Refactoring-inducing in pull requests code review: An exploratory case study

An overview

Q3 2019

Abstract. In short, we present our case study's objective, the research questions, the formal definition of the refactoring-inducing term, and the proposed design.

Contents

Objective

Our work aims to investigate what to extent the pull requests code review induces refactorings.

Refactoring-inducing definition

In this work, the refactoring-inducing review means the occurrence of refactoring edits performed as part of the changes in commits at a GitHub pull request (PR). Thus, suppose an initial commit with 0 or a few refactorings in a PR. Then, developers add new commits with refactorings that means the refactorings were a result of the code review process. In order to clarify, consider the following mathematical definition.

Let $S = \{s_1, s_2, ..., s_s\}$, a set of GitHub projects. Each project s_i, $1 \leq i \leq s$, has a set of pull requests $P(s_i) = \{p_1, p_2, ..., p_p\}$. Each pull request p_j, $1 \leq j \leq p$, has a set of commits $C(p_j) = \{c_1, c_2, ..., c_c\}$, where each commit has a set of refactoring edits $R(c_k) = \{r_1, r_2, ..., r_r\}$, $1 \leq k \leq c$. A code review is defined as a function $Q(F(c_k), F(c_{k-1})) = \{(d_1, M(d_1)), (d_2, M(d_2)), ..., (d_d, M(d_d))\}$, where each pair $(d_d, M(d_d)))$ means a diff region d_z, $1 \leq z \leq d$, and its respective set of review comments left by reviewer(s), $M(d_z) = \{m_1, m_2, ..., m_m\}$. Thus, a refactoring-inducing review occurs when there is at least one $|R(c_k)| \geq 1$, $1 \leq k \leq c$.

Research questions

RQ₁ How common are refactoring-inducing pull requests?

RQ₂ What refactoring types often take place in pull requests?

RQ₃ What characterize these refactoring edits?

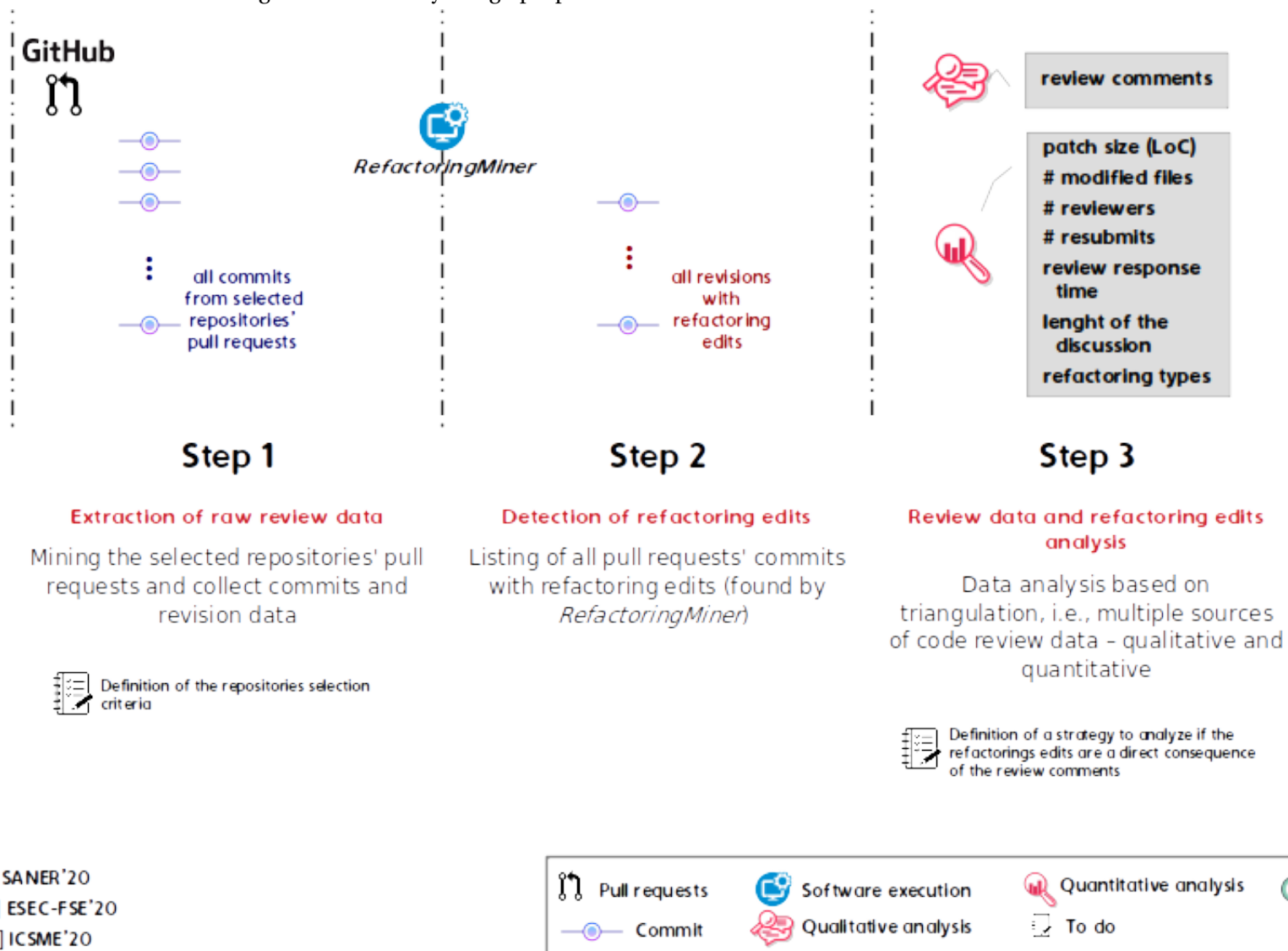**RQ₄** How do these refactorings improve code maintainability?

RQ₅ How do refactoring-inducing pull requests compare to non-refactoring-inducing ones?

Case study design

In figure 1, we show our case study design proposal, which consists of three steps: extraction of review data, *RefactoringMiner*[1] execution, quantitative and qualitative data analysis.

---

[1] https://github.com/tsantalis/RefactoringMiner

Figure 1: Case study design proposal

GitHub

all commits
from selected
repositories'
pull requests

RefactoringMiner

all revisions
with
refactoring
edits

review comments

patch size (LoC)
modified files
reviewers
resubmits
review response
time
lenght of the
discussion
refactoring types

Step 1

Extraction of raw review data

Mining the selected repositories' pull
requests and collect commits and
revision data

Definition of the repositories selection
criteria

Step 2

Detection of refactoring edits

Listing of all pull requests' commits
with refactoring edits (found by
RefactoringMiner)

Step 3

**Review data and refactoring edits
analysis**

Data analysis based on
triangulation, i.e., multiple sources
of code review data – qualitative and
quantitative

Definition of a strategy to analyze if the
refactorings edits are a direct consequence
of the review comments

[Oct 2019] SANER'20
[Feb 2020] ESEC-FSE'20
[Mar 2020] ICSME'20

Pull requests   Software execution   Quantitative analysis   Target (one of them)

Commit   Qualitative analysis   To do