

---

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

---

## **A brief "case history": Merged pull requests**

---

Q4 2019

This report presents the results obtained in a brief "case history" carried out on merged pull requests from the Apache project at GitHub, on August to December 2019.

## **Contents**

<b>1. Collection of the repositories' raw data</b>	<b>2</b>
<b>2. Detection of refactoring edits</b>	<b>2</b>
<b>3. Collection of code's maintainability and review data</b>	<b>3</b>
3.1. Selection of code maintainability and review metrics . . . . .	3
3.2. Data collection . . . . .	5
<b>4. Data analysis</b>	<b>5</b>
4.1. Quantitative analysis . . . . .	5
4.2. Qualitative analysis . . . . .	21
<b>Appendix A. Strategy to deal with squashed commits on GitHub</b>	<b>23</b>

## 1. Collection of the repositories' raw data

Table 1 presents the search results for *not-archived Java repositories* from *Apache*<sup>1</sup> at GitHub. We chose Apache because of its considerable size, complexity, and its developers who, in turn, follow the pull request (PR) model in their practice.

Table 1: Search results for Apache's repositories  
(on August 26 to September 7, 2019)

Number of repositories	Number of repositories with merged PRs	Number of merged PRs
956	467	65,006

When the commits of one PR are squashed into a single commit, we need to recover the 'lost' commits' history, because they are omitted when we run the refactoring detection at the PR level. After the recovering of the squashed commits, we can execute the detection of refactoring edits at the commit level<sup>2</sup>. In Appendix A, we describe the developed strategy for the squashed commits recovering, which results are listed in table 2.

Table 2: Number of PRs and recovered commits for the refactoring detection on Apache's repositories

Number of PRs	Number of recovered commits
48,338	53,915

## 2. Detection of refactoring edits

During the *RefactoringMiner* [1] execution, some PRs have put away because of the following situation (*java.lang.IllegalArgumentException*):

- a commit has no changed files and **two parents** (for example, <https://github.com/apache/cloudstack/commit/3afc4a176f5287c271db4d81293343f242b1c792>),
- a commit has changed files and **two parents** (for example, <https://github.com/apache/cloudstack/commit/95f1de89b40a5989742b6f1ded46d8def139f97c>),

As a result, in Table 3, we present the extent of the sample, in terms of the total number of covered commits, PRs, and repositories, obtained from the *RefactoringMiner*.

In table 4 we specify the number of repositories, PRs, and commits considered in the *RefactoringMiner* execution at commit and PR levels.

<sup>1</sup><https://github.com/apache>

<sup>2</sup>Running the *RefactoringMiner*'s *detectAtPullRequest* and *detectAtCommit* methods, respectively.

Table 3: Results from the *RefactoringMiner* execution for Apache's repositories  
 (on September 18 to October 2, 2019)

Number of repositories	Number of PRs	Number of commits	Number of detected refactorings
236	15,885	56,478	170,029

Table 4: Results from the *RefactoringMiner* execution at commit and pull request levels

Level	Number of repositories	Number of PRs	Number of commits	Number of detected refactorings
commit	145	3,375	20,908	65,198
PR	146	12,510	35,570	104,831

### 3. Collection of code's maintainability and review data

#### 3.1. Selection of code maintainability and review metrics

In order to study code maintainability, we followed the Goal-Question-Metric (GQM) model [2], because it eases the definition of measurements, on a top-down strategy, for selecting and interpreting metrics in accordance with the research's objective. As a result, the selected metrics were defined as shown in figure 1.

Figure 1: Selected metrics based on GQM model

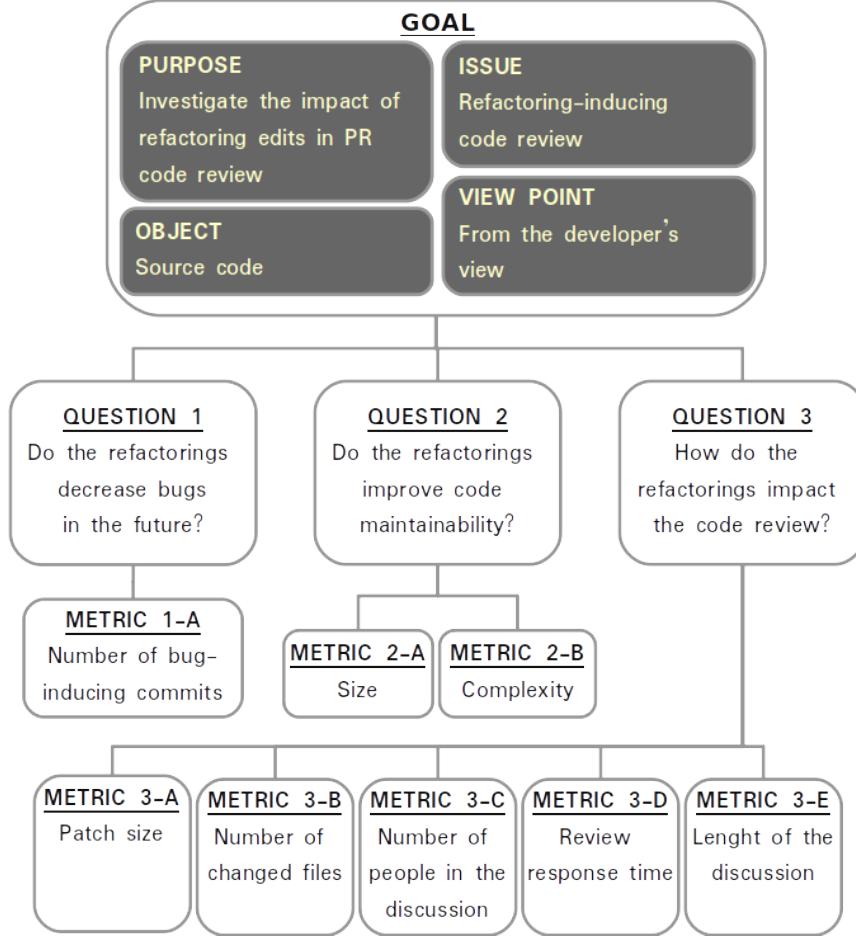


Table 5: Previous research that supports the selection of the metrics

Metric(s)	Work	Reason for selection
1-A	[3]	Bavota et al. found, supported by the SZZ algorithm, that the percentage of bugs likely induced by refactorings is relatively low (15%), nevertheless, refactorings involving hierarchies tend to induce bugs very frequently (40%).
2-A – 2-B	[4]	Refactoring decreases the complexity at design and source code levels and, consequently, improves the code maintainability. In this way, Kádár et al. found that complexity and size play an important role when investigating refactored codes.
3-A – 3-E	[5]	These are a few factors influencing code review quality, in the developers' perception, found by Kononenko et al.

### 3.2. Data collection

The collection of review data was performed from November 26 to November 28, 2019. During this process, data from six pull requests could not be gathered due to the following errors:

- *UnprocessableEntity: the patch could not be processed because too many files changed*: apache/beam PR 5753, apache/dubbo PR 3257, apache/rocketmq-externals PR 131, and apache/usergrid PR 65,
- *ConnectionError: read time out*: apache/incubator-dolphinscheduler PR 218,
- *TypeError: 'NoneType' object is not subscriptable*: apache/kafka PR 5338.

## 4. Data analysis

### 4.1. Quantitative analysis

**RQ<sub>1</sub>. How common are refactoring-inducing pull requests?**

Figure 2: Occurrence of refactoring edits in the Apache's repositories

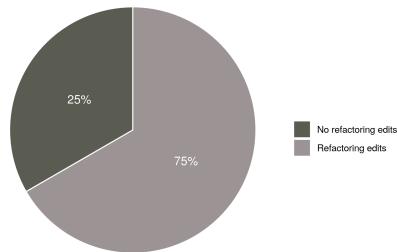


Table 6: Number of detected refactoring edits at Apache's repositories

<b>Number of repositories with refactoring edits</b>	177
<b>Number of repositories without refactoring edits</b>	59

Figure 3: Occurrence of refactoring edits in the Apache's pull requests

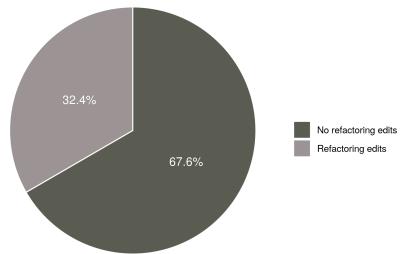


Table 7: Number of detected refactoring edits at Apache's pull requests

<b>Number of PRs with refactoring edits</b>	5,151
<b>Number of PRs without refactoring edits</b>	10,734

Figure 4: Occurrence of refactoring edits in the Apache's commits

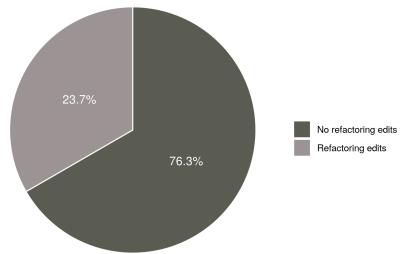


Table 8: Number of detected refactoring edits at Apache's commits

<b>Number of commits with refactoring edits</b>	13,360
<b>Number of commits without refactoring edits</b>	43,118

Table 9: Measures of central tendency for refactoring edits in the Apache's repositories

Measure	at PRs	at commits
mean	33.0	12.7
standard deviation	179.4	49.7
median	5	3
interquartile range (IQR)	17	7

Figure 5: Distribution of refactoring edits in the Apache's pull requests

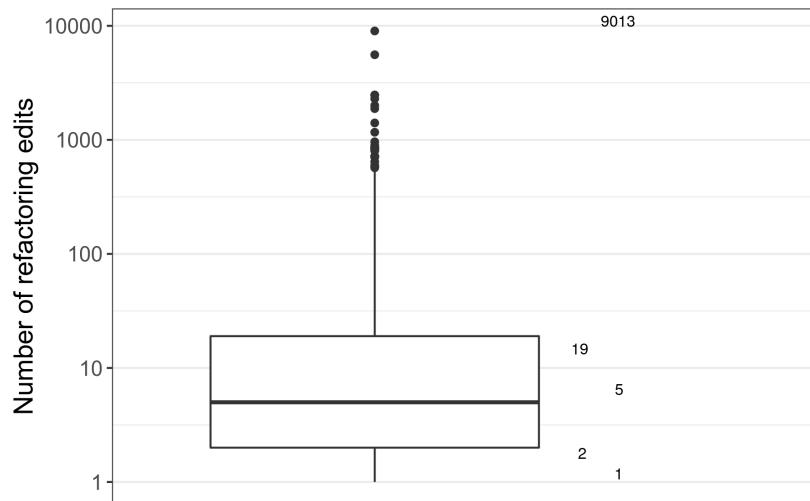
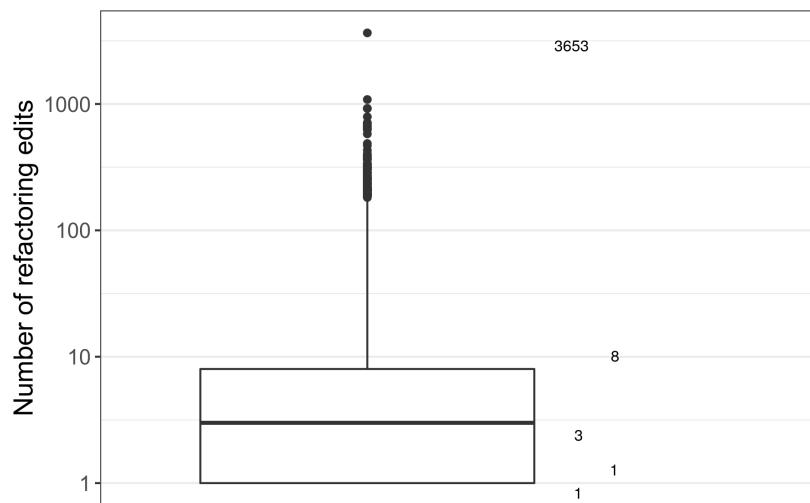


Figure 6: Distribution of refactoring edits in the Apache's commits



**RQ<sub>2</sub>. What refactoring types often take place in pull requests?**

Table 10: Distribution of refactoring edits through commits in the Apache's pull requests

Position of the refactoring edits	Number of PRs	Percentage (%)
(all) in one commit – PR has only one commit	2,942	57.12
(most) in the first commit	703	13.65
(most) in the intermediate commits	693	13.45
(most) in the last commit	515	10.00
equally distributed among commits	298	5.79
<b>Total</b>	<b>5,151</b>	<b>100.00</b>

Table 11: Number of edits by refactoring kind in the Apache's pull requests

Refactoring kind	Number of edits	Percentage (%)
Change Type	70,459	41.44
Rename	47,088	27.69
Move	23,593	13.88
Extract	18,853	11.09
Pull	4,740	2.79
Inline	1,906	1.12
Push	1,311	0.77
Replace	789	0.46
Parameterize	704	0.41
Merge	426	0.25
Split	160	0.09
<b>Total</b>	<b>170,029</b>	<b>100.00</b>

Figure 7: Types of the detected refactorings in the Apache's pull requests

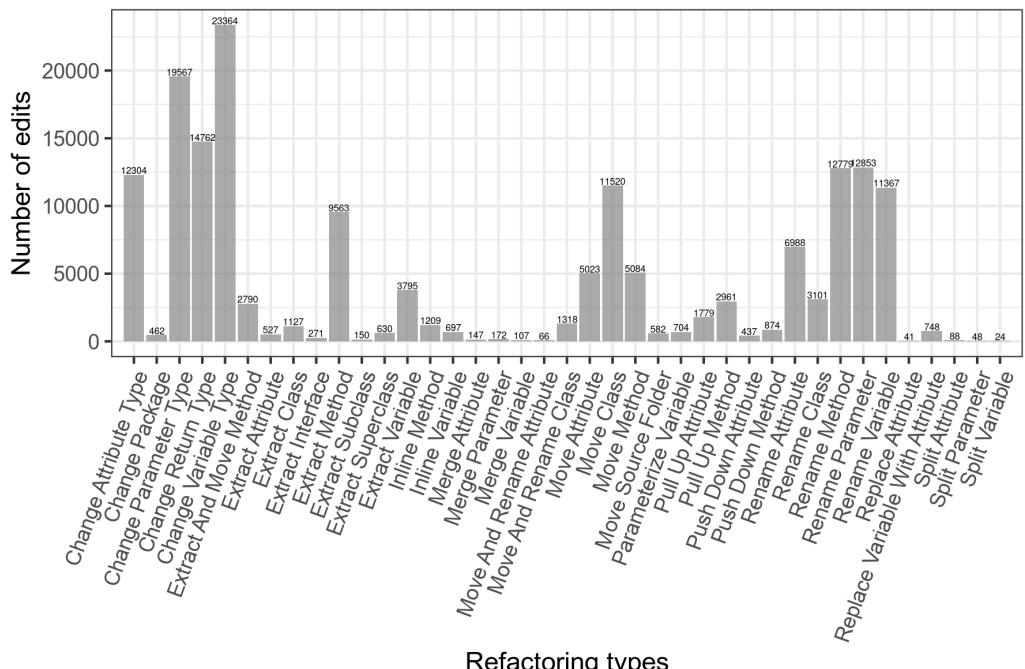


Table 12: Measures of central tendency for refactoring types in the Apache's repositories

Measure	at PRs	at commits
mean	4.9	3.2
standard deviation	5.4	3.5
median	3	2
interquartile range (IQR)	5	3

Figure 8: Distribution of refactoring types in the Apache's pull requests

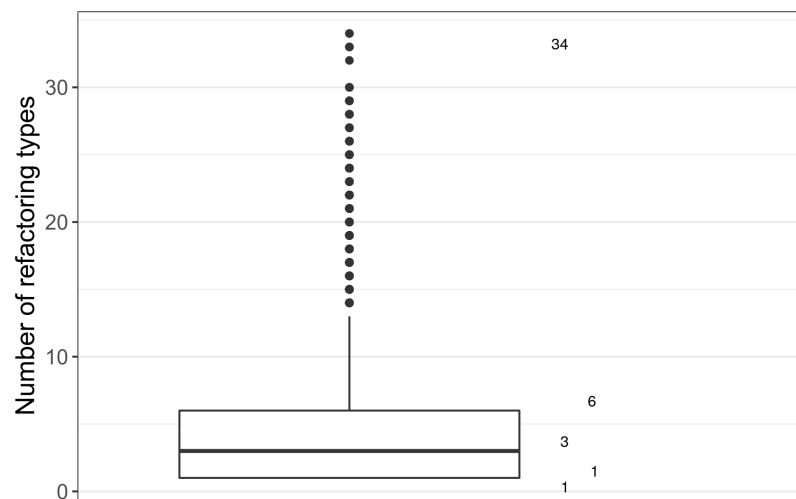


Figure 9: Occurrence of refactoring types in the Apache's pull requests

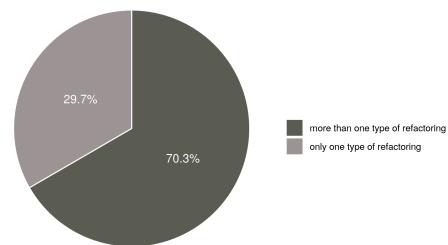


Figure 10: Distribution of refactoring types in the Apache's commits

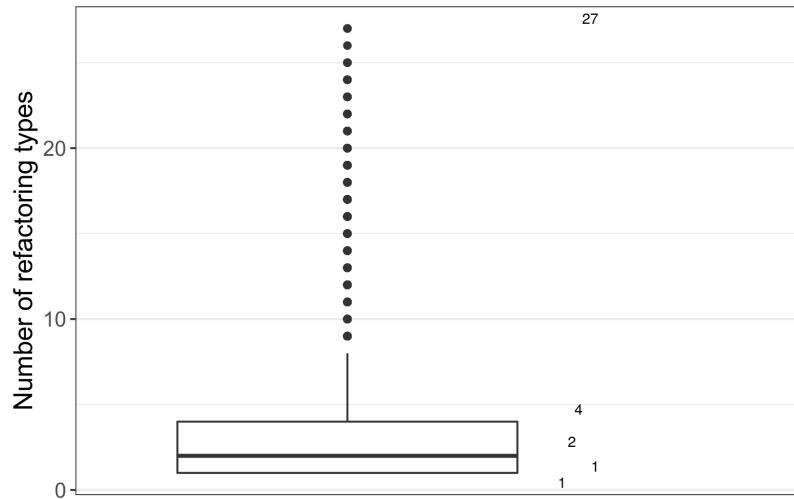
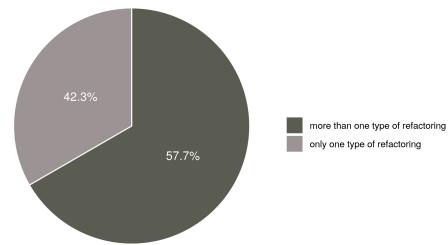


Figure 11: Occurrence of refactoring types in the Apache's commits



### **RQ<sub>3</sub>. What characterize these refactoring edits?**

Table 13: Number of edits by target's granularity level of the refactorings in Apache's pull requests

Target's granularity level	Number of edits	Percentage (%)
package/folder	1,044	0.61
class/interface	18,117	10.66
method	50,022	29.42
attribute/parameter/variable	100,846	59.31
<b>Total</b>	<b>170,029</b>	<b>100.00</b>

Figure 12: Targets of the refactoring edits in the Apache's pull requests

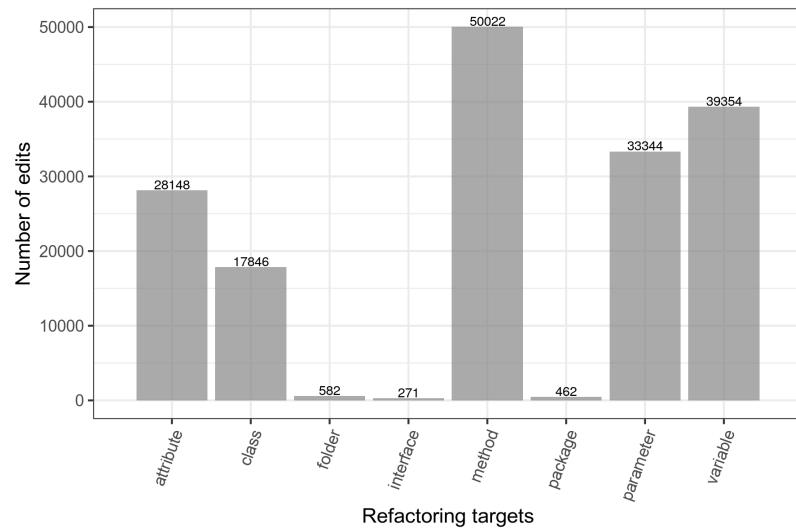


Table 14: Measures of central tendency for refactoring targets in the Apache's repositories

Measure	at PRs	at commits
mean	2.6	2.1
standard deviation	1.6	1.3
median	2	2
interquartile range (IQR)	3	2

Figure 13: Distribution of refactoring targets in the Apache's pull requests

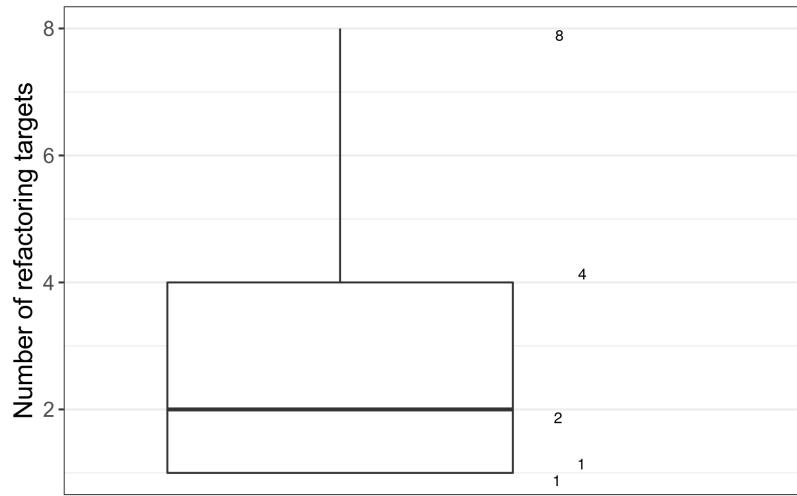


Figure 14: Refactorings' targets in the Apache's commits

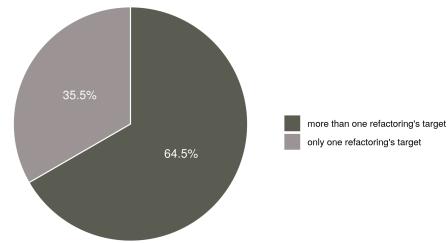


Figure 15: Distribution of refactoring targets in the Apache's commits

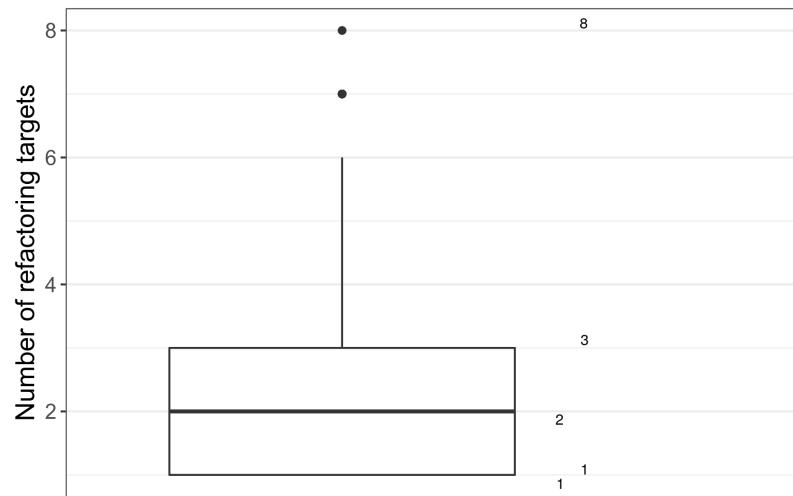
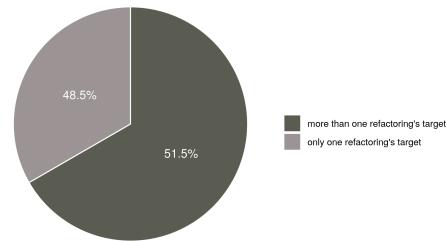


Figure 16: Refactorings' targets in the Apache's commits



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

**Doubts**

- 
- Are size and complexity enough?
  - How about coupling and inheritance?
- 

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

Table 15: Measures of central tendency for **patch size (in KB)** in the Apache's pull requests

Measure	refactoring edits	no refactoring edits
mean	263.5	79.8
standard deviation	2,649.7	1,447.2
median	31.5	3.9
interquartile range (IQR)	73.2	9.9

Figure 17: Distribution of patch size (in KB) in the Apache's pull requests with refactoring edits

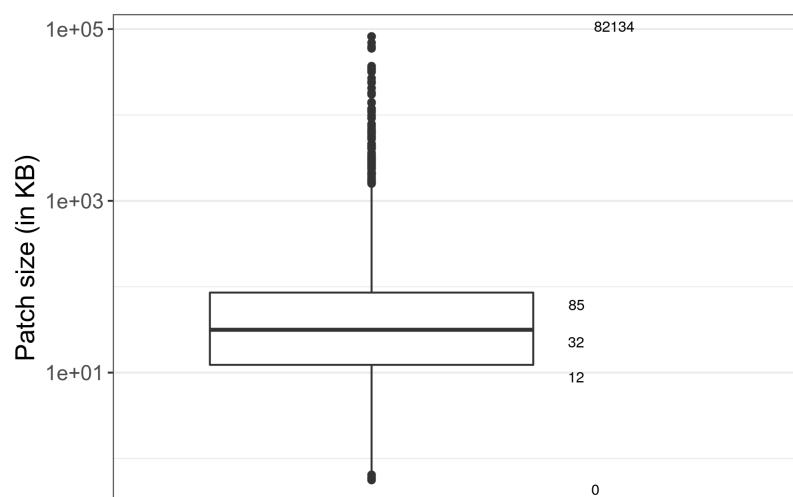


Figure 18: Distribution of patch size (in KB) in the Apache's pull requests without refactoring edits

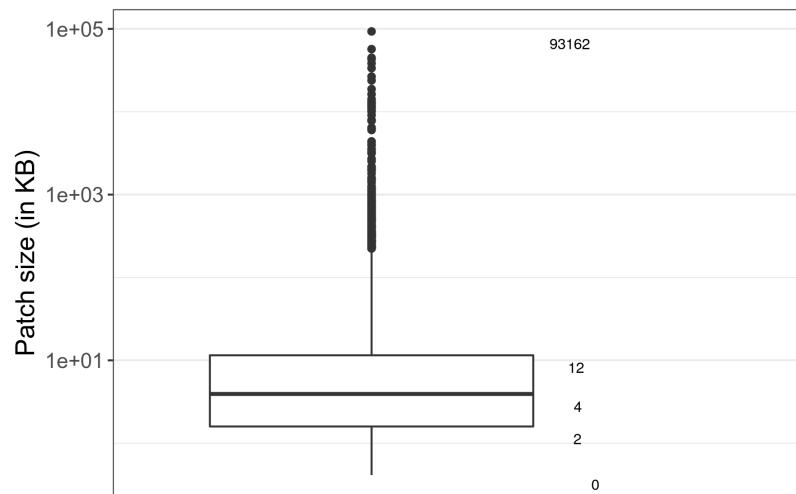


Table 16: Measures of central tendency for **number of changed files** in the Apache's pull requests

Measure	refactoring edits	no refactoring edits
mean	26.6	7.9
standard deviation	122.1	67.6
median	8.0	2.0
interquartile range (IQR)	16.0	3.0

Figure 19: Distribution of number of changed files in the Apache's pull requests with refactoring edits

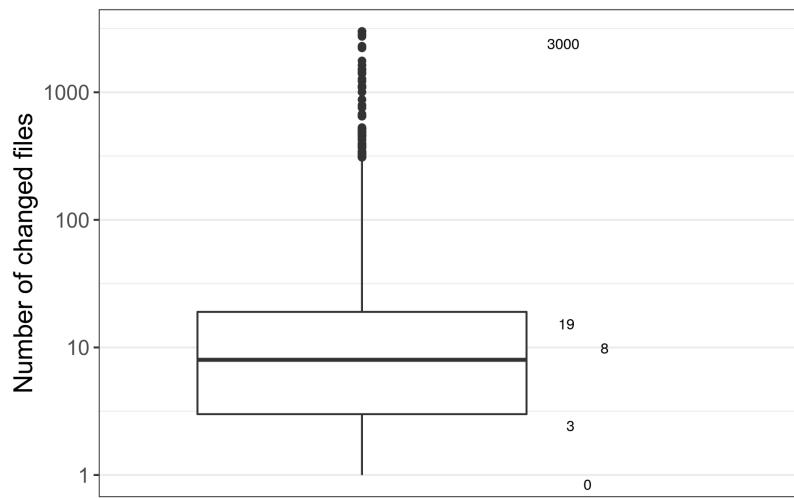


Figure 20: Distribution of number of changed files in the Apache's pull requests without refactoring edits

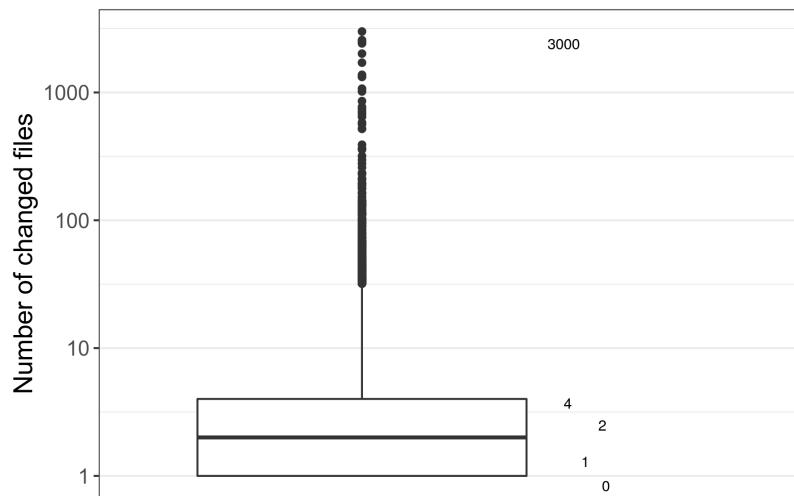


Table 17: Measures of central tendency for **number of people in the discussion** in the Apache's pull requests

Measure	refactoring edits	no refactoring edits
mean	1.2	0.4
standard deviation	1.4	0.9
median	1.0	0.0
interquartile range (IQR)	2.0	0.0

Figure 21: Distribution of number of people in the discussion in the Apache's pull requests with refactoring edits

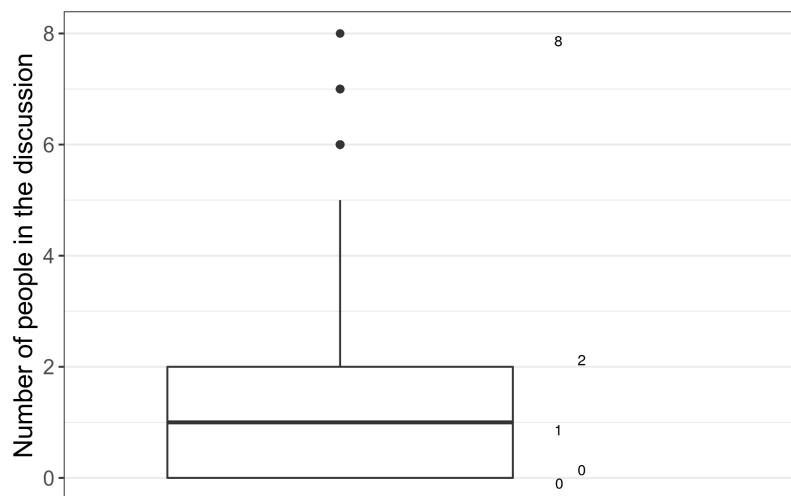


Figure 22: Distribution of number of people in the discussion in the Apache's pull requests without refactoring edits

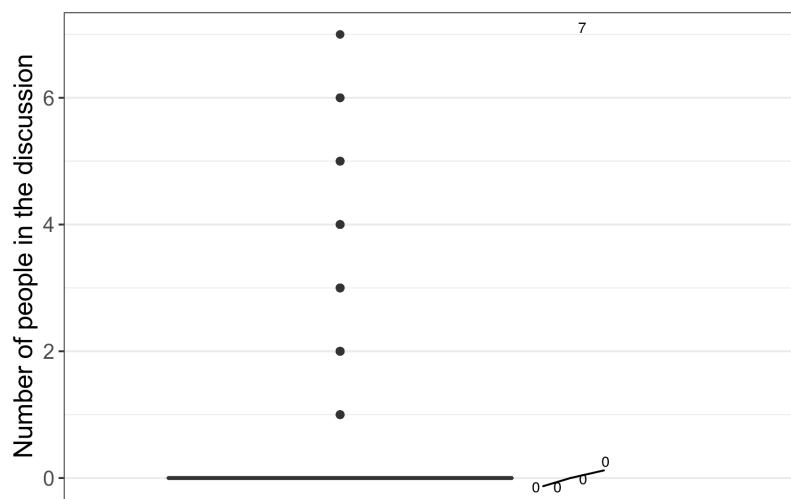


Table 18: Measures of central tendency for **length of the discussion (total number of comments – including review comments)** in the Apache's pull requests

Measure	refactoring edits	no refactoring edits
mean	10.3	2.5
standard deviation	21.4	8.0
median	2.0	0.0
interquartile range (IQR)	12.0	0.0

Figure 23: Distribution of length of the discussion in the Apache's pull requests with refactoring edits

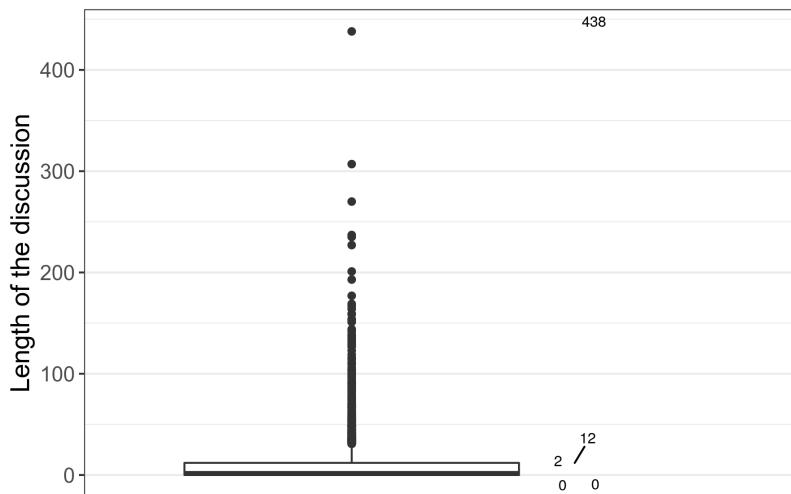


Figure 24: Distribution of length of the discussion in the Apache's pull requests without refactoring edits

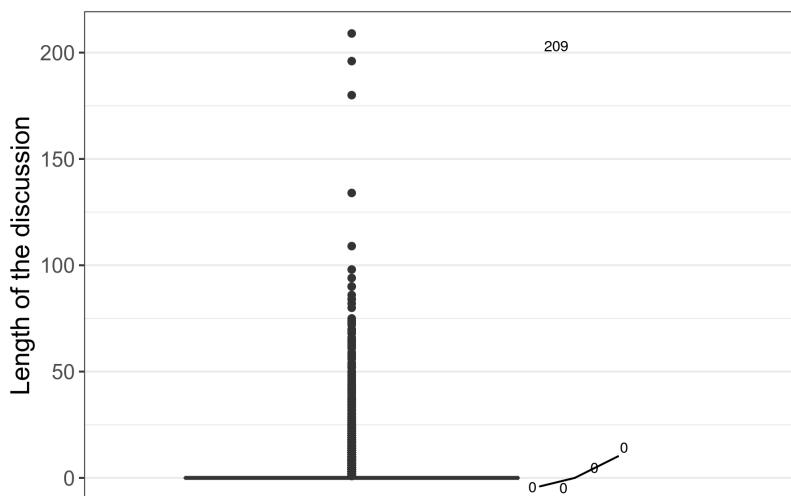


Table 19: Measures of central tendency for **number of review comments** in the Apache's pull requests

Measure	refactoring edits	no refactoring edits
mean	6.9	1.4
standard deviation	16.6	5.1
median	1.0	0.0
interquartile range (IQR)	7.0	0.0

Figure 25: Distribution of number of review comments in the Apache's pull requests with refactoring edits

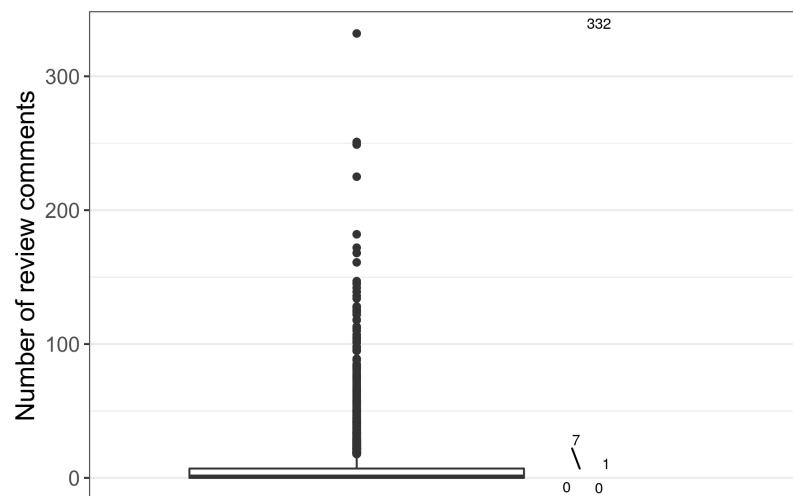
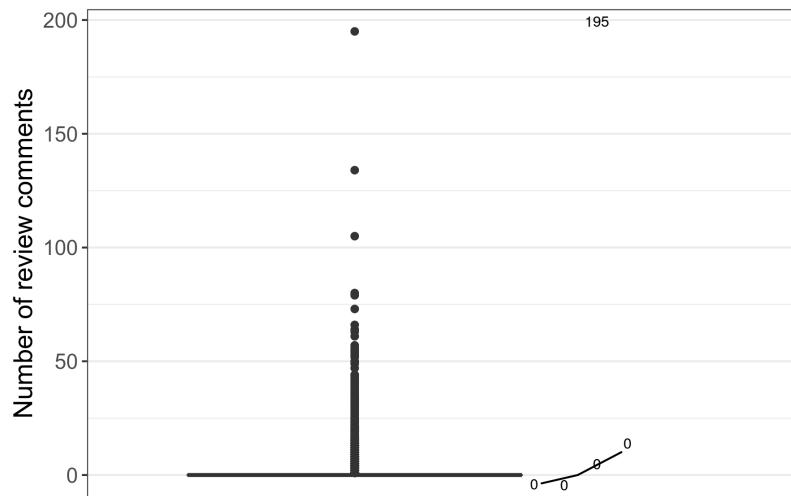


Figure 26: Distribution of number of review comments in the Apache's pull requests without refactoring edits



### Doubts

What relations should we investigate? For instance, patch size *vs* number of refactorings, number of changed files *vs* number of refactorings, number of people in the discussion *vs* number of refactorings, review response time *vs* number of refactorings, number of review comments *vs* number of refactorings, patch size *vs* number of changed files *vs* number of refactorings.

Should we investigate the best model to describe the relations among these review quality metrics?

### Notes

The studying on review response time, and the number of bug-inducing commits are in progress. It is needed to implement our mining script.

## 4.2. Qualitative analysis

### Doubts

What will be the methodology to investigate the relationship between refactoring edits and review comments (qualitative analysis)?

## References

- [1] N. Tsantalis, M. Mansouri, L. M. Eshkevari, D. Mazinanian, and D. Dig, “Accurate and efficient refactoring detection in commit history,” in *Proceedings of the 40th International Conference on Software Engineering*, ICSE ’18, (New York, NY, USA), pp. 483–494, ACM, 2018.
- [2] V. R. Basili, G. Caldiera, and D. H. Rombach, *The Goal Question Metric Approach*, vol. I, p. 528–532. John Wiley & Sons, 1994.
- [3] G. Bavota, B. De Carluccio, A. De Lucia, M. Di Penta, R. Oliveto, and O. Strollo, “When does a refactoring induce bugs? an empirical study,” in *2012 IEEE 12th International Working Conference on Source Code Analysis and Manipulation*, pp. 104–113, Sep. 2012.
- [4] I. Kádár, P. Hegedus, R. Ferenc, and T. Gyimóthy, “A code refactoring dataset and its assessment regarding software maintainability,” in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, vol. 1, pp. 599–603, March 2016.
- [5] O. Kononenko, O. Baysal, and M. W. Godfrey, “Code review quality: How developers see it,” in *Proceedings of the 38th International Conference on Software Engineering*, ICSE ’16, (New York, NY, USA), pp. 1028–1038, ACM, 2016.
- [6] “Merging a pull request - GitHub Help.” <https://help.github.com/en/github/collaborating-with-issues-and-pull-requests/merging-a-pull-request>, accessed 2019-10-27.
- [7] “GraphQL API v4 - GitHub Developer.” <https://developer.github.com/v4/>, accessed 2019-10-27.
- [8] “Git-merge reference - Git Documentation.” <https://git-scm.com/docs/git-merge>, accessed 2019-11-03.

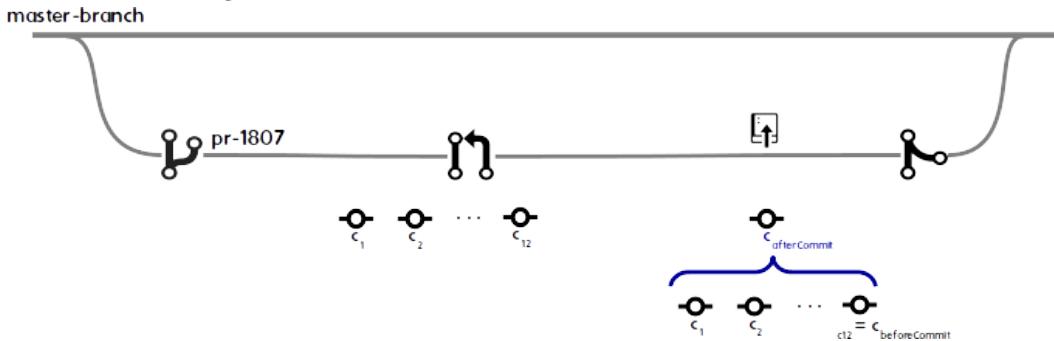
## A. Strategy to deal with squashed commits on GitHub

There are three options to merge a PR [6]:

- **Merge pull request** – merge all the commits into the base branch, in a merge commit,
- **Squash and merge** – squash the commits into one commit, and
- **Rebase and merge** – rebase the commits individually onto the base branch.

In order to deal with the PRs merged by **squash and merge** option, we need to recover the original PR's commits to detect their refactoring edits. To clarify, let us consider the Apache's PR 1807<sup>3</sup>. As we see in figure 27, originally, this PR had 12 commits (from  $c_1$  to  $c_{12}$ ) that were squashed after a *force-pushed* event. Consequently, now, only one commit may be gathered from the PR ( $c_{\text{afterCommit}}$ ).

Figure 27: An overview of the PR 1807 events on GitHub



In this context, we carried out the following strategy<sup>4</sup>:

1. since the *HeadRefForcePushedEvent* is available through GitHub GraphQL API v4 [7], we have recovered the after and before commits of the PR 1807 (table 20), and
2. we rebuilt the original commit's history (table 21), through tracking the commits from  $c_{\text{beforeCommit}}$ , which has the same value of  $c_{12}$ , until reaching the same SHA of the  $c_{\text{afterCommit}}$ 's parent (77cf7e2ee61 fb40e7efd85148ac76947d13dda38), through the *compare* operation available in GitHub REST API v3.

Table 20: GitHub GraphQL API's *HeadRefForcePushedEvent* fields

$c_{\text{afterCommit}}$	ed9b9f2fe279bbafec1680b1229ecbaa40b3ed23
$c_{\text{beforeCommit}}$	b1a5446174485c55a162771b1e804a0587eef361

We applied the strategy's step 1 for gathering the after and before commits from all 65,006 PRs, obtaining 16,708 squashed commits, as indicated in table 22.

<sup>3</sup><https://github.com/apache/drill/pull/1807>

<sup>4</sup>The script, developed in Python, is available in <https://github.com/flaviacoelho>.

Table 21: The recovered history of the PR 1807's commits

<b>Commit index</b>	<b>PR's original commits</b>
$c_1$	21fc7b6d3e6064ff2c28bb1b9920487e7cf995ba
$c_2$	2041aca8887882b6f33a1a4366f44b5f2dac681c
$c_3$	0d521265e79ac05b33480dd3adb2a078ca28e54b
$c_4$	02fb0e9945353e187f5eaa8bea6a5763f3f9b9fb
$c_5$	ab512953b0e097b02fb25e33529bba0e27651fb7
$c_6$	a07936715378f41d7e375be53218d3dfc0a8e45e
$c_7$	6320a77caca59b886a50c5ef47ccb1d6461d98fb
$c_8$	6a712ddf7fe4693594805f64692c2677239fbe08
$c_9$	a14ebf31d6c924cbe877e1f1f672e211e3207e89
$c_{10}$	e816deba8dfbd937f89c216cc8c74aa6adf01aed
$c_{11}$	539bd0edd8348d03df6d17ae4ff2387c10dd10e9
$c_{12}$	b1a5446174485c55a162771b1e804a0587eef361

Table 22: Number of PRs and squashed commits obtained after the strategy's step 1

<b>Number of PRs</b>	<b>Number of squashed commits</b>
48,226	16,708

After the collection of the  $c_{afterCommit}$ 's parent, we realized that some commits have two parents, i.e., they are the result of merging one branch to another by a commit merge that has both of them as its parents, merging known as "true merge" [8]. Therefore, these commits were not considered squashed commits, so their PRs were added for refactoring detection at PR level. After that, we obtained the results shown in table 23.

Table 23: Number of PRs and recovered commits obtained after the strategy's step 2

<b>Number of PRs</b>	<b>Number of recovered commits</b>
48,338	53,915