, it is hard to measure and thus remained not evaluated.

In this study, we aim to evaluate the improvements of Gerrit. The basic question is “Does Rietveld evolve into Gerrit as the developers intended?” To answer this question, we first compare qualitative features of two code review tools. We then conducted an interview with a developer of Gerrit and obtained the developer’s original intention of improvements in Gerrit. By analyzing mined data from code review logs, we try to clarify the effects of improvements quantitatively.

The rest of this paper is organized as follows: Section 2 describes the code review process with tools and comparison of code review tools. Section 3 shows the result of interview with a developer of Gerrit and research questions. Mined data from code review repositories are explained in Section 4. Section 5 investigates the research questions. The threats to validity are discussed in Section 7. Finally, Section 8 concludes this study.

2. CODE REVIEW TOOLS

2.1. Code Review Process

Figure 1 shows an example process of code review using code review tools. Figure 1(a) shows a case of Rietveld and Figure 1(b) shows a case of Gerrit. In both tools, the fundamental process is similar. First, a developer creates a review issue with his/her patch to the source code. Then some developers (reviewers) look at the patch and write messages or inline comments to the creator if they find any problems. Gerrit calls back to the original namesake of Rietveld, Gerrit Rietveld, a Dutch architect.” For this reason, Gerrit is a successor of Rietveld, and thus it is natural that Gerrit has improved features from Rietveld. Like Rietveld and Gerrit, software is sometimes improved and evolved into another new software. Such improvement is done by various reasons, for example, user’s requirements, developer’s intuition, performance issues, and so on. Although the effect of improvement should be evaluated, it is hard to measure and thus remained not evaluated.

In this study, we aim to evaluate the improvements of Gerrit. The basic question is “Does Rietveld evolve into Gerrit as the developers intended?” To answer this question, we first compare qualitative features of two code review tools. We then conducted an interview with a developer of Gerrit and obtained the developer’s original intention of improvements in Gerrit. By analyzing mined data from code review logs, we try to clarify the effects of improvements quantitatively.

The rest of this paper is organized as follows: Section 2 describes the code review process with tools and comparison of code review tools. Section 3 shows the result of interview with a developer of Gerrit and research questions. Mined data from code review repositories are explained in Section 4. Section 5 investigates the research questions. The threats to validity are discussed in Section 7. Finally, Section 8 concludes this study.
Automated Synthesis and Ranking of Secure BPMN Orchestrators
www.igi-global.com/article/automated-synthesis-and-ranking-of-secure-bpmn-orchestrators/113726?camid=4v1a

Information Systems Curriculum Design Processes
www.igi-global.com/chapter/information-systems-curriculum-design-processes/21091?camid=4v1a