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

A brief "case history": open pull requests

Q3 2019

Abstract. This report presents the results obtained from a brief "case history" carried out on an Apache repository at GitHub, in July 2019. First, we show the steps followed in order to get a better understand of the data and units of analysis, i.e. the review data and repositories on GitHub, respectively. In short, based on randomly selected open pull requests and their respective refactorings, aiming to answer our research questions, we carried out an investigation on one specific pull request (exploring the commit level), and on more 24 pull requests. We found the occurrence of refactoring edits, of multiple types, most with attribute, parameter, and variable targets; in terms of maintainability, we just explored the number of added and deleted lines of code in each commit, so we realize that the pull requests' changes added lines to the code, in turn, this result drives further investigation on the correlation between refactoring edits and the number of changed lines; we figured out that, in the specific pull request, all resolved conversations are related to commits with refactoring edits, whose most comments were left when methods were the main refactoring's targets; and we also observed a little number of refactorings induced by reviewing and review comments motivated from previously performed refactorings, through a manual inspection of the detected refactorings against the review comments.

Contents

1 One specific *open* pull request

1.1 Selection of the pull request

In table 1, we present one randomly selected GitHub project's PR on July 16th. In that date, this PR was open but currently is merged.

Table 1: A few attributes of the selected pull request

Selected project	#Commits	#Conversations	#Files changed
Apache/drill[1]	12	40	20

[1] Selected pull request available on `https://github.com/apache/drill/pull/1807`

1.2 *RefactoringMiner* execution

In table 2–13, we relate the refactorings detected by *RefactoringMining* and the review comments for all PR's commits.

Table 2: Refactorings detected by *RefactoringMiner* and review comments of `https://github.com/apache/drill/pull/1807/commits/21fc7b6d3e6064ff2c28bb1b9920487e7cf995ba`

#Refactorings	Refactoring Type(s)	Targets	Review comment(s)
10	Move Class	LogRecordReader.ColumnDefn to LogBatchReader.ColumnDefn LogRecordReader.VarCharDefn to LogBatchReader.VarCharDefn LogRecordReader.BigIntDefn to LogBatchReader.BigIntDefn LogRecordReader.SmallIntDefn to LogBatchReader.SmallIntDefn LogRecordReader.IntDefn to LogBatchReader.IntDefn LogRecordReader.Float4Defn to LogBatchReader.Float4Defn LogRecordReader.DoubleDefn to LogBatchReader.DoubleDefn LogRecordReader.DateDefn to LogBatchReader.DateDefn LogRecordReader.TimeDefn to LogBatchReader.TimeDefn LogRecordReader.TimeStampDefn to LogBatchReader.TimeStampDefn	At log.LogFormatPlugin (line 67): "Is it possible not to use guava list?" At log.README.md (line 4): – "... a regex. The original ..." + "... a regex. The original ..." (line 26): – "... This tells..." + "... This tells..." (line 27): – "...‘logRegex‘. This..." + "... ‘logRegex‘. This..." (line 29): "Looks like everywhere in the doc there are two spaces before sentences instead of one. Could you please check and fix?"
1	Rename Parameter	patternIndex to fieldIndex in class log.LogFormatConfig	
1	Move Attribute	formatConfig from log.LogFormatPlugin to log.LogBatchReader	

Table 3: Refactorings detected by *RefactoringMiner* and review comments of `https://github.com/apache/drill/pull/1807/commits/2041aca8887882b6f33a1a4366f44b5f2dac681c`

#Refactorings	Refactoring Type(s)	Targets	Review comment(s)
2	Rename Method	testExplicit() to testExplicitProject() in log.TestLogReader testMissing() to testMissingColumns() in log.TestLogReader	At log.LogFormatPlugin (line 49): "Same here." (line 98): "Why we are matching test here?" (line 128): "Add @param and @return to javadoc" (line 170): "Same here."
1	Change Variable Type	batches:List<QueryDataBatch> to iter:QueryRowSetIterator at testWildcardLargeFile() in log.TestLogReader	
1	Rename Variable	batches:List<QueryDataBatch> to iter:QueryRowSetIterator at testWildcardLargeFile() in log.TestLogReader	

Table 4: Refactorings detected by *RefactoringMiner* and review comments of `https://github.com/apache/drill/pull/1807/commits/0d521265e79ac05b33480dd3adb2a078ca28e54b`

#Refactorings	Refactoring Type(s)	Targets	Review comment(s)
0	–	–	At log.TestLogReader #1 (line 418): "Could you please add unit tests to check how this format plugin works with schema parameter in table function? Example: org.apache.drill.TestSchemaWithTableFunction We might need to check two cases: select * from table(t(schema=>'inline=(col1 varchar)')) select * from table(t(type=>'logRegex', schema=>'inline= (col1 varchar)'))" #2 (line 418): "select * from table(t(schema=>'inline=(col1 varchar)')) should work disregarding format properties. But log format has schema property so I am wondering if there will be a clash or schema parameter will be correctly resolved, since log format has it as list and schema provisioning in string. Since now log format supports schema provisioning, the above query should apply schema for log files, could you please check this query?"

Table 5: Refactorings detected by *RefactoringMiner* and review comments of `https://github.com/apache/drill/pull/1807/commits/02fb0e9945353e187f5eaa8bea6a5763f3f9b9fb`

#Refactorings	Refactoring Type(s)	Targets	Review comment(s)
2	Change Attribute Type	logger:slf4j.Logger to logger:Logger in log.LogBatchReader logger:slf4j.Logger to logger:Logger in log.LogFormatPlugin	At log.LogFormatField (line 31): "Extra space before Field" At log.LogFormatPlugin (line 135): "Add @param and @return to javadoc" (line 193): "Same here." At log.README.md (line 29): "Looks like everywhere in the doc there are two spaces before sentences instead of one. Could you please check and fix?"

Table 6: Refactorings detected by *RefactoringMiner* and review comments of `https://github.com/apache/drill/pull/1807/commits/ab512953b0e097b02fb25e33529bba0e27651fb7`

#Refactorings	Refactoring Type(s)	Targets	Review comment(s)
0	–	–	At log.README.md #1 (line 148): "Table function will work for all log format properties, except of list. Knowing that list is not supported, does it makes sense to replace list schema parameter with String and rename it to avoid clash with schema parameter for schema provisioning." #2 (line 148): "I guess initial choice of list property did not take into account that it does not work with table function. I don't think you can fix this backward compatibility in ZK but since this plugin is a role model for others I think it should have proper configuration, so changing schema property to be String instead List might be reasonable. Or can we have both properties one in String, another in List<String>? We can indicate in release notes that log plugin has been changed and config in ZK must be updated."

Table 7: Refactorings detected by *RefactoringMiner* and review comments of `https://github.com/apache/drill/pull/1807/commits/a07936715378f41d7e375be53218d3dfc0a8e45e`

#Refactorings	Refactoring Type(s)	Targets	Review comment(s)
1	Extract Method	buildTable(...) from testProvidedSchema() in log.TestLogReader	At log.LogFormatPlugin (line 53): "Since properties are log specific should we add log in the properties naming as well as we did for text properties?" At log.README.me (line 29): "Looks like everywhere in the doc there are two spaces before sentences instead of one. Could you please check and fix?" (line 159): #1 "I think we should use different naming, drill.regex.regex look awkward. Maybe drill.regex.pattern or something like this?" #2 "I think using drill.logRegex.regex will be fine."
1	Rename Attribute	testDir to schemaAndConfigDir in log.TestLogReader	
1	Rename Method	defineOutputSchema(...) to defineOutputSchemaFromConfig() in log.LogFormatPlugin	
1	Extract Variable	regex in setupPattern(...) from log.LogFormatPlugin	

Table 8: Refactorings detected by *RefactoringMiner* and review comments of `https://github.com/apache/drill/pull/1807/commits/6320a77caca59b886a50c5ef47cbb1d6461d98fb`

#Refactorings	Refactoring Type(s)	Targets	Review comment(s)
0	–	–	–

Table 9: Refactorings detected by *RefactoringMiner* and review comments of `https://github.com/apache/drill/pull/1807/commits/6a712ddf7fe4693594805f64692c2677239fbe08`

#Refactorings	Refactoring Type(s)	Targets	Review comment(s)
0	–	–	–

Table 10: Refactorings detected by *RefactoringMiner* and review comments of `https://github.com/apache/drill/pull/1807/commits/a14ebf31d6c924cbe877e1f1f672e211e3207e89`

#Refactorings	Refactoring Type(s)	Targets	Review comment(s)
0	–	–	–

Table 11: Refactorings detected by *RefactoringMiner* and review comments of `https://github.com/apache/drill/pull/1807/commits/e816deba8dfbd937f89c216cc8c74aa6adf01aed`

#Refactorings	Refactoring Type(s)	Targets	Review comment(s)
0	–	–	At log.TestLogReader (line 621): "Please add fail()" (line 669): "Same here."

Table 12: Refactorings detected by *RefactoringMiner* and review comments of `https://github.com/apache/drill/pull/1807/commits/539bd0edd8348d03df6d17ae4ff2387c10dd10e9`

#Refactorings	Refactoring Type(s)	Targets	Review comment(s)
1	Inline Method	defineReaderSchema(...) to frameworkBuilder(...) in log.LogFormatPlugin	At log.LogBatchReader (line 155): "Not quite sure I understand meaning of such reader."
1	Rename Variable	capturingGroups to groupCount at frameworkBuilder(...) in log.LogFormatPlugin	
1	Extract Method	hasSchema() from getFieldNames() in log.LogFormatConfig	
1	Extract Class	log.LogBatchReader.LogReaderConfig from log.LogBatchReader	
8	Move Attribute	COLUMNS_COL from columns.ColumnsArrayManager to columns.ColumnsScanFramework LogFormatPlugin from LogFormatPlugin.LogReaderFactory to LogBatchReader.LogReaderConfig pattern from LogFormatPlugin.LogReaderFactory to LogBatchReader.LogReaderConfig schema from LogFormatPlugin.LogReaderFactory to LogBatchReader.LogReaderConfig maxErrors from LogFormatPlugin.LogReaderFactory to LogBatchReader.LogReaderConfig pattern from log.LogBatchReader to LogBatchReader.LogReaderConfig schema from log.LogBatchReader to LogBatchReader.LogReaderConfig maxErrors from log.LogBatchReader to LogBatchReader.LogReaderConfig	
1	Extract and Move Method	allowOtherCols(...) from frameworkBuilder(...) in log.LogFormatPlugin to ColumnsScanFramework.ColumnsScanBuilder	
1	Move Method	loadVectors(...) from log.LogBatchReader to loadVectors(...) from LogBatchReader.ScalarGroupWriter	

Table 13: Refactorings detected by *RefactoringMiner* and review comments of `https://github.com/apache/drill/pull/1807/commits/b1a5446174485c55a162771b1e804a0587eef361`

#Refactorings	Refactoring Type(s)	Targets	Review comment(s)
8	Change Variable Type	expectedSchema:BatchSchema to expectedSchema:TupleMetadata at testStarQueryNoSchema() in log.TestLogReader expectedSchema:BatchSchema to expectedSchema:TupleMetadata at testSomeFieldsQueryNoSchema() in log.TestLogReader expectedSchema:BatchSchema to expectedSchema:TupleMetadata at testProvidedSchema() in log.TestLogReader expectedSchema:BatchSchema to expectedSchema:TupleMetadata at testSchemaOnlyNoCols() in log.TestLogReader expectedSchema:BatchSchema to expectedSchema:TupleMetadata at testSchemaOnlyWithCols() in log.TestLogReader expectedSchema:BatchSchema to expectedSchema:TupleMetadata at testSchemaOnlyWithMissingCols() in log.TestLogReader expectedSchema:BatchSchema to expectedSchema:TupleMetadata at testTableFunction() in log.TestLogReader expectedSchema:BatchSchema to expectedSchema:TupleMetadata at testTableFunctionWithSchema() in log.TestLogReader expectedSchema:BatchSchema to expectedSchema:TupleMetadata at testTableFunctionWithConfigAndSchema() in log.TestLogReader	–

1.3 Preliminary data analysis

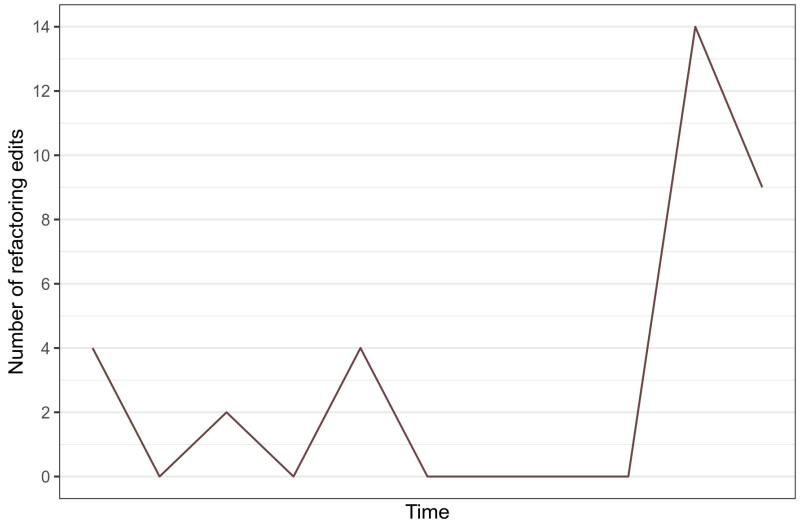
1.3.1 Quantitative analysis

In order to clarify, this analysis considers data from second to the last commit and the review comments from the first one of the selected PR.

RQ_1. How common are refactoring-inducing pull requests?

In the PR, 32 refactoring edits, of multiple types (table 3–13), were performed in 45% of the commits. From figure 1, we can also observe an irregular frequency of refactoring edits by commit over time. Initially there are refactoring edits, followed by an interchange period between 0 and few refactoring edits, and a sharp increase in the end.

Figure 1: Number of refactoring edits over time



RQ_2. What refactoring types often take place in pull requests?

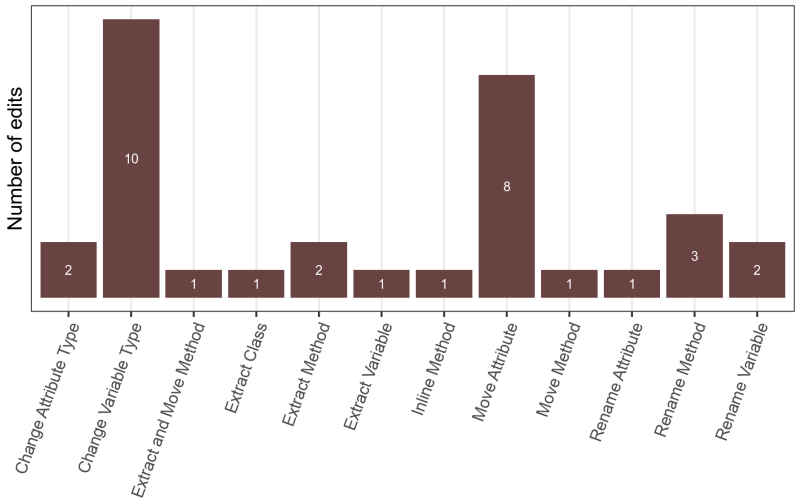
In accordance with figure 2, we can see that most of the refactorings are of the *Move* and *Change* kinds. In particular, *Change Variable Type* and *Move Attribute* amount 56,2% of refactoring edits.

Main outcome
✓ The number of edits (table 3–13) and multiple refactoring types (figure 2) boost the idea of exploring the refactoring-inducing code review.
✓ The irregularity in the frequency of refactorings by commit is also interesting for further investigation (figure 1).

RQ_3. What characterize these refactoring edits?

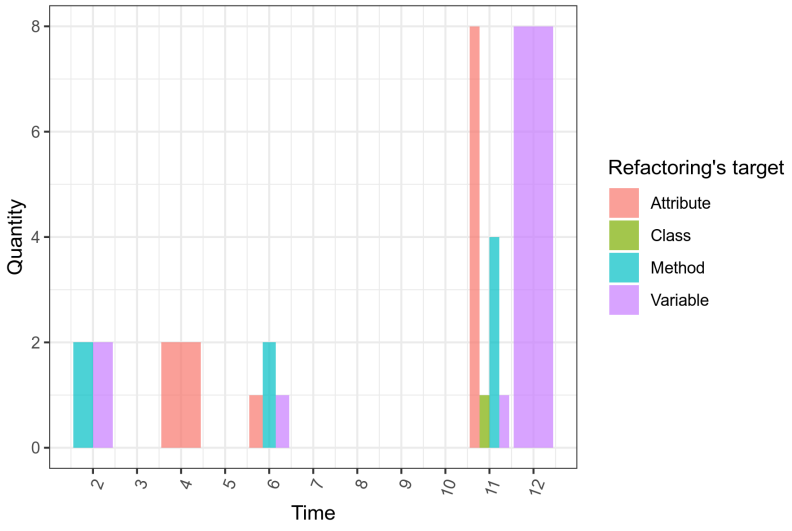
As we can see from figure 3, 71,9% of refactorings' targets are attribute and variable, 25% are

Figure 2: Number of edits per type of refactoring



methods, and only 3,1% are classes.

Figure 3: Number of edits by refactoring's target over time



RQ$_4$. How do these refactorings improve code maintainability?

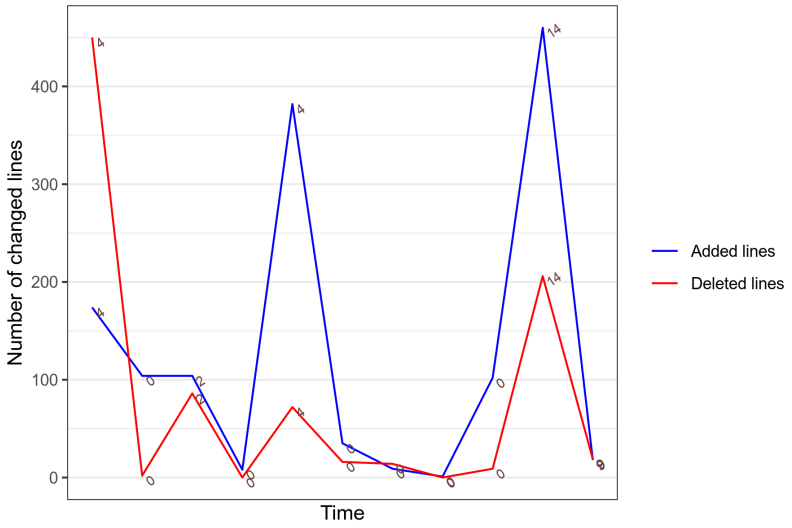
For this brief, we consider one negatively correlated metric to readability, which impacts maintainability: *lines of code*, in terms of the number of added lines and deleted ones in each commit.

In commits with 0 refactorings, from figure 4, we can see the "number of added lines greater

than deleted ones" pattern, except to the eighth one. In commits with refactoring edits, we found the "number of added lines greater than deleted ones" pattern, except to the second and twelfth ones.

In total, the same pattern is also present if we consider the total number of added and deleted lines at commits without/with refactorings (259 (+) and 41 (–) and 1,138 (+) and 833(–), respectively). Thus, the changes in the selected pull request added lines to the code.

Figure 4: Number of changed lines over time



Main outcome

✓ In general, the changes added lines to the code. This result can leverage further investigation on the correlation between refactoring edits and the number of changed lines.

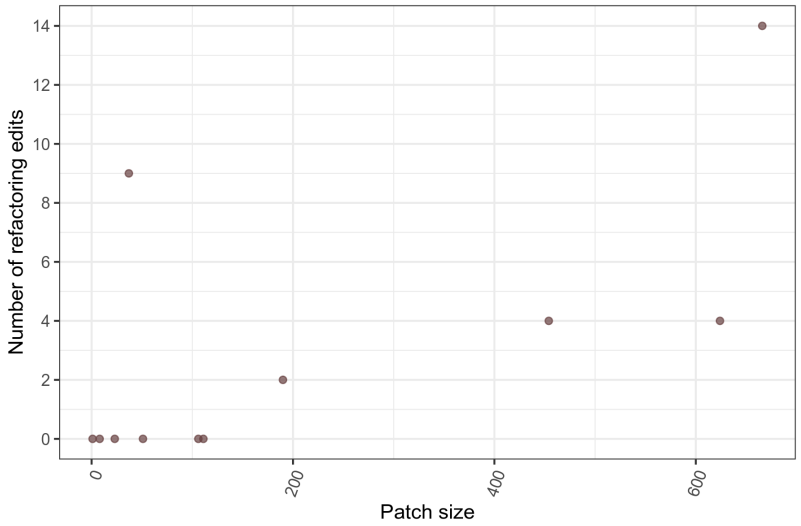
RQ$_5$. How do refactoring-inducing pull requests compare to non-refactoring-inducing ones?

Since the status of the selected PR was *open*, we could not verify how the refactoring edits prevent bugs in the future. However, we collected some code review quality factors, as follows:

- patch size (= lines of changed code per commit),
- number of modified files,
- number of reviewers,
- review response time, and
- length of discussion (= number of comments).

As figure 5 shows, commits without refactorings have a lower number of changed lines (less than 104) than commits with refactoring edits, except in the twelfth commit. At eleventh commit, there are 666 changed lines and seven types of applied refactorings while at twelfth, 37 changed lines and one type of applied refactorings.

Figure 5: Number of refactoring edits per patch size



Main outcome
✓ A suggestion of further investigation on the correlation between the number of refactoring edits and the number of changed lines.

In accordance with figure 6, in spite of the twelfth commit, with 9 refactoring edits of the same type, the number of modified files increases with the number of refactoring edits.

Main outcome
✓ It is interesting to investigate a potential correlation between the number of refactoring edits and of modified files.

In the PR, there is only one reviewer. In terms of review response time and length of discussion, figure 7 shows the response time in days, and its respective number of review comments.

We can realize that commits' conversations with four and five review comments were resolved in until one day, while there are conversations with one and two review comments not resolved yet, which last for more than 13 days. So, we explored some details regarding these two groups: commits with resolved and not resolved conversations, as shown in figure 8 and 9.

Figure 6: Number of modified files per refactoring edits

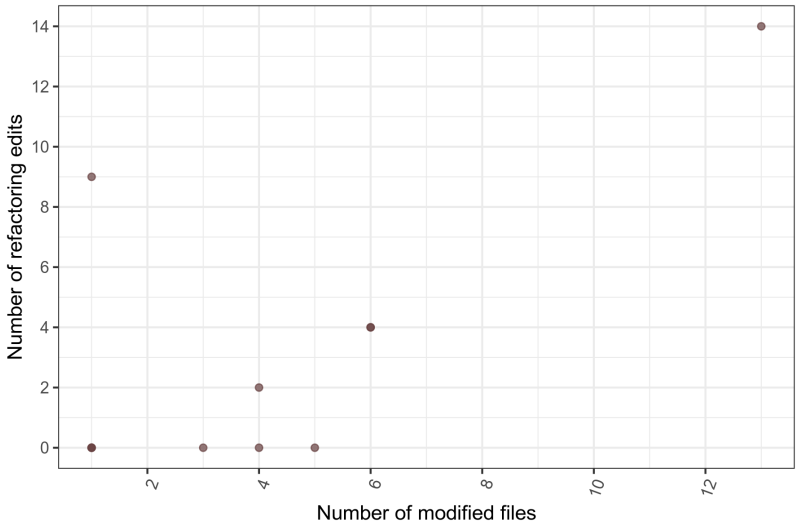
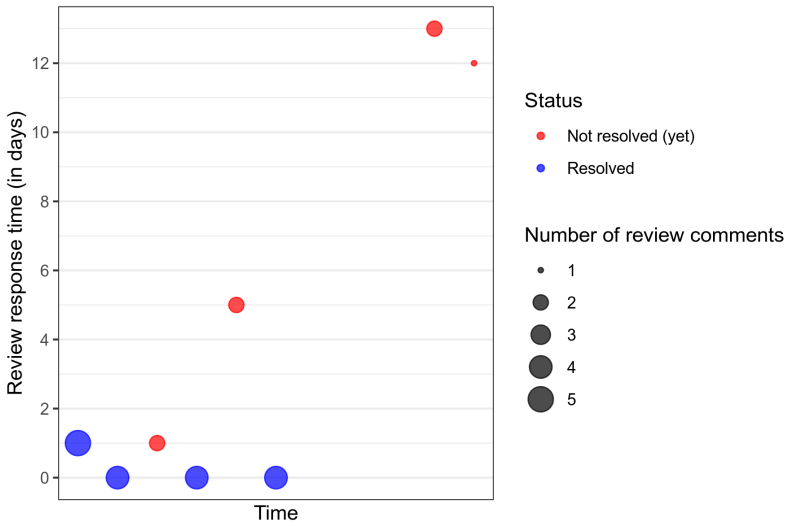


Figure 7: Review response time per commit



From figure 8, we can note that all resolved conversations are related to commits with refactoring edits. As a complement, in figure 10, we found five review comments on a commit with 12 edits (in which, classes are the main target), besides of commits with four review comments (in which, methods are the main refactorings' targets). In particular, in the sixth commit, we found an author's response to one of the reviewer's comments, and it is probable that one of the comments had been driven by a refactoring edit (see figure 13).

From figure 9, we can see that not resolved conversations occur at commits without refactorings and at one with 14 edits of different kinds (*Extract*, *Extract and Move*, *Inline*, *Move*, and

Figure 8: Number of refactoring edits per resolved commits' conversations

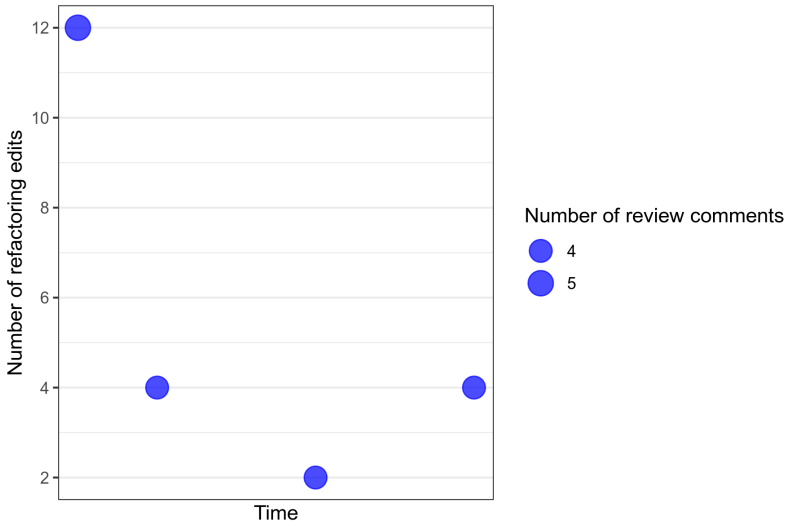
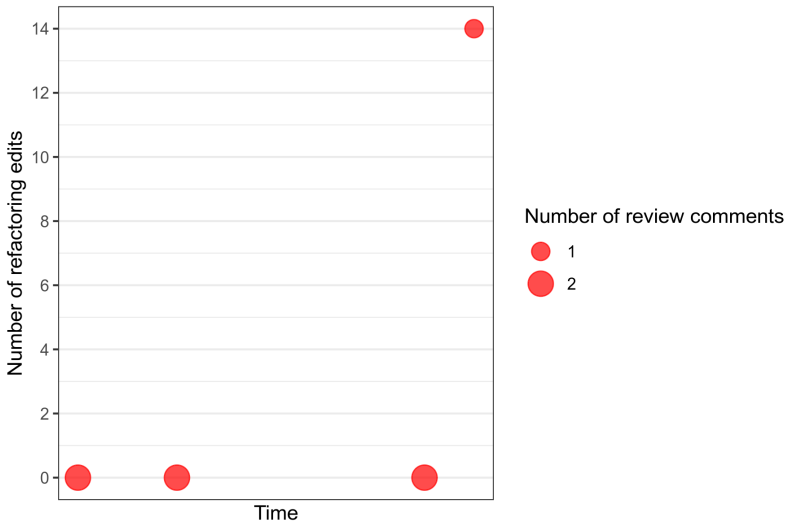


Figure 9: Number of refactoring edits per not resolved commits' conversations



Rename), as shown in 12, and distinct targets (see figure 11. For both third and fifth commits (both with 0 refactorings) we found an author's response to one of the reviewer's comments.

In addition, it is likely that the third commit's review comments had driven refactoring edits in the sixth and twelfth commits (see figure 13), and that the refactoring performed in the eleventh commit had been motivated by a review comment (see figure 12).

Figure 10: Number of refactoring edits per commit and refactoring's target (Resolved status)

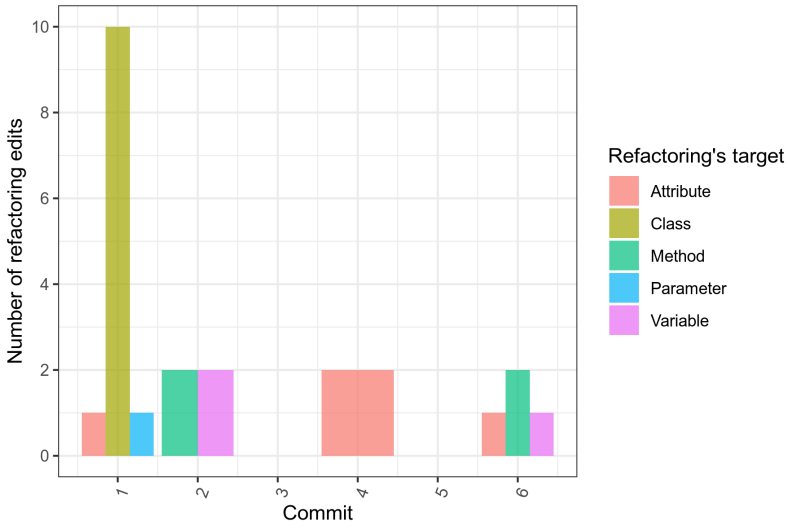
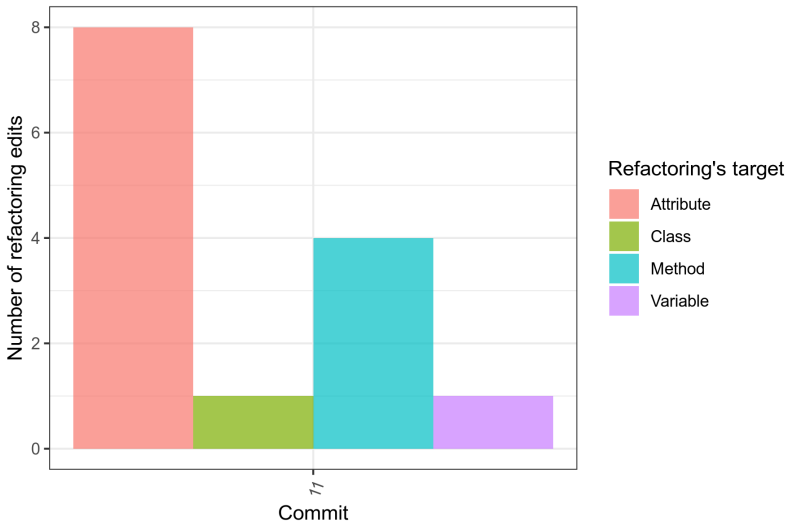


Figure 11: Number of refactoring edits per commit and refactoring's target (Not resolved status)



Main outcome

✓ All conversations in commits with refactoring edits are resolved (figure 8). Thus, it is relevant an in-depth exploration about the relationship between the review comments and refactoring edits.

1.3.2 Qualitative analysis

Through manual inspection of the detected refactorings and review comments of table 2–13, we found (in potential) refactorings induced by reviewing (figure 12) and review comments left after the refactoring edits (figure 13). As we can see, the third commit induced ten refactoring edits in 2 different commits, while two *Extract* refactorings motivated review comments.

Figure 12: Refactorings potentially induced by reviewing

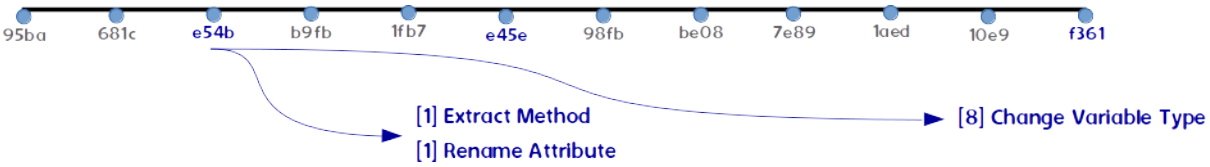
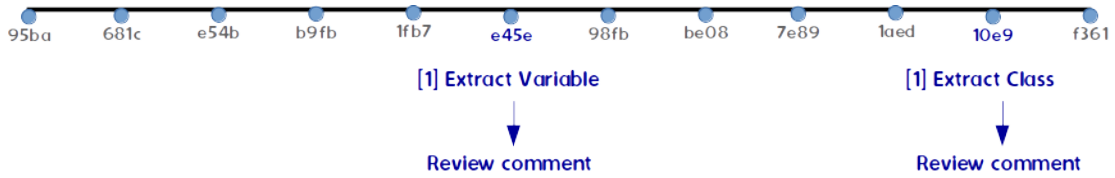


Figure 13: Review comments potentially induced by refactorings



The inspection strategy consisted of the search for keywords, for instance, target names, and "format" and "schema" terms from table 4) mentioned in the review comments and description of the detected refactorings.

Curiously, we found some expressions in review comments that might have induced the refactoring edits in the subsequent commits (table 6): "*... it makes sense to replace list schema parameter with String and rename it to avoid clash...*", and a refactoring edit that might have induced the review comment (table 7): "*... I think you should use different naming, ...*".

Main outcome
✓ The possibility of refactorings induced by reviewing and review comments left after the refactoring edits suggest that a qualitative analysis may reinforce the potential evidence of refactoring-inducing code review.

2 Some *open* pull requests

In order to get a better understand of the data, we carried out another brief "case history" on this selected unit of analysis: Apache/drill project at GitHub[2].

[2] `https://github.com/apache/drill`

2.1 Selection of pull requests

On July 24th, we gathered the *open* pull requests, summarized in table 14. To clarify, from 67 open pull requests, we only selected those with more than two commits, in total 24 ones.

Table 14: A few attributes of the selected *open* pull requests

PR number	#Commits	#Files changed	#Patch size	#Reviewers[1]	#Comments[2]	#Refactorings	#Refactoring types	Creation date	First comment date
1787	2	3	29	1	2	0	NA	2019-05-17 05:36	2019-05-21 14:39
1779	2	6	101	3	13	0	NA	2019-05-02 18:05	2019-05-02 19:37
1778	14	28	2,562	–	1	6	2	2019-05-02 09:25	2019-05-15 06:47
1762	7	108	6,853	1	20	204	18	2019-04-21 23:04	2019-05-01 21:36
1750	2	12	848	1	2	7	4	2019-04-15 14:45	2019-04-15 14:48
1749	7	17	1,065	1	3	0	NA	2019-05-13 14:56	2019-05-13 15:00
1746	4	38	208	2	12	0	NA	2019-04-11 15:59	2019-04-12 05:25
1710	2	2	4	3	3	0	NA	2019-03-21 00:51	2019-03-21 03:02
1662	3	23	574	1	14	9	2	2019-02-27 10:09	2019-02-28 04:40
1606	10	12	528	1	37	12	5	2019-01-09 01:58	2019-01-10 06:52
1500	47	46	6,915	1	48	466	28	2018-10-15 09:58	2018-11-05 05:49
779	17	75	4,091	–	1	14	7	2017-03-17 07:18	2017-06-22 19:33
617	2	2	5	–	1	118	9	2016-10-14 14:34	2018-06-08 19:45
565	3	49	1,067	2	19	7	6	2016-08-10 20:01	2016-10-05 22:16
549	2	4	293	–	18	1	1	2016-07-20 12:20	2016-07-21 22:41
489	8	48	1,825	–	9	74	7	2016-04-28 13:45	2016-05-04 14:58
446	2	16	174	–	13	4	2	2016-03-26 16:43	2016-03-26 18:47
429	4	78	1,621	–	16	21	8	2016-03-13 21:16	2016-03-14 00:07
398	5	722	29,147	–	5	456	13	2016-02-29 22:48	2016-03-11 22:20
335	3	18	790	–	3	176	19	2016-01-22 11:04	2017-11-17 15:49
324	2	3	51	–	1	0	NA	2016-01-15 11:42	2018-06-01 18:14
136	4	3	12,329	–	7	0	NA	2015-08-26 19:57	2015-08-26 20:21
135	5	114	1,933	–	4	63	12	2015-08-26 19:13	2015-09-08 18:46
100	24	34	4,119	–	2	80	14	2015-08-03 02:52	2018-06-01 16:35

[1] *A reviewer is a person who accepted an author's invitation to review the proposed changes in a pull request.*

[2] *Review comments are left by collaborators (reviewers or not) on specific sections of a file on a PR's 'Files changed' tab as part of a pull request review. This number also contains the number of changes-related comments left on the 'Conversation' tab.*

2.2 *RefactoringMiner* execution

The results, based on the *RefactoringMining* version available in July 24th, are summarized in table 15.

Table 15: Refactorings detected by *RefactoringMiner* on the selected open pull requests

#PR Number	#Refactorings	Refactoring Type(s)
1787	0	–
1779	0	–
1778	5	Change Return Type
	1	Extract Variable
1762*	12	Change Attribute Type
	44	Change Parameter Type
	16	Change Return Type
	25	Change Variable Type
	6	Extract Method
	1	Extract Superclass
	5	Extract Variable
	1	Inline Variable
	1	Move and Rename Class
	4	Move Attribute
	2	Move Class
	4	Pull Up Method
	11	Rename Attribute
	2	Rename Class
	25	Rename Method
	30	Rename Parameter
	11	Rename Variable
	4	Replace Variable with Attribute
1750*	1	Extract Variable
	1	Inline Variable
	2	Rename Parameter
	3	Rename Variable
1749	0	–
1746	0	–
1710	0	–
1662*	8	Rename Parameter
	1	Rename Variable
1606	1	Change Parameter Type
	1	Change Variable Type
	6	Rename Attribute
	2	Rename Method

Continued on next page

Table 15 – continued from previous page

#PR Number	#Refactorings	Refactoring Type(s)
	2	Rename Variable
1500**	19	Change Attribute Type
	55	Change Parameter Type
	28	Change Return Type
	29	Change Variable Type
	9	Extract and Move Method
	14	Extract Class
	31	Extract Method
	1	Extract Subclass
	4	Extract Superclass
	10	Extract Variable
	24	Inline Method
	4	Inline Variable
	4	Merge Attribute
	1	Merge Parameter
	3	Move and Rename Class
	33	Move Attribute
	11	Move Class
	35	Move Method
	1	Move Source Folder
	6	Parameterize Variable
	3	Pull Up Method
	9	Push Down Method
	19	Rename Attribute
	3	Rename Class
	28	Rename Method
	52	Rename Parameter
	27	Rename Variable
	3	Replace Variable with Attribute
779*	4	Change Parameter Type
	1	Change Return Type
	2	Change Variable Type
	1	Move Attribute
	1	Move Class
	2	Rename Method
	3	Rename Variable
617*	5	Change Parameter Type
	14	Change Return Type
	1	Extract Method
	1	Move Attribute
	1	Move Method

Continued on next page

Table 15 – continued from previous page

#PR Number	#Refactorings	Refactoring Type(s)
	15	Rename Attribute
	1	Rename Class
	73	Rename Method
	7	Rename Parameter
565*	1	Change Attribute Type
	1	Change Parameter Type
	2	Change Variable Type
	1	Extract and Move Method
	1	Move Method
	1	Rename Method
549	1	Extract and Move Method
489*	1	Change Attribute Type
	21	Extract Method
	1	Extract Variable
	48	Move Attribute
	1	Rename Attribute
	1	Rename Parameter
	1	Rename Variable
446	1	Change Attribute Type
	3	Rename Attribute
429*	2	Extract Variable
	1	Merge Parameter
	1	Move Class
	1	Move Method
	3	Rename Attribute
	6	Rename Method
	5	Rename Parameter
	2	Rename Variable
398*	138	Change Attribute Type
	1	Change Package
	94	Change Parameter Type
	44	Change Return Type
	158	Change Variable Type
	4	Extract Variable
	1	Move and Rename Class
	1	Move Attribute
	4	Move Class
	1	Move Method
	1	Rename Class
	1	Rename Method

Continued on next page

Table 15 – continued from previous page

#PR Number	#Refactorings	Refactoring Type(s)
	8	Rename Variable
335	25	Change Attribute Type
	39	Change Parameter Type
	18	Change Return Type
	26	Change Variable Type
	1	Extract and Move Method
	1	Extract Class
	10	Extract Method
	2	Extract Subclass
	2	Extract Superclass
	5	Extract Variable
	1	Inline Variable
	10	Move Attribute
	7	Move Method
	6	Push Down Method
	4	Rename Attribute
	1	Rename Class
	4	Rename Method
	10	Rename Parameter
	4	Rename Variable
324	0	–
136	0	–
135	3	Change Attribute Type
	3	Change Parameter Type
	1	Extract Class
	1	Extract Superclass
	1	Inline Method
	2	Move and Rename Class
	9	Move Attribute
	1	Move Class
	33	Move Method
	6	Rename Method
	2	Rename Parameter
	1	Rename Variable
100	14	Change Attribute Type
	16	Change Parameter Type
	16	Change Return Type
	6	Change Variable Type
	1	Extract and Move Method
	1	Extract Method
	1	Extract Variable

Continued on next page

Table 15 – continued from previous page

#PR Number	#Refactorings	Refactoring Type(s)
	1	Merge Variable
	1	Move Method
	3	Parameterize Variable
	3	Rename Attribute
	11	Rename Method
	4	Rename Parameter
	2	Rename Variable

*The number of refactoring edits include that ones occurred at the first commit.
**One of the commits has no content (`https://github.com/apache/drill/pull/1500/commits/5629136f490b39550dd1e5608d89e71a7e823052`), thus it has been ignored by *RefactoringMiner*.

2.3 Preliminary data analysis

2.3.1 Quantitative analysis

RQ_1. How common are refactoring-inducing pull requests?

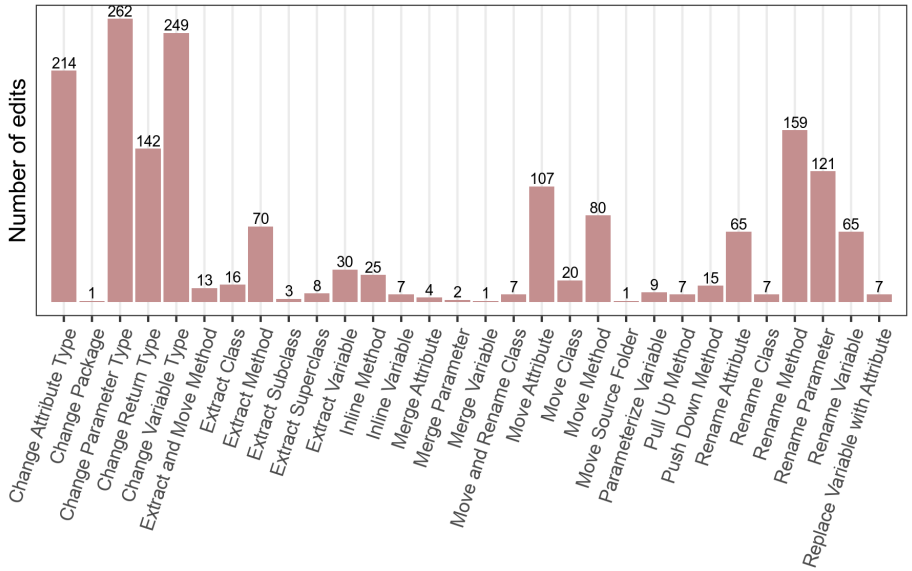
We found refactoring edits, of multiple types (table 15), in 70,8% of the selected open PRs (a total of 1,717 edits, median 21 and mean 101). From this result, 52,9% of them has refactoring edits since their first commit.

Main outcome
✓ The number of the found refactoring edits boosts the investigation idea regarding refactoring-inducing code review.

RQ_2. What refactoring types often take place in pull requests?

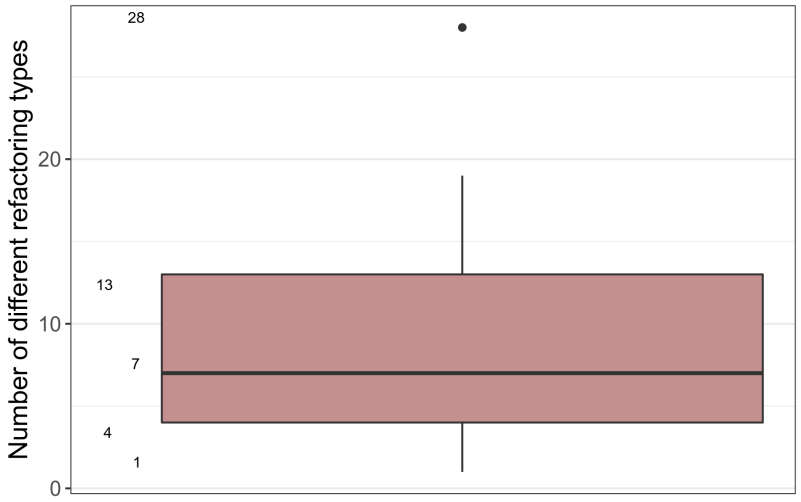
In accordance with figure 14, we can see that most of the refactorings are of the *Change Type*, which amounts 50,5% of all edits, and *Rename* kinds. We found 30 different types of refactoring, from which, 15,2% of edits are of the *Change Parameter Type* while *Change Package*, *Merge Variable*, and *Move Source Folder* represent together only 0,2% of edits.

Figure 14: Number of edits by refactoring type



Moreover, from figure 15, we can observe that the number of different refactoring types ranges from one to 28 per pull request. In until 50% of pull requests occurs less than seven different types of refactoring, and until 75% this number is less than 13.

Figure 15: Number of different refactoring types by pull request



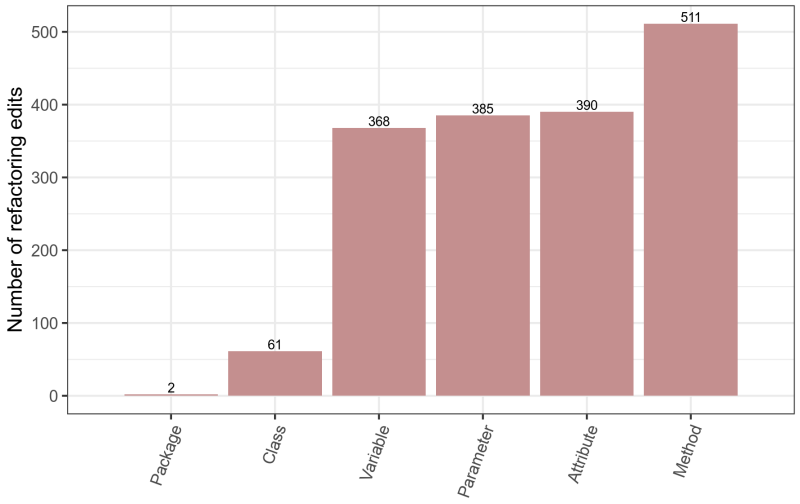
Main outcome

✓ The significant number of different refactoring types (30), in relation to the Fowler's catalog, support the exploration about refactoring-inducing code review practice.

RQ$_3$. What characterize these refactoring edits?

From figure 16, we can see that 66,6% of refactorings' targets are attributes, parameters and variables, 29,8% are methods, 3,5% are classes, and only 0,1% are packages. In addition, we can realize an increase from coarse-grained to fine-grained contexts in terms of the refactoring edits (package → class → method → attribute/parameter/variable).

Figure 16: Targets of the refactoring edits



Main outcome

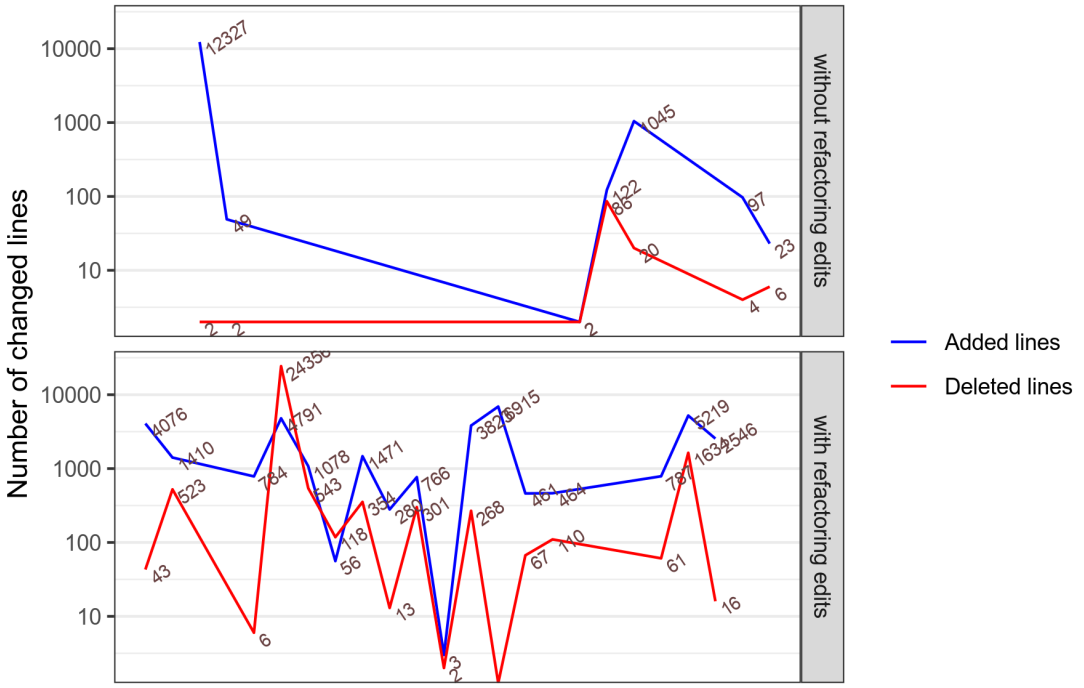
✓ This increasing order in the refactoring targets' granularity suggests further investigation regarding, for instance, what are the impacts towards future bugs.

RQ$_4$. How do these refactorings improve code maintainability?

For this brief, we consider one negatively correlated metric to *readability*, which impacts maintainability: *lines of code*, in terms of the number of added lines and deleted ones in each pull request.

From figure 17, we can note a "number of added lines greater or equal than deleted ones" pattern in the PRs without refactoring edits, and a "number of added lines greater than deleted ones" pattern in most of PRs with refactorings. In the last case, the exception was found in the PRs 446 and 398, with 4 and 446 refactoring edits, respectively. In short, the proposed changes in the selected PRs added lines to the code (48,595 added lines versus 28,537 deleted ones, in total).

Figure 17: Targets of the refactoring edits



In accordance with table 16, the difference between the number of added code lines and deleted ones is greater in PRs without refactoring edits.

Table 16: A summary of the number of added and deleted code lines in the pull requests

Category	#Added lines	#Deleted lines	Difference
without refactorings	13,665	122	13,543
with refactorings	34,930	28,415	6,515

Main outcome

✓ The proposed changes in the PRs can trend to increase the number of code lines, independent of the presence of refactorings, however, the difference between the number of added lines and deleted ones is lesser in PRs with refactoring edits. Therefore, this suggests an in-depth investigation of the impacts of refactorings to the code maintainability. In particular, lines of code may be an interesting metric to exploration.

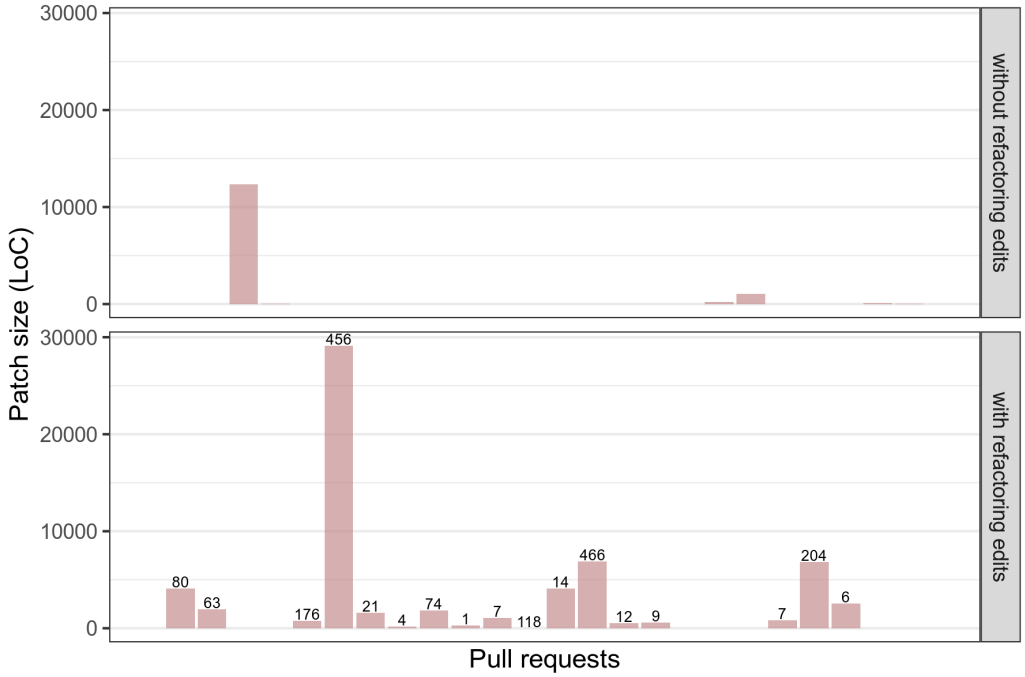
RQ$_5$. How do refactoring-inducing pull requests compare to non-refactoring-inducing ones?

Since the status of the selected pull requests is *open*, we cannot verify how the refactoring edits prevent bugs in the future. However, we collected the following code review quality factors:

- patch size (= lines of changed code),
- number of modified files,
- number of reviewers,
- review response time, and
- length of discussion (= number of comments).

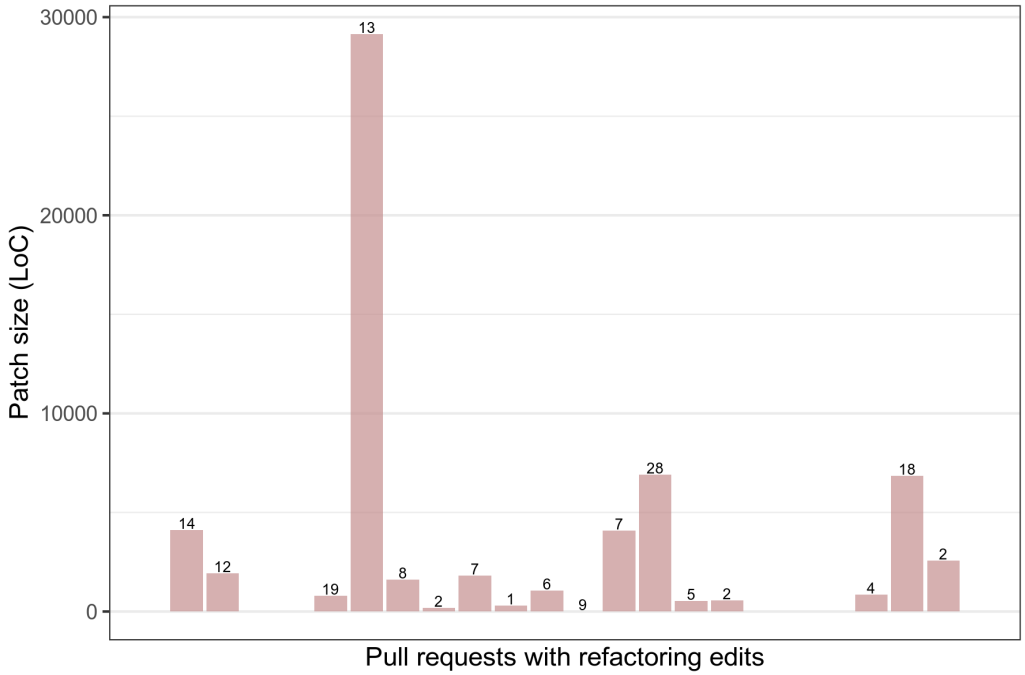
As figure 18 shows, PRs without refactorings have a lower number of changed lines (mean 1,969 and median 101) than PRs with refactoring edits (mean 3,726 and median 1,621).

Figure 18: Patch size (LoC) by pull request (and number of refactoring edits)



For PRs with refactoring edits, figure 19 displays no relationship between the patch size and the number of different refactoring types. In particular, we found a PR with refactoring edits of 9 different types in a patch size 9.

Figure 19: Patch size (LoC) by pull request with refactoring (and number of different types) edits



Main outcome

✓ The patch size by PR visualizations suggest a further investigation on the correlation between the number of refactoring edits and the patch size (figure 18), and no correlation between the patch size and the number of different applied refactoring types (figure 19). This last outcome can be potentially due to the most of refactorings are of the *Change Type* and *Rename* kinds (figure 14).

In terms of modified files, from figure 20, the PRs without refactorings have a lower number of modified files (mean 10,3 and median 3) than PRs with refactoring edits (mean 81,7 and median 34).

Figure 20: Number of modified files by pull request (and number of refactoring edits)

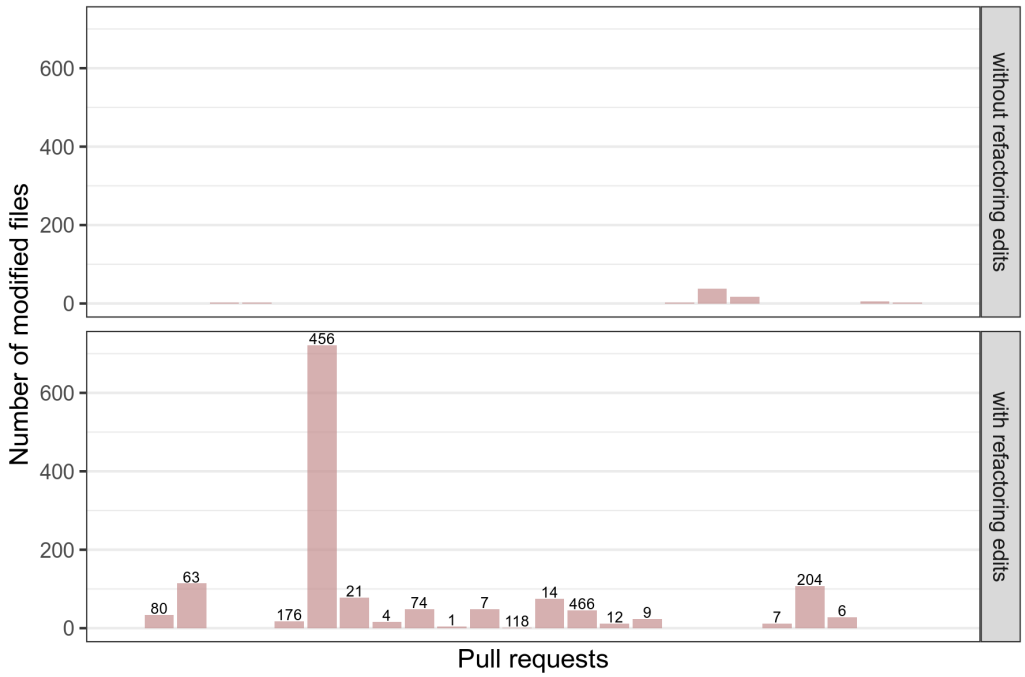
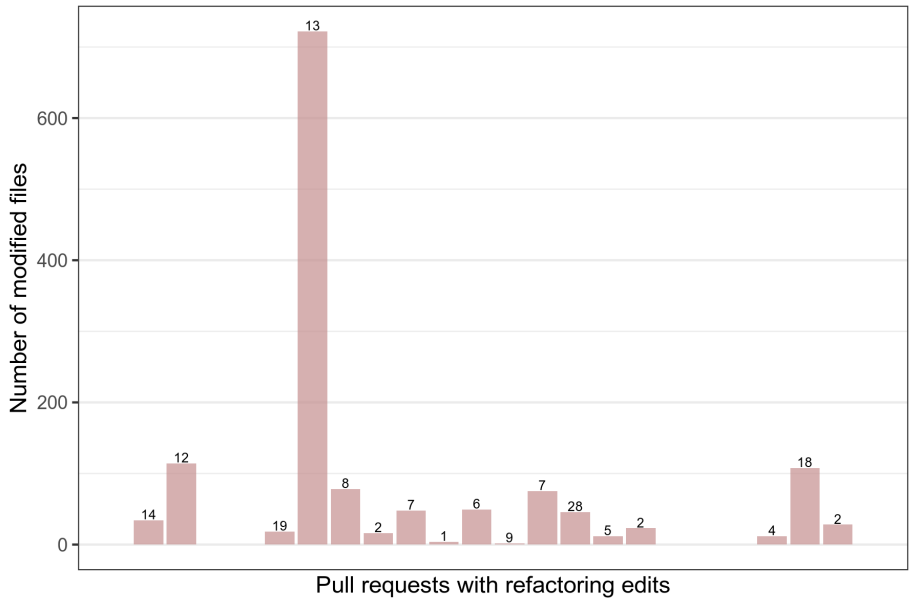


Figure 21 displays no relationship between the number of modified files and the number of different refactoring types for PRs with refactoring edits.

Figure 21: Number of modified files by pull request with refactoring (and number of different types) edits

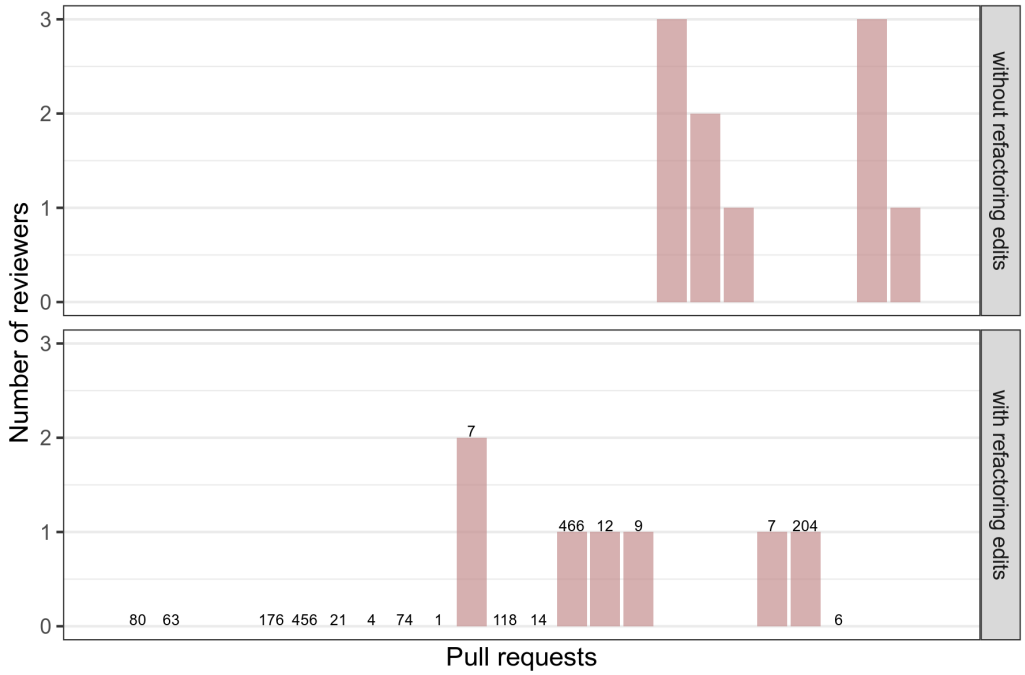


Main outcome

✓ The number of modified files by PR visualizations indicate examination on the correlation between the number of refactoring edits and the number of modified files (figure 20), and no correlation between the number of modified files and the number of different applied refactoring types (figure 21). This last outcome can be potentially explained by the fine-grained context of applied refactorings, whose targets are mainly methods, attributes, parameters, and variables (figure 16).

In accordance with figure 22, the most recent pull requests have had assigned reviewers, precisely from August 2016. In special, 5 of 7 PRs without refactoring edits have from 1 to 3 reviewers; however, only 6 of 17 PRs with refactoring edits, whose patches size are higher than in PRs without ones (figure 18), have had reviewers.

Figure 22: Number of reviewers by pull request with refactoring (and number of different types) edits



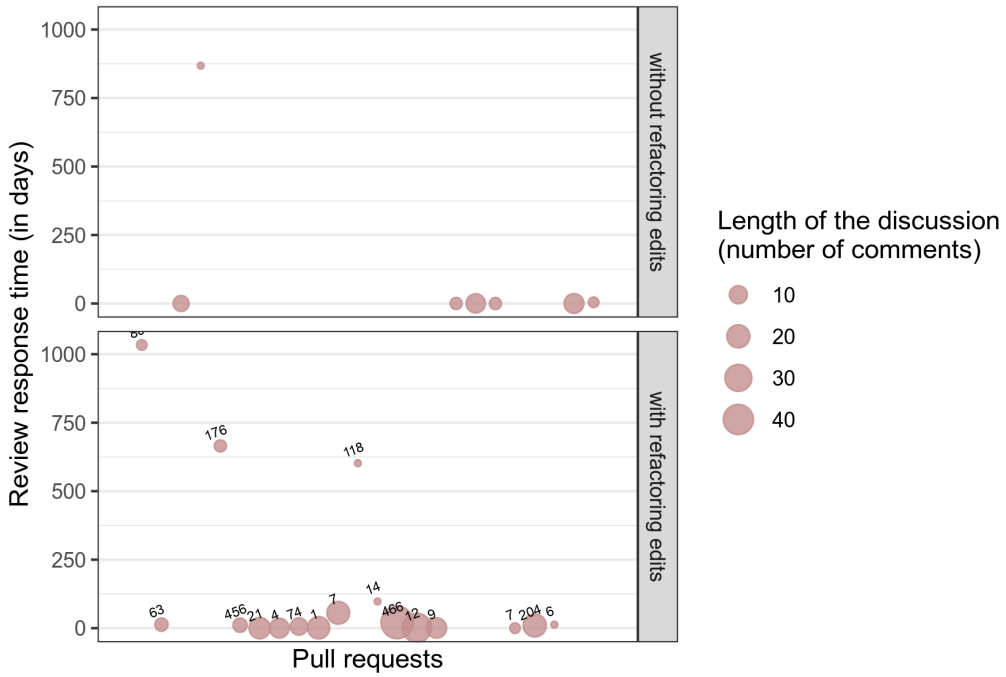
Main outcome

✓ Most of recent PRs have had assigned reviewers, since August 2016. In this context, curiously, all PRs without refactoring edits have had assigned reviewers, but this is not a fact to PRs with refactoring edits, that suggests an in-depth examination of factors.

The review response time, calculated as the difference between the PR's open date and its first comment date, and the length of the discussion are displayed in figure 23. In addition, we realized that only two PRs have comments marked as resolved: 1500 and 1606, with 5 and 1 resolved comments from 48 and 37 comments, respectively.

In PRs without refactorings, we found comments (mean 5,9 and median 3) left in until 4 days, except for the PR 324. An irregular behavior is noted from PRs with refactoring edits, in spite of comments (mean 12,5 and median 9) left in until 10 days, since January 2019. This irregularity is based on a few extreme cases, for instance, only 5 comments were left in PR 398 and 18 ones in PR 549 with 456 and 1 refactorings, respectively.

Figure 23: Review response time by pull request (and number of refactoring edits)

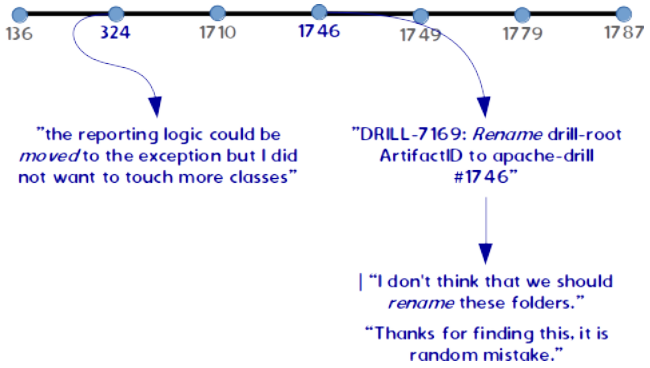


2.3.2 Qualitative analysis

By manual inspection of table 15 and comments in the PRs, based on searching for keywords such as "refac", "mov", "extract", "renam", and reading of comments, we found interesting stuff. In the following figures, we display the PR numbers over time.

From figure 24, we can see two PRs, where a few comments possibly motivated no refactoring edits.

Figure 24: No refactoring edits potentially induced by reviewing



In respect of PRs with refactorings, in 41,2% of them, we found a few edits potentially

induced by reviewing (figure 25) and comments motivated by edits (figure 26). Only the PR 135 indicated the intention of performing refactorings in its description.

Figure 25: Refactoring edits potentially induced by reviewing per pull request

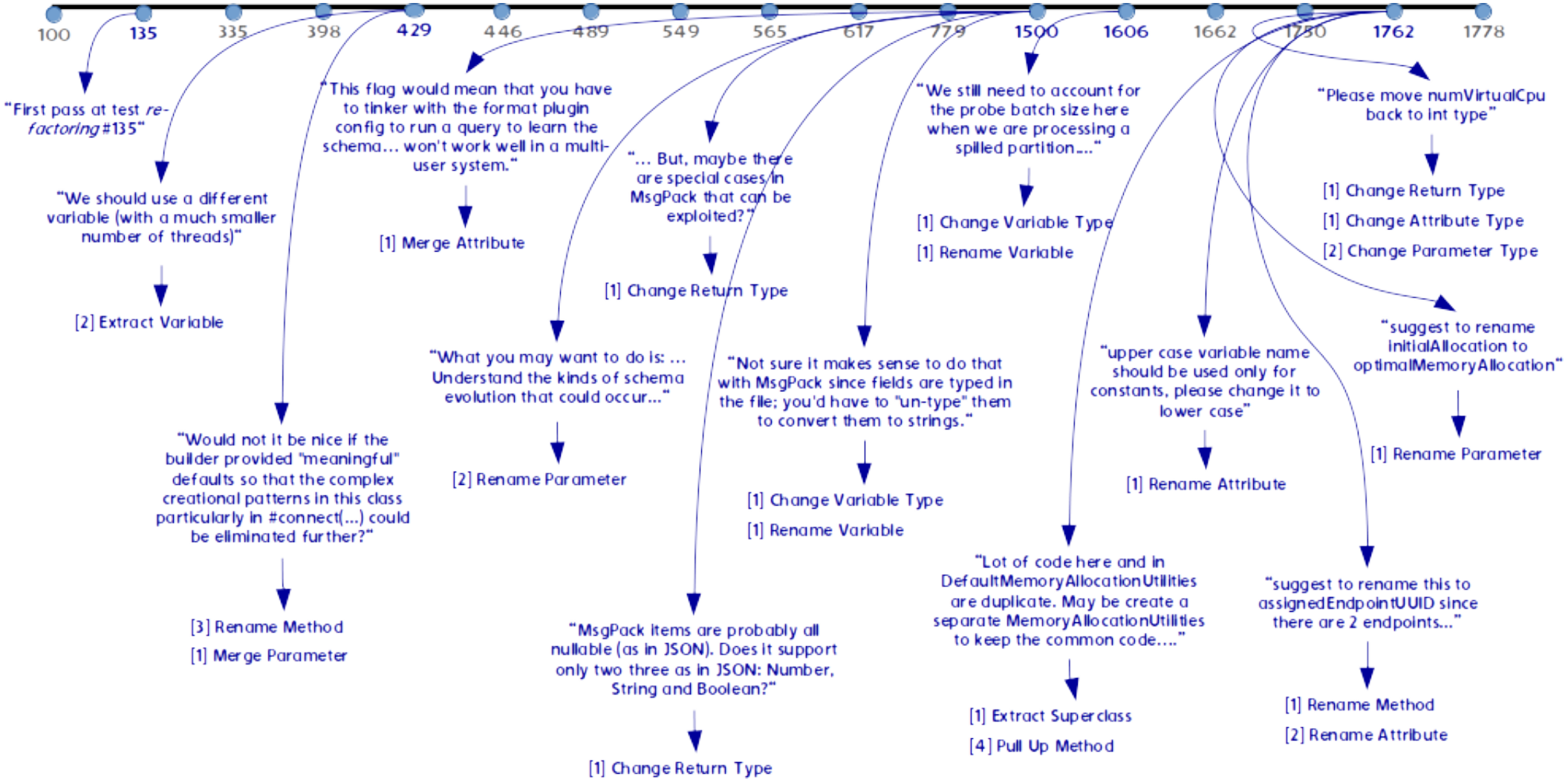
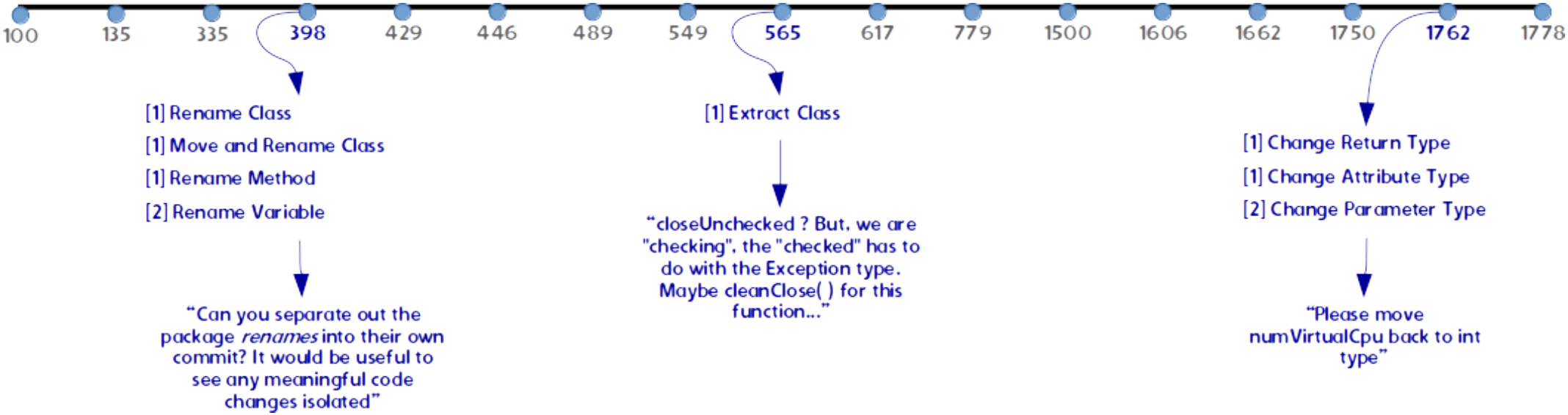


Figure 26: Comments potentially induced by refactoring edits per pull request



Most of these PRs have a number of comments, refactoring edits, different refactoring types, modified files, and patch size greater than their respective median value.

These observations (figure 25-26) follow the pattern found regarding refactoring's targets (figure 16), because of the increasing order of the related comments from a coarse-grained to fine-grained (class → method → attribute/parameter/variable).

Main outcome
✓ We can realize a greater number of review comments left in PRs with refactoring edits in relation to the without edits, in a general view. This provides a basis for further exploration regarding a potential correlation between the length of discussion and the number of refactoring edits. ✓ Although 1,717 refactoring edits, we found only 39 comments that potentially either were motivated from the refactorings or induced the edits. ✓ It is interesting a further investigation of the observed pattern in relation to comments and refactorings' targets. ✓ The manual inspection required a substantial effort for exploration of PRs' refactoring descriptions and comments, that suggests the need for an automatic strategy in support of our qualitative analysis.