Qualitative Analysis

Round 2, June 12 – July 30, 2021

Topics

Method of Analysis

We adopted a non-probability sampling, *purposive sampling*, to select refactoring-inducing and non-refactoring-inducing PRs, according to specific criteria, for an in-depth investigation of their review comments and discussion while cross-referencing their detected refactoring edits. We also validated these refactorings by checking for false positives and false negatives.

It is worth clarifying that we will follow that strategy until we reach a point of data saturation (when no new information emerges), through rounds of analysis. Accordingly, at each round, we will examine a purposive sample fitting a valuable scenario to the current purposes of the analysis. We chose that sampling strategy because it provides us gaining an in-depth understanding of data by exploring scenarios suitable at each step, in line with emergent patterns or ideas.

Addressing the second sample

We experimentally considered 20 as the size for the second purposive sample: ten refactoring-inducing PRs and ten non-refactoring-inducing PRs that contain only one subsequent commit, randomly selected from 114 refactoring-inducing PRs and 681 non-refactoring-inducing PRs, respectively. Why? Intending to explore PRs consisting of only one refactoring edit, we need to consider such a simple setting, which gives us an opportunity of comparing refactoring-inducing and non-refactoring-inducing PRs (to deal with RQ_1 and RQ_2).

Then, each analyst apart examined all sample's PRs. Next, one analyst checked all individual judgments in order to achieve a concluding judgment concerning the sample. As decision criteria, we considered the agreement of responses by at least two analysts. It is worth clarifying that, in such subjective decision-making, we considered the refactoring-inducement in settings

where review comments either explicitly suggested refactoring edits or left any actionable recommendation that induced refactoring. We give examples as follows.

Sample 2

- 11 refactoring-inducing PRs
- 9 non-refactoring-inducing PRs
- Number of review comments: 87
- Number of discussing comments: 60
- Number of subsequent commits: 20
- Number of refactoring edits: 11

Concluding Remarks

(1) 9/11 (81.2%) of refactoring-inducing PRs are due to code review.

Conclusion: We found an almost similar result in round 1 – 10/13 (76.9%).

(2) 1/10 (10%) of "previously categorized as" refactoring-inducing PR was a false positive (renaming a method's name for the same name).

(3) 2/10 (20%) of "previously categorized as" non-refactoring-inducing PRs were false negatives (all refactoring edits due to code review): RefactoringMiner 1.0 should but did not detect 2 refactoring edits (*Rename Method* and *Inline Variable*).

(4) Accordingly (2) and (3), TP = 9, FP = 1, and FN = 2 (precision 90%, recall 81.8,%). As a whole, in sample 1 and sample 2, RefactoringMiner presented precision 98.7% and recall 94.9%.

(5) 0/11 (0%) refactoring-inducing PRs present self-affirmed refactorings. Such edits are explicit in the commit message.

Conclusion: It seems that self-affirmed refactorings are rare in our sample (considering rounds 1 and 2), since only 1/24 (4.2%) of refactoring-inducing PRs present self-affirmed refactorings.

(6) Both refactoring-inducing and non-refactoring-inducing PRs comprise the three primary types of changes (adaptive, corrective, and perfective). Our classification follows the descriptions given by [Swanson, 1976] and the identification method provided by [Mockus and Votta, 2000]. To clarify, we explored PR descriptions and commit messages by searching for keywords that could denote the type of changes; for instance, keywords such as fix and correct indicate corrective changes.

Conclusion: It seems that there is no relationship between the type of the change and refactoring-inducement.

In specific, in sample 1, except in Flink #7970, we found refactoring edits led by PR's author in refactoring-inducing PRs, which changes are adaptive (Samza #1030, Flink #7945, Dubbo #2279)

and perfective (Beam #4460). In sample 2, such a pattern persists in refactoring-inducing PRs: adaptive changes in Incubator-pinot #479 and perfective changes in Kafka #5423.

Therefore, we can further investigate the existence of such a pattern in additional refactoring-inducing PRs.

(7) Only 1 non-refactoring-inducing PRs present a code review bot. A repository's code review bot is easily detectable since it left a comment in the PR, including the bot commands performed. For instance, the *flinkbot* checks the PR description, whether a PR needs attention from a specific reviewer, the architecture, and the overall code quality.

Conclusion: It seems that there is no relationship between code review bot running and refactoring-inducement since we found both refactoring-inducing and non-refactoring-inducing PRs that ran code review bots.

(8) We found none PR description in both refactoring-inducing and non-refactoring-inducing PRs.

Conclusion: It seems that there is no relationship between PR description and refactoring-inducement.

(9) We found self-affirmed minor PR (in which, a title or a description self-declares a minor PR) in 2/24 refactoring-inducing PR (Kafka #5194 in sample 1; Kafka #5423 in sample 2) and in 2/16 non-refactoring-inducing PRs (Brooklyn-server #411, Kafka #5111 in sample 1). Moreover, we found self-affirmed minor review comments (in which, a reviewer declares a review comment as minor) in 4/24 refactoring-inducing PR (Flink #7945 in sample 1; Brooklyn-server #1049, Kafka #5784, Dubbo #3185 in sample 2) and in 4/16 non-refactoring-inducing PRs (Brooklyn-server #4111, Flink #91 in sample 1; Beam #5785, Flink #9451 in sample 2).

(10) By observing the age of sample's PRs (concerning their repository's beginning), we found no difference regarding refactoring-inducing and non-refactoring-inducing PRs. Refactoring-inducing PRs have 4.7 years on average, 5 years on median, whereas non-refactoring-inducing PRs have 5 years on average and on median. We carried out such an action to explore some potential particularity of refactoring edits against repositories' maturity.

(11) As we argued in round 1, reviewers present experience in code review as from 2016[1]. Accordingly, there may be more experienced reviewers than noted.

(12) In refactoring-inducing PRs, in which code review induced refactorings, *Rename* (3/9) and *Extract* (3/9) instances are the most frequent refactoring kinds, whereas *method* (4/9) and *attribute* (3/9) are the most frequent refactoring targets in sample 2. In refactoring-inducing PRs, in which authors led refactorings, *Extract* (1/2) and *Change Type* (1/2) instances occurred, being *variable* (2/2) the refactoring target.

Conclusion: Sample 1 differs from sample 2 regarding code size and the number of subsequent commits. Thus, when decreasing the PRs magnitude, we realize that the most common refactoring

[1] We believe this is due to the inclusion of more contributing activities on GitHub in 2016, available in https://github.blog/2016-05-19-more-contributions-on-your-profile/.

types include edits that alter code design (e.g., *Extract* instances are more common in sample 2). Maybe it is early to speculate regarding the influences of PR size on refactoring-inducement, even though we can explore refactorings that alter the code design intending to understand the context of inducement (e.g., what is the influence of code size? Do small PRs promote code design changes?).

(13) When considering the number of contributions of the PRs' participants, it seems that:

1. refactoring-inducing PRs authors are less experienced than non-refactoring-inducing PRs authors (see median values in Table 1).
2. reviewers of refactoring-inducing PRs are less experienced than reviewers of non-refactoring-inducing PRs (Table 1, Table 2) – different from that observed in sample 1. When exploring sample 2, we are dealing with less complex code (regarding the number of file changes, added lines, and deleted lines) than in sample 1. For this reason, we speculate that, in such a scenario (sample 2), the reviewer's experience is not so relevant to put forward code improvements.

Conclusion: Based on results from sample 1 and sample 2, we conjecture that non-refactoring-inducing PRs' authors develop less problem-prone code and, therefore, less refactoring-prone code, because they are more experienced.

Table 1 – Stats of PRs' participants by category

Category	Stats	Author	Reviewer
Refactoring-inducing PRs	Average	758.2	956.1
	SD	547.5	974.9
	Median	1032	438
	IQR	1024.5	1909
	Outliers	3699, 28964	10853, 14688
Non-refactoring-inducing PRs	Average	3137	1645.5
	SD	4058.6	1311.3
	Median	1674.5	1639
	IQR	3499	2765
	Outliers	11965	none

Table 2 – Apache roles of PRs' participants by category

Category	Author	Reviewer
Refactoring-inducing PRs	1 committer 1 commiter/PMC	6 PMC 4 commiters/PMC
Non-refactoring-inducing PRs	4 committer/PMC	6 PMC 5 committers/PMC

Research Questions

- **RQ_1 How are refactoring-inducing and non-refactoring-inducing PR's review comments characterized?**

Refactoring-inducing PRs

Reviews in refactoring-inducing PRs from sample 1, where refactoring edits were induced by code review, address code logic (providing rationale, in a few cases), and suggest improvements (given reasons). In sample 2, the same patterns emerged: reviews ask questions on code logic (Kafka #7132, Beam #4407, Tomee #275) and suggest improvements (Brooklyn-server #1049, Dubbo #3185, Beam #4458, Servicecomb-java-chassis #346); additionally, a few reviews ask regarding *Rename* refactoring (Samza #1051, Kafka #5784).

In sample 1, reviews in refactoring-inducing PRs, where refactoring edits were submitted by the authors, directly address code logic. The same occurs in sample 2 (Kafka #5423); additionally, reviews suggest tests and error checks (Incubator-pinot #479).

Non-refactoring-inducing PRs

In sample 1, reviews in non-refactoring-inducing PRs directly suggest the insertion of tests and address simple code logic (e.g., correct use of a method, range of variable's values, and constant), typos, code aesthetics, and documentation. In sample 2, reviews addressing tests (Kafka #6818, Beam #6050) and simple code logic (e.g., confusing iterator, better argument values, and error message) persist, such as we can note in Kafka #6565, Dubbo #3317. Typos (Beam #7696, Flink #9451) and code aesthetics (Beam #5785) remain discussed in reviews. In addition, reviews also deal with code logic by providing rationale (Kafka #5219, Tinkerpop #524, Kafka #6818). In Beam #5785 and Tinkerpop #524, we observe a few discussions, in turn, won by PR's author. In Beam #6050, PR's author provides clarifications to attend to the reviewer's issues.

Two PRs present no review comments in Java files (Beam #5785, Tinkerpop #524).

Note: We can consider that non-refactoring-inducing PRs, even in Java repositories, present a few non-Java files (e.g., Scala and Python code). Such characteristic is common in 5/16 non-refactoring-inducing PRs examined.

- **RQ_2 What are the differences between refactoring-inducing PRs and non-refactoring-inducing PRs, in terms of review comments?**

By considering sample 1 and sample 2, we found distinct characteristics between reviews in both categories of PRs. In refactoring-inducing PRs, reviews ask questions concerning code logic, provide warnings on good practices of programming, give explanations usually supported by auxiliary code, and provide direct

suggestions of improvement. In both samples, we realize that it is not rare for a PR to only have refactoring suggestions (e.g., Samza #1051, Kafka #5784, Servicecomb-java-chassis #346, Tomee #275, Beam #4458).
Specifically, in a non-refactoring-inducing PR (Tinkerpop #524), there is a case in which a reviewer suggests code inlined in comments. This is not welcome, since PR's author began a discussion, ignoring that suggestion. We speculate that such an approach might affect code ownership, so explaining the behavior of PR's author. However, when reviewers suggest simplification by inlining the code in a comment, such proposals are welcome (Dubbo #3185 and Kafka #5219).

Conclusion: We can speculate that refactoring-related suggestions are less likely to have explanations about why doing it when compared to non-refactoring-related proposals, and that reviews in non-refactoring-inducing PRs address simpler code issues (warnings about something wrong, missing or bug-related, as in Kafka #6565, Dubbo #3317, Beam #5785, Beam #7696, Flink #9451) than in refactoring-inducing PRs; maybe, due to the experience of PRs' authors, reflected in the proper characteristics of the source code, and by providing clarifications and winning discussions.

- **RQ$_3$ How do reviewers suggest refactorings in review comments in refactoring-inducing PRs?**

In sample 2, we realize that reviewers suggest refactoring-related changes in a single comment.
In both examined samples, reviewers inspire/induce refactoring edits by asking questions concerning code logic (using terms such as "should we...?", "can we...?", "maybe..."), by giving warnings on good practices of programming, by providing direct explanations supported by auxiliary code (using structures of the code under analysis, e.g., a class, a method), and by using direct sentences to suggest improvements (e.g., "a method name...", "you can replace to...").
In refactoring-inducing PRs, reviews ask questions concerning code logic (using terms such as "should we...?", "can we...?" , "maybe..."), provide warnings on good practices of programming, give explanations usually supported by auxiliary code (using structures of the code under analysis, e.g., a class, a method), and provide direct suggestions of improvement (using sentences like "a method name...", "you can replace to...").
In both samples, we realize that it is not rare for a PR to only have refactoring suggestions (e.g., Samza #1051, Kafka #5784, Servicecomb-java-chassis #346, Tomee #275, Beam #4458).
Conclusion: It seems that reviewers suggest or inspire refactoring edits when they provide direct sentences of improvement that, in turn, are well-understood by PRs' authors. We conjecture that the experience of reviewers against authors might explain such a pattern.

- **RQ$_4$ Do refactoring suggestions justify the reasons?**

In 2/9 (22.2%) of refactoring-inducing PRs, in which refactoring edits were induced by review, the reviewers are reasonable on the explanation, being concise while providing a rationale regarding a *Split Attribute* (Brooklyn-server #1049) and a *Change Variable Type* (Tomee #275) instances. In the others (7/9), reviewers provide no reason for *Rename* and *Extract* suggestions.

In sample 1, we found reasons in the form of contextualized sentences, assisted by examples.
Conclusion: We speculate that likely there is a relationship between the refactoring type and the provision of a rationale.

- **RQ_5 What is the relationship between refactoring recommendations and actual refactorings in refactoring-inducing PRs?**

When analyzing sample 1 and sample 2, we realize that when the reviewers get straight to the point (by providing either concise comments or succinct rationales), refactoring edits are done as observed in Samza #1051, Beam #4407, Kafka #5784, Servicecomb-java-chassis #346, Beam #4458, Kafka #7132, Dubbo #3185, Brooklyn-server #1049, and Tomee #275 (in these last two, reviewers present reasons).
When there is space for discussion, refactorings may not be done, as it occurs in Kafka #7132, through longer review comments and discussion. In such a PR, a reviewer submits his uncertainty concerning code logic, whereas PR's author clarifies those issues, thus resulting in no discussion-related change.
In sample 2, since we consider only refactoring-inducing PRs consisting of one refactoring edit, we do not investigate refactoring complexity, for instance, by exploring the effects of a refactoring edit (e.g, a Rename Class which promotes Change Type instances, as it occurs in sample 1, Flink #7970).
Conclusion: It seems that review comments induce refactoring edits when review comments (i) provide concise reasons, (ii) give suggestions through code simplifications (Dubbo #3185), or using terms like "Maybe..." (Servicecomb-java-chassis #346, Beam #4458), and (iii) politely question code logic by using terms such as "How about...?" (Samza #1051), and "Should this be...?" (Kafka #5784, Kafka #7132).

Suggestions

- (RQ_1, RQ_2) examine a randomly selected sample of non-refactoring-inducing PRs, and (RQ_3, RQ_4, RQ_5) investigate a sample of refactoring-inducing PRs consisting of inheritance-related refactorings and refactorings changing code design in order to explore whether persist the emergent patterns.

Notes

- **Apache roles[2]:**
  - **Contributor** is a developer who contributes to a project in the form of code or documentation.
  - **Committer** is a contributor who has write access to the code repository.
  - **PMC member** is a committer who has write access to the code repository and can agree and approve/disapprove the changes.

- **Repositories timeline (contribution activities):**
  - tomee – Jan 2006

[2] https://apache.org/foundation/how-it-works.html#roles

- flink – Dec 2010
- brooklyn-server – Jun 2011
- dubbo – Jun 2012
- kafka – Dec 2012
- samza – Aug 2013
- tinkerpop – Sep 2013
- beam – Dec 2014
- incubator-pinot – Oct 2014
- servicecomb-java-chassis – May 2017

Refactoring-inducing PRs:

kafka 6565: 2019 | commit merge | 1 external reference after PR merge | initial refactoring edits | nobot | corrective changes | none PR description | no refactorings (false positive) | review comments on simple issues concerning code logic (confusing iterator, better arguments – e.g., "XXX" was replaced by "A3")

- 1 subcommit, 2 file changes, 894 added lines, 894 deleted lines, 5 days to merge
- Author, contributions since 2014 (2,032)
- Contributor (Apache Kafka PMC) contributions since 2011 (2,468), reviews since 2016
- Reviewer (Apache Kafka committer and PMC), contributions since 2015 (3,401), reviews since 2016
- 5 requested reviews, only two reviewers
- PR opened on Thursday, reviewed on Monday

incubator-pinot 479: 2016 | squash and merge | 1 mention after PR merge | nobot | adaptive changes (adding a new test) | None PR description | **refactoring led by PR's author** | review comments on code logic, provinding reason (questioning assertion instead of a method call) that, in turn, was ignored by PR's author, and regarding adding more tests and error checks | 1 refactoring edit

- 1º subsequent commit: 1 *Extract Variable*
- 1 subcommit, 2 file changes, 46 added lines, 2 deleted lines, 22 days to merge
- Author, contributions since 2010 (1,032)
- Reviewer, contributions since 2012 (124), reviews since 2016
- PR opened on Thursday, reviewed (inducing refactoring) on Thursday

brooklyn-server 1049: 2019 | commit merge | 2 mentions after PR merge | no bot | corrective changes (fixing faults) | **refactorings inspired by code review** | review comment suggest such a refactoring, by providing reason (due to code conventions) | self-affirmed minor review comments | 1 refactoring edit

- 1º subsequent commit: *1 Split Attribute*
- 1 subsequent commit, 1 file change, 5 added lines, 4 deleted lines, 8 days to merge
- Author, contributions since 2012 (3,699)

- Reviewer (Apache Brooklyn PMC), contributions since 2011 (10,853), reviews since 2016
- Reviewer (Apache Brooklyn PMC), contributions since 2011 (822), reviews since 2016
- PR opened on Tuesday, reviewed (inducing refactoring) on Thursday

samza 1051: 2019 | commit merge | no bot | adaptive changes (adding a new feature) | **refactoring induced by code review** | direct review comment questioning to rename a class ("How about..."?) | 1 refactoring edit

- 1º subsequent commit: *1 Rename Class*
- 1 subsequent commit, 3 file changes, 6 added lines, 6 deleted lines, 3 days to merge
- Author, contributions since 2014 (301)
- Reviewer, contributions since 2016 (140), reviews since 2017
- PR opened on Friday, reviewed (inducing refactoring) on Tuesday

beam 4407: 2018 | commit merge | no bot, PR presents a fulfilled checklist for contribution | corrective changes (fixing faults) | **refactoring induced by code review** | direct review comment questioning code logic (*reviewer 1*) and documentation (*reviewer 2*) | 1 refactoring edit

- 1º subsequent commit: *1 Extract Attribute*
- 1 subsequent commit, 1 file change, 6 added lines, 4 deleted lines, 0 days to merge
- Author, contributions since 2014 (1,372)
- Reviewer (Apache Beam PMC), contributions since 2015 (2,270), reviews since 2016
- Reviewer, contributions since 2017 (42), reviews since 2017
- PR opened on Friday, reviewed (inducing refactoring) on Friday

kafka 5784: 2018 | commit merge | initial refactoring edits | no bot | corrective changes (fixing faults) | **refactoring inspired by code review** | review comment questioning about a *Rename Attribute* refactoring, which inspired a *Rename Method* | self-affirmed minor review comment | 1 refactoring edit

- 1º subsequent commit: *1 Rename Method*
- 1 subsequent commit, 2 file changes, 6 added lines, 6 deleted lines, 3 days to merge
- Author (Apache Kafka Committer), contributions since 2015 (1,123)
- Reviewer (Apache Kafka PMC), contributions since 2007 (14,688), reviews since 2016
- PR opened on Thursday, reviewed (inducing refactoring) on Sunday

servicecomb-java-chassis 346: 2017 | commit merge | no bot | adaptive changes (adding a new feature) | **refactoring induced by code review** | direct review comment regarding code logic (suggesting a method extraction – "maybe...") and documentation | 1 refactoring edit

- 1º subsequent commit: *1 Extract Method*
- 1 subsequent commit, 2 file changes, 109 added lines, 23 deleted lines, 1 day to merge
- Author, contributions since 2016 (39)
- Reviewer (Apache Servicecomb-Java-Chassis PMC), contributions since 2016 (438), reviews since 2017
- Reviewer (Apache Servicecomb-Java-Chassis PMC), contributions since 2017 (234), reviews since 2017
- PR opened on Sunday, reviewed (inducing refactoring) on Monday

tomee 275: 2018 | commit merge | 1 external reference after PR merge | no bot | corrective (fixing a fault) and perfective (update) changes | none PR description | **refactoring induced by code review** | direct review comment questioning code logic, providing a reason (suggesting the use of boolean instead of Boolean) | 1 refactoring edit

- 1º subsequent commit: *1 Change Variable Type*
- 1 subsequent commit, 1 file change, 1 added line, 1 deleted line, 1 day to merge
- Author, contributions since 2012 (318)
- Reviewer, contributions since 2015 (98), reviews since 2018
- PR opened on Friday, reviewed (inducing refactoring) on Friday

kafka 5423: 2018 | commit merge | 3 mentions, 7 external references after PR merge | no bot, PR presents a checklist for contribution | perfective changes (update) | self-affirmed minor PR | none PR description | **refactoring led by the author** | review comment questioning to add an unit test | 1 refactoring edit

- 1º subsequent commit: *1 Change Variable Type*
- 1 subsequent commit, 2 file changes, 200 added lines, 192 deleted lines, 0 days to merge
- Author (Apache Kafka, Flink, Storm Committer and PMC), contributions since 2014 (1,387)
- Reviewer (Apache Kafka Committer and PMC), contributions since 2015 (2,461), reviews since 2016
- PR opened on Wednesday, reviewed (inducing refactoring) on Wednesday

beam 4458: 2018 | commit merge | no bot, PR presents a fulfilled checklist for contribution | adaptive changes (adding a new feature) | none PR description | **refactoring induced by code review** | direct review comment regarding code logic (suggesting an attribute extraction – "maybe...") and another review comment concerning a test | 1 refactoring edit

- 1º subsequent commit: *1 Extract Attribute*
- 1 subsequent commit, 2 file changes, 4 added lines, 47 deleted lines, 1 day to merge
- Author, contributions since 2009 (28,964)
- Reviewer (Apache Beam, Avro committer and PMC), contributions since 2010 (1,665), reviews since 2016
- PR opened on Sunday, reviewed (inducing refactoring) on Monday

Non-refactoring-inducing PRs:

dubbo 3317: 2019 | commit merge | initial refactoring edit | 3 mentions, 1 external reference after PR merge | no bot, PR presents a fulfilled checklist for contribution | corrective changes (fixing faults) | review comments concerning an error message and deleted lines of code

- 1 subsequent commit, 1 file changes, 2 added lines, 2 deleted lines, 5 days to merge
- Author, contributions since 2016 (1,107)

- Contributor, contributions since 2016 (828), reviews since 2017
- Reviewer (Apache Dubbo PMC), contributions since 2011 (809), reviews since 2019
- PR opened on Wednesday, reviewed on Wednesday

kafka 5219: 2018 | commit merge | 1 mention, 7 external referecences after PR merge | no bot, PR presents a checklist for contribution | corrective changes (fixing faults) | None PR description | This PR is a follow up to Kafka PR #4782 (closed status) | review comment questioning code logic and providing a reason for a code simplification

- 1 subsequent commit, 1 file change, 1 added line, 3 deleted lines, 0 days to merge
- Author (Apache Kafka, Flink, Storm Committer and PMC), contributions since 2014 (1,317)
- Reviewer (Apache Kafka Committer and PMC), contributions since 2015 (2,378) , reviews since 2016
- Reviewer (Apache Kafka Committer and PMC), contributions since 2015 (967) , reviews since 2016
- PR opened on Wednesday, reviewed on Wednesday

beam 5785: 2018 | commit merge | no bot, PR presents a checklist for contribution | adaptive changes (adding a new feature) | None PR description | review comments about code style (no effect – PR's autor won the discussion) and requesting additional features (no effect in one of them – PR's author won the discussion, by providing reasons) | self-affirmed minor review comment | no Java files

- 1 subsequent commit, 1 file change, 2 added lines, 2 deleted lines, 2 days to merge
- Author, contributions since 2016 (247)
- Reviewer (Apache Beam, Avro committer and PMC), contributions since 2009 (2,037), reviews since 2016
- PR opened on Wednesday, reviewed on Thursday, reviewed on Friday

tinkerpop 524: 2017 | commit merge | initial refactoring edits | no bot | corrective (fixing faults) and perfective (optimization) changes | a detailed PR description | Review comments concerning code logic (iterator value in a loop and an argument value) that led to a discussion (PR's author answered both comments, winning the discussion, by providing reasons)

- 1 subsequent commit, 6 file changes, 47 added lines, 5 deleted lines, 3 days to merge
- PR's author, contributions since 2009 (11,965)
- Reviewer, contributions since 2011 (2,950), reviews since 2016
- Reviewer, contributions since 2013 (92), reviews since 2016
- PR opened on Thursday, reviewed on Friday

kafka 6818: 2019 | commit merge | 4 external references after PR merge | no bot, PR presents a checklist for contribution | corrective changes (fixing faults) | review comments regarding code logic and need for additional tests, providing reasons for alterations

- 1 subsequent commit, 2 file changes, 65 added lines, 50 deleted lines, 4 days to merge
- Author, contributions since 2015 (76)

- Reviewer (Apache Kafka PMC), contributions since 2010 (1,639), reviews since 2016
- Reviewer (Apache Kafka committer and PMC), contributions since 2015 (3,547), reviews since 2016
- PR opened on Sunday, reviewed on Tuesday

beam 6050: 2018 | commit merge | initial refactoring edits | 1 mention, 3 external references after PR merge | no bot, PR presents a checklist for contribution | adaptive changes (adding new feature) | review comments questioning simple code logic (use of a specific method, by providing reason) and complex code logic (answered by PR's author, by providing clarifications)

- 1 subsequent commit, 4 file changes, 23 added lines, 10 deleted lines, 1 day to merge
- PR's author, contributions since 2014 (2,074)
- Contributor, contributions since 2010 (3,467), reviews since 2016
- Reviewer (Apache Beam PMC), contributions since 2017 (96), reviews since 2017
- PR opened on Tuesday, reviewed on Thursday

kafka 7132: 2019 | commit merge | 4 external references after PR merge | no bot | corrective changes (fixing faults) | a review comment questioning code logic (null value of a variable), answered by the PR's author and a review comment questioning a method call ("Should this be..."?) inspired a refactoring edit | **refactoring induced by code review** | 1 refactoring edit

- 1º subsequent commit: *1 Rename Method*
- 1 subsequent commit, 4 file changes, 23 added lines, 10 deleted lines, 1 day to merge
- Author, contributions since 2018 (145)
- Reviewer (Apache Kafka, Flink, Storm committer and PMC), contributions since 2014 (2,280), reviews since 2016
- Reviewer (Apache Kafka committer and PMC), contributions since 2010 (1,770), reviews since 2016
- PR opened on Monday, reviewed on Monday, reviewed (inducing refactoring) on Tuesday

beam 7696: 2019 | commit merge | 1 external reference after PR merge | no bot | adaptive changes (adding a new feature) | None PR description | direct review comment on a typo | none review comment in Java code

- 1 subsequent commit, 1 file change, 1 added line, 1 deleted line, 0 days to merge
- Author (Apache Beam committer and PMC), contributions since 2012 (1,385)
- Reviewer (Apache Beam PMC), contributions since 2012 (346), reviews since 2017
- PR opened on Thursday, reviewed on Thursday

dubbo 3185: 2019 | commit merge | 3 mentions after PR merge | no bot, PR presents a checklist for contribution | perfective changes (pre-checking) | a direct review comment suggesting a simplification of code induced a refactoring edit | **refactoring induced by code review** | self-affirmed minor review comment | 1 refactoring edit

- 1º subsequent commit: 1 *Inline Method*
- 1 subsequent commit, 2 file changes, 5 added lines, 9 deleted lines, 0 days to merge

- Author, contributions since 2016 (1,107)
- Reviewer, contributions since 2017 (85), reviews since 2018
- PR opened on Thursday, reviewed on Thursday

flink 9451: 2019 | commit merge | flinkbot | corrective changes (fixing faults) | direct review comment on a typo | self-affirmed minor review comment | none review comment in Java code

- 1 subsequent commit, 1 file change, 1 added line, 1 deleted line, 6 days to merge
- PR's author (Apache Flink committer and PMC), contributions since 2013 (6,278)
- Reviewer (Apache Flink, Hbase PMC), contributions since 2013 (185), reviews since 2019
- PR opened on Thursday, reviewed on Thursday

References

Swanson, E. B. "The Dimensions of Maintenance". In *Proceedings of the 2nd International Conference on Software Engineering*, 492–497. Washington, USA: IEEE Computer Society, 1976.

Mockus, A. and Votta, L. G. "Identifying Reasons for Software Changes Using Historic Databases". In *Proceedings of the International Conference on Software Maintenance.* Washington, USA: IEEE Computer Society, 2000.