FSE/CACM Rebuttal^2

Correcting *A Large-Scale Study of Programming Languages and Code Quality in GitHub*

Emery D. Berger
University of Massachusetts Amherst

Petr Maj
Czech Technical University in Prague

Olga Vitek
Northeastern University

Jan Vitek
Northeastern University
Czech Technical University in Prague

November 27, 2019

Abstract

Ray, Devanbu and Filkov ("the FSE authors") issued a rebuttal [1] of our TOPLAS paper *On the Impact of Programming Languages on Code Quality: A Reproduction Study* ("TOPLAS") [2]. Our paper reproduced *A Large-Scale Study of Programming Languages and Code Quality in GitHub* [3], which appeared at FSE 2014 ("FSE") and was subsequently republished as a CACM research highlight in 2017 [4] ("CACM"). This article is a rebuttal to that rebuttal.

Executive Summary

The crux of our TOPLAS paper's contribution is a reanalysis of the key research question (RQ1) of the FSE paper: "*Are some languages more defect prone than others?*". The FSE paper reported that ***six*** languages had a positive association with bugs (C, C++, Objective-C, PHP, Python, JavaScript) and ***five*** had a negative association (Typescript, Clojure, Haskell, Scala, Ruby). To readers, this result appeared meaningful as it lent support to the view that typed functional languages are better than untyped imperative ones.

Our reanalysis is based on the same data as the FSE paper. The FSE authors claim in their response that our results confirm their conclusions; *we emphatically disagree*. After data cleaning and statistical improvements, our analysis shows only ***one*** language with a positive association with errors (C++) and ***three*** with a negative association (Clojure, Haskell, and Ruby). This result has no clear interpretation as Clojure is untyped, Haskell is typed, and Ruby is imperative. This is *not* a confirmation of the FSE results, or the CACM results for that matter, as we show that they are substantially the same.

Our TOPLAS paper uncovered additional issues with the FSE paper. However, correcting for these would require a full reproduction from first principles. We did not choose to embark on that journey because *we fundamentally do not believe that the question the FSE authors set out to answer, namely whether some languages are more defect prone than others, can be meaningfully answered by scraping GitHub.*

The FSE paper mixes correlation and causation in a way that makes it easy to misread its conclusions. An analysis of GitHub is necessarily measuring correlation and not establishing causation, as it does not constitute an experiment. In some parts of the text, the FSE authors are in fact careful to qualify their results as having a "small effect" and being “associations” (i.e., correlations). However, it is clear that their work is only of interest if it becomes actionable. Without a causal link between languages and bugs, these results are no more useful than observing that it often rains on Tuesdays.

Unfortunately, in numerous places in both the FSE and CACM papers and in the FSE authors’ own follow up work, a more causal formulation of their conclusions is presented (e.g., “static typing is better than dynamic”). The citation analysis reported in our TOPLAS paper shows that most people interested in their conclusions read the CACM and FSE papers as taking a stand on what is a better computer language. Out of all citations we have reviewed, only 4 were couched in terms of associations while 26 (incorrectly) implied causality. As a community, we must aim to minimize the chances that our research will be misinterpreted. We believe this is a goal worth fighting for.

A word about statistics. In the TOPLAS paper, we discuss the treatment of p-values and how to correct for the testing of multiple hypotheses. That discussion should not obscure the fact that, in the end, p-values are a tool ill-suited to large scale data analysis, as is the case for the analysis of GitHub repositories. Second, and more importantly, in the words of the American Statistical Association: *“Scientific conclusions should not be based only on whether a p-value passes a specific threshold.”* Our TOPLAS paper shows how to use prediction intervals to estimate the practical significance of observed effects (i.e., bugs). In Fig. 6, we take the language with the strongest association with bugs after reanalysis, namely C++, and the one with the smallest association, Clojure. Fig. 6 shows that these are mostly indistinguishable in practice. Since the difference is so small for languages at the two extremes, there is little point in discussing the others.

We note the presence of numerous demonstrably incorrect statements in the response, which we address in Sections 2.1, 2.2, 2.7, 2.9, and 2.10.

The remainder of this document provides an in-depth response to the rebuttal.

1 Overview

We briefly review and classify the 20 issues found in TOPLAS, marking them as either *accepted*, *unchallenged* or *challenged* by the FSE authors. The impact of 8 issues could be captured in our reanalysis as we corrected for them; they are marked with a $*$. We link to a version of TOPLAS with line numbers [PDF]; references are Section:Line. “Data” refers to the data set provided by the FSE authors. Cites refer to the citation list of TOPLAS.

Issues:

1. Data contains 148 unused projects. (S3.2:L268) Impact of unused data, projects for which data is available but where not analyzed, is unclear. We did not correct for it. ***Unchallenged***.

2. Committer vs Authors. (S3.2:L271) The FSE authors confused “git committers” with “git authors” , resulting in an under-estimate of developers on a project (29K reported v 47K actual). Developers is a control variable. We did not correct for this. ***Unchallenged***.

3. Code size. (S3.2:L280) The data has 80 million lines of code, a 17 mSLOC difference from the reported number. LOCs are a control variable. We did not correct for this. ***Unchallenged.***

4. $*$ Log v Log10. (S3.2:L297) FSE used a combination of log and log10 transformations of control variables for NBR. We corrected for this in reanalysis. ***Accepted***. FSE authors fixed it in CACM.

5. Language classification. (S3.2:L304) The classification of languages is incorrect as it does not capture a meaningful difference between languages. We reclassified in repetition and observe changes to the results. ***Challenged.*** The FSE authors argue that their classification is one of many possible. Sec. 2.10 explains why their classification is not meaningful.

6. $\underline{\ast}$ Bug categories. (S3.2:L333) The data regarding bug categories was inconsistent, and did not match what is in the paper. We were not able to correct for this. *This made it impossible to repeat or reanalyse RQ4.* ***Unchallenged***.

7. Missing commits. (S4.1:L364) The data has 106K missing commits (19% of the dataset). For Perl, 80% of commits are missing. We did not correct for this as it would require labelling commits as buggy and the automated labelling code was not shared with us. ***Accepted***. The FSE authors question the importance of missing commits. (See 2.6)

8. $\underline{\ast}$ Duplicated commits. (S4.1:L375) 27,450 commits (1.86% of the data) duplicated. *We corrected by removing duplicates.* ***Accepted***. The FSE authors question the impact of duplicates. Sec. 2.5 shows that duplicates can amount to 15% of some languages, thus controlling for them is important.

9. $\underline{\ast}$ TypeScript. (S4.1:L384) The language was misidentified. Out of 41 projects, only 16 contained TypeScript code. The other were translation files. 34% of the remaining TypeScript commits are to type declarations only. *We corrected by removing TypeScript from the dataset.* ***Accepted***. The FSE authors fixed this partially in CACM. They did not challenge the issue of type declarations. (See 2.7)

10. C++ and C. (S4.1:L397) Code in files ending with .h (C/C++) and in .C, .cc, .CPP, .c++, and .cxx were all ignored. This may lead to a systematic bias; for example, macro definitions in C are in .h files, so are templates and classes in C++. Unlike TypeScript, this cannot be explained by an error in GitHub Linguist. We did not correct for this. ***Unchallenged***. (See 2.8)

11. $\underline{\ast}$ Labelling accuracy. (S4.1:L421) The automated labelling of bug fixing commits is imprecise. We estimated a 36% false positive rate. *We have corrected for this using bootstrap.* ***Challenged.*** The FSE authors disagree with the false positive rate and object to the Bonferroni adjustment within the bootstrap. Sec 2.14 gives a statistical response and Sec. 2.16 backs up the FP rate.

12. $\underline{\ast}$ Zero sum contrasts. (S4.1:L471) FSE coded programming languages with weighted contrasts. Such contrasts are sensitive and compromise interpretability. A more common choice is zero-sum contrasts [17]. *We corrected for this and applied zero-sum contrasts.* ***Challenged***. The FSE authors did not see a justification for zero-sum contrasts. Sec. 2.15 gives a pointer to the proper statistical literature.

13. $\underline{\ast}$ Multiplicity of hypothesis testing. (S4.2:L482) Comparing 16 independent p-values to a significance cutoff of implies the family-wise error rate. *We corrected with Bonferroni and FDR.* ***Accepted.*** The FSE authors agreed on the need to correct, but object to the use of Bonferroni. Sec 2.13 explains that both Bonferroni and FDR results are provided and that they differ only by one language.

14. $\underline{\ast}$ Statistical v practical significance. (S4.2:L500) FSE focused on statistical significance of regression coefficients. P-values are largely driven by the number of observations in the data [11]. Small p-values do not necessarily imply practically important associations [4, 30]. *We corrected by evaluating practical significance with model-based prediction intervals.* ***Unchallenged.***

15. $\underline{\ast}$ Accounting for uncertainty. (S4.2:L511) The false positive rate in labelling introduces uncertainty not accounted for in the regression model. *We assessed accuracy with the help of independent developers. To correct the analysis we used the bootstrap method to assess the impact of that uncertainty on the results.* ***Accepted.*** The FSE authors agreed with the use of the bootstrap, but object to Bonferroni. Sec. 2.14 explains why Bonferroni is appropriate.

16. Tests. (S5.1:L562) About 16% of the data are tests, the nature of bugs in tests is unclear: When is a test buggy? We did not correct for this as the correction is unclear. ***Accepted.*** The FSE authors claim that errors in tests should be measured. Sec 2.8 explains why tests should be treated with care to avoid blaming tests for bugs in the application.

17. Project selection bias. (S5.3:L585) The use of GitHub stars led to over representing "popular" areas such as the 17 variants of bitcoin, and the presence of projects that contain design patterns but no executable code. We did not correct for this. ***Unchallenged.***

18. Uncontrolled influences. (S5.6:L614) We report on a number of potential biases that were not controlled for such as the influence of project age on bug rate or developers active in multiple projects. We did not correct for this. ***Challenged.*** The FSE authors argue that it is not possible to control for all sources of possible bias. (See 2.4) We agree that *all* sources of bias cannot be captured, but believe that some effort should be made to capture the ones that can.

19. Project provenance. (S5.4:L597) FSE is limited to open source projects hosted on GitHub. Commercial software may have different error characteristics. Thus, any conclusion of the work is limited to that context. We did not correct for this. ***Unchallenged.***

20. Relevance to the RQs. (S5.7:L630) We observe that many of the bug fixing commits are not affected by programming languages, e.g., setting the wrong TCP/IP port is not a bug sensitive to the choice of language. Only a portion of the bugs are relevant. For two projects we looked at, only 5% of the bugs were "language related" . We did not correct for this. ***Unchallenged.***

To summarize: we reported 20 issues. Out of those, *we corrected for 8* in reanalysis. The FSE authors challenge 4 issues, including two we corrected for, namely 11 (labelling accuracy) and 12 (0 sum contrasts), and take issue with the use of Bonferroni adjustments.

2 Response

This section is structured chronologically around quotes from the rebuttal. We answer all points.

2.1 Choice of reproducing FSE rather than CACM

The FSE authors say it would have been proper to focus on the 2017 CACM archival version rather than the older FSE 2014 conference paper.

> *"It is standard practice in Computer Science to have conference paper abstracts extended/improved and published in an archival form in a journal. Once the journal version is published, other papers begin to cite it and stop citing the conference version. Berger et al. refer to results from both our preliminary, FSE paper and our final CACM paper in their TOPLAS paper comparisons, which creates confusion." – Rebuttal*

We requested the CACM data on November 13th 2017, a second request was issued a week later, and some data was provided on December 6th 2017:

```
Please use the following Dropbox link to download the data and scripts we used
for the project.
https://www.dropbox.com/sh/pfjkg2oztsohsls/AABzCIUCx1TyJqHYa0vf4MfEa?dl=0 I think
the scripts are quite self-explanatory and will help you to start the project.
We are in the process of releasing the data and scripts with detailed README files.
I will share the link once we finish that, hopefully by end of the year.  Please
let me know if you have any questions.  Thanks Baishakhi
```

No further data was shared. As of this writing, as far as we are aware, no dataset or scripts have been released to the public. The FSE authors write:

> *“CACM uses the corrected TypeScript data, and Berger et al. were aware of that, yet they chose to compare to the version of our FSE paper that has a mistake in it. This seems unnecessarily tendentious.” – Rebuttal*

> *“TOPLAS authors were aware that our CACM was the definitive version, yet chose to compare to FSE” – Rebuttal*

Nowhere in the email exchange with the FSE authors is there mention that we were given FSE data. We discovered it later, when we found numbers that did not match CACM.

To sum up, we “chose” to reproduce the paper we had the data for. The FSE authors further claim:

> *“We believe the apparent small differences are attributable to changes our artifacts were going through as we were transitioning the FSE artifact to the CACM artifact (close in time to when we shared the FSE artifact with Dr. J. Vitek and his student in 2017).” – Rebuttal*

We requested the CACM data after that paper had been published, months after the FSE authors sent their camera ready version to the ACM. One would expect the CACM data to be available.

2.2 Improvements between FSE and CACM

The FSE authors argue that CACM vastly improved over FSE, and thus any comparison to FSE is moot.

> *“That conference paper reported on preliminary results. Upon it being invited as a research highlight, after being recommended by the FSE conference program committee chairs, it underwent multiple review rounds at CACM. The CACM version is much improved over the FSE version and completely supersedes it.” – Rebuttal*

Reading CACM does not reveal any changes in methodology or approach. In fact, the paper does not describe any changes to the research approach, methodology or result with respect to FSE.

To evaluate the putative improvements in CACM over FSE, we take every numeric constant and every table appearing in CACM and compare those constants and tables to their corresponding values in FSE. The expectation is that a “much improved” paper would have different numbers, and that those differences would have a qualitative impact on the scientific conclusions. We show the differences below in yellow.

CACM	728 projects, 63 million SLOC, 29,000 authors, 1.5 million commits, in 17
FSE	729 projects, 80 Million SLOC, 29,000 authors, 1.5 million commits, in 17
CACM	...top 19....top 50 projects... we analyze 850 projects spanning 17 different
FSE	...top 19....top 50 projects... we analyze 850 projects spanning 17 different
CACM	The archive logs 18 different GitHub events
FSE	The archive logs eighteen different GitHub events
CACM	...top **3** projects in C...we select the top 50 projects ...we drop the projects
FSE	We select the top 50 projects...we filter out the
CACM	with fewer than 28 commits (28 is the first...
FSE	having less than 28 commits, where 28 is the first...
CACM	This leaves us with **728** projects. Table 1 shows the top 3 projects in each
FSE	This leaves us with 729 projects. Table 1 shows the top three projects in each
CACM	For each of **728** projects, we downloaded...
FSE	For each of these 729 projects, we downloaded
CACM	... fewer than 20 commits in that language, where 20 is the first quartile...
FSE	... fewer than 20 commits in that language, where 20 is the first quartile
CACM	For example, we find 220 projects that use more than 20 commits in C.
FSE	For example, we find 220 projects that use more than 20 commits in C.
CACM	728 projects ... 17 languages with 18 years ... 29,000 developers, **1.57** million
FSE	729 projects ... 17 languages with 18 years ... 29 thousand developers, 1.58
CACM	and **564,625** bug fix
FSE	and 566,000 bug fix
CACM	We detect 30 distinct domains, ...
FSE	We detect distinct 30 domains (i.e. topics)...
CACM	annotated 180 randomly chosen bug fixes, ...

FSE annotated 180 randomly chosen bug fixes, ...

CACM 70% . . . to a high of 100% ...an average of 84% . ranged from 69% to 91% ...average of 84%
FSE [table shows: 100% ... 70% (average 84%), recall from 69% to 91% , average 84%]

CACM Our technique could not classify 1.04% of the bug fix
FSE Our technique could not classify 1.04% of the bug fix

CACM Analysis of deviance reveals that language accounts for less than 1%
FSE which accounts for less than one percent of the total deviance, is language.

CACM one additional defective commit since **e0.18× 4=4.79** . ...
FSE one additional buggy commit since e0.23× 4=5.03. ...

CACM we should expect about one fewer defective commit as **e− 0.26× 4=3.08**
FSE we should expect about one fewer defective commit as e− 0.23 × 4 = 3.18.

CACM with 17 languages across 7 domains,...
FSE with 17 languages across 7 domains,...

CACM Of 119 cells, 46, that is, 39% , are below the value of 5 which is too high.
FSE out of 119 cells in our data set, 46, i.e. 39% , are below the value of 5.

CACM No more than 20% of the counts should be below 5.
FSE that no more than 20% of the counts should be below 5.

CACM the low strength of association of 0.191 as measured by Cramer's V
FSE the low strength of association of 0.191 as measured by Cramer's V,

CACM significantly different with p = 0.00044.
FSE significantly different with p = 0.00044.

CACM language class with p = 0.034.
FSE language class with p = 0.034.

CACM language class explaining much less than **1** % of the deviance.
CACM Chi-square...of 99.05 and df=30 with p=2.622e− 09 ... a value of 0.133,
FSE Chi-square...of 99.0494 and df=30 with p=2.622e− 09 ... a value of 0.133,

CACM ...that have defect density below 10 and above 90
FSE ...that have defect density below 10 percentile and above 90

CACM p-values are significant (<0.01).
FSE p-values are significant (< 0.01)

CACM Generic programming errors account for around 88.53%
FSE Generic programming errors account for around 88.53%

CACM Memory errors account for 5.44%
FSE Memory errors account for 5.44%

CACM 28.89% of all the memory errors in Java
FSE 28.89% of all the memory errors in Java

CACM 1.99% of the total bug fix commits
FSE 1.99% of total bug fix commits

CACM C and C++ introduce 19.15% and 7.89% of the
FSE C and C++ introduce 19.15% and 7.89% of the

CACM example, 92% in Go.
FSE to 92% is Go.

CACM Around 7.33% of all the
FSE Around 7.33% of all the

The CACM numbers are *extremely* close to those appearing in FSE. The differences are the removal of one project (729→ 728), 17m fewer LOCs (80→ 63), .01m fewer commits (1.59→ 1.58), and 1375 fewer bug fixes (566'000→ 564'625). Most are explained by removal of TypeScript data. The reduction in LOCs is a mystery. We emphasize that CACM does not list these changes, nor does it provide a justification. Next, we turn our attention to the tables.

Table 1: Top 3 projects in each language
As expected, the only differences are for TypeScript:
CACM: Typescript-node-definitions, StateTree, typescript.api
FSE: Bitcoin, litecoin, qBittorrent

Table 2. Study subjects
As expected, the only differences are in the TypeScript:
CACM 14 projects, 0.9k bug fixes; Total bug fixes 564k ; 728 projects
FSE: 96 projects, 2.4K bug fixes; Total bug fixes 566K; 729 projects

Table 3. Different types of language classes.

The only difference is the name of the categories. The grouping of languages has not changed.

Table 4. Characteristics of domains.
The tables are identical across the CACM and FSE. Adding the projects in the "total projects" column yields 729 projects whereas CACM has only 728. Perhaps the table was not updated when moving from FSE to CACM.

Table 5. Categories of bugs and their distribution in the whole dataset
The tables are identical. The "Count" column counts bug fixing commits. Removing TypeScript decreased bugs by 1375, yet that difference does not show up. Perhaps the CACM table was not updated. Stranger, the total number of bugs is 583,924, higher than FSE.

Table 6. Some languages induce fewer defects than other languages.
As expected, the regression results are different; this is explained by the removal of TypeScript.

Table 7. Functional languages have a smaller relationship to defects than other...
The tables are identical. This is explained by the fact that TypeScript was not included in both.

Table 8. While the impact of language on defects varies across defect category,...
The tables are different, as expected since TypeScript data changed.

To conclude, the difference between CACM and FSE can be attributed in its *entirety* to the removal of 1375 TypeScript bug-fixing commits. Since we correct for TypeScript, our TOPLAS paper subsumes the changes made by the FSE authors in CACM.

2.3 Project size

The FSE authors paraphrase a talk we gave:

> *"The size of projects are too different (some have millions of lines of code, others just tens of lines), yet in CACM the authors don't normalize for the number of commits. Hence the results may be wrong!" – Rebuttal*

Project size and commits are part of the FSE model. This is not in dispute.

In the talk, we made the following argument: In the data, the 25 Perl projects have an average of 194 commits (4,863 in total) whereas the 82 C projects have an average 5,451 commits (447,043). The regression with "project size" as predictor assumes that the log of the expected number of bugs increases linearly with size, and with the same slope for every language. This assumption is difficult to verify when projects in different languages differ in size substantially. When the assumption does not hold, the adjustment may be insufficient, and the difference in the number of bugs between languages may as well be attributed to a non-linear scaling in complexity of larger-size projects.

This can easily be avoided by changing the way projects are selected. The projects should be chosen by controlling their characteristics rather than relying on GitHub "stars" which capture popularity and are unrelated to software development.

2.4 Uncontrolled effects

The FSE authors take issue with uncontrolled effects:

> *"There seem to be uncontrolled effects, hence the results may be wrong!" – Rebuttal*

This refers to issue 18 (S5.6:L614) where we point out potential biases that were not controlled, such as a prolific developer working on multiple projects in one language. Such a developer can influence the analysis (e.g. positively, if she writes non-buggy code). This, and other possible sources of bias, were not corrected for in TOPLAS as it would require work outside the scope of a reproduction. We argue it is worth investigating.

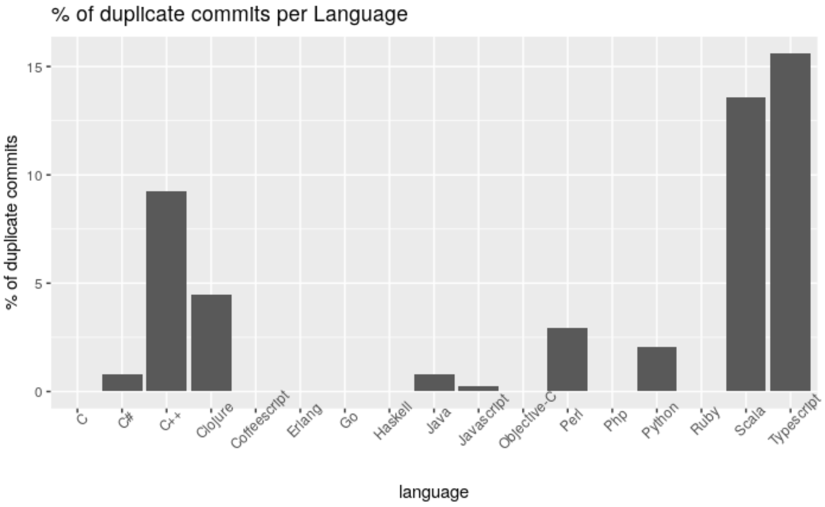
2.5 Controlling duplicates

The FSE authors dismiss the importance of duplicates, they suggests it is unlikely that they can matter:

> *"We did not look for duplicated commits but are not surprised that such commits may exist: notably they amount to less than 2% overall! TOPLAS doesn't provide any evidence that the duplicated commits are anyhow biased, i.e., either more or less buggy than the 98+% non-duplicated ones. Thus, it is very unlikely that this affects our results." – Rebuttal*

What the FSE authors miss is that their data is skewed with a few languages constituting the majority of the commits. Furthermore, they assume duplicates are uniformly distributed, i.e., that each language is affected the same way.

However, duplicate commits are decidedly not uniformly distributed. The attached graph shows duplicates by language: they account for 9% of C++ and close to 15% of Scala and TypeScript.



We corrected for this by removing all duplicates. For reference, read: [M. Allamanis. The Adverse Effects of Code Duplication in Machine Learning Models of Code. https://arxiv.org/abs/1812.06469]

2.6 Missing data

The FSE authors mischaracterize issue 7 (S4.1:L364) where we found 106K missing commits:

> *"There is missing data, about 20% over all projects, and up to 80% for some projects. Hence, the results may be wrong!" – Rebuttal*

We make no such claim. We simply assert that missing data is something to worry about, especially when languages like Perl or C++ lose so many of their commits. We did not correct for this, and thus this issue did not affect our conclusions. We suggest that this should be corrected in future versions of the FSE authors' work.

The FSE authors also remark that our figure is "highly disingenuous":

> *"Figure 3 are highly disingenuous. Fig 3 shows all the bar plots on the same chart, and truncates (shortens) the 80% bar (for Perl), thus most bars look comparable to it, appearing as if for most languages we have lost most commits. In fact, looking at the figure's y-axis units, we see that there is only one language for which the discrepancy is at 80% , Perl. For all others the discrepancy is below 15% , and for most way below. This is classical data misrepresentation, usually only seen in tendentious*

> *political debates." – Rebuttal*

The figure is clearly labeled and the case of Perl is called out in the text.

2.7 TypeScript

Our correction for TypeScript was challenged by the FSE authors. This is issue 9 (S4.1:L384), out of 41 projects labeled as TypeScript, only 16 contained TypeScript code. The other were translation files (.ts). The FSE authors say (a) they fixed it and (b) we were aware of it:

> *"This is an unnecessary misrepresentation of our work. We were the first to correct our work immediately after noticing this very issue." – Rebuttal*

> *"CACM uses the corrected TypeScript data, and Berger et al. were aware of that, yet they chose to compare to the version of our FSE paper that has a mistake in it. This seems unnecessarily tendentious." – Rebuttal*

The paper in the ACM DL (https://dl.acm.org/citation.cfm?id=2635922) is what we reproduced and are commenting on. That version does not include any information about errors in TypeScript. Neither does CACM.

Part of issue 9 is left unchallenged: 34% of the remaining TypeScript commits are to type declarations. In TypeScript, some files do not contain code, they only have function signatures. These are the most popular and biggest projects in the dataset. We corrected for this by removing TypeScript.

2.8 V8 and C++

Our TOPLAS paper makes a point with respect to C++ and V8 which the FSE authors may have missed:

> *"The V8 example is interesting from a software engineering perspective. Yes, a significant amount of the JavaScript code in V8 is test code. However, coding (and fixing) test files is a big part of what developers do, and thus, there are good experimental reasons to include them" – Rebuttal*

While V8 is a C++ project, the dataset only contains 16 C++ commits, and 2,907 JavaScript commits. To account for this one should recover the commits that were lost and disambiguate whether a bug fixing commit to a JavaScript file means that the test (in JavaScript) is buggy or that there was a bug in C++ code and that the test is a reproduction. In the latter case, the FSE analysis blames the wrong language.

2.9 Correlation vs. Causation

A main points of contention is about causation. In TOPLAS, we argue that FSE and CACM were read as advocating a causal link. The FSE authors take umbrage:

> *"In FSE and CACM we were clear that we are talking of associations and correlations, not causality" – Rebuttal*

> *"Berger et al. object to the world's interpretation of our work, which is not something that should be addressed to us. They disingenuously attempted to pin this on us by erroneously quote mining" – Rebuttal*

We wrote: *"Correlation is not causality, but it is tempting to confuse them. The original study couched its results in terms of associations (i.e., correlations) rather than effects (i.e., causality) and carefully qualified effect size. Unfortunately, many of the paper's readers were not as careful. The work was taken by many as a statement on the impact of programming languages on defects . . . Out of the citations that discussed the results, 4 were careful to talk about associations (i.e., correlation), while 26 used language that indicated effects (i.e., causation)."* – TOPLAS (S1:L68)

We stand by the above and argue that, as scientists, it is our responsibility to write our results in a way that minimizes the likelihood of being misconstrued.

In this spirit, consider the following from CACM:

1. *"Most notably, it does appear that disallowing type confusion is modestly better than allowing it"*
2. *"among functional languages, static typing is also somewhat better than dynamic typing"*
3. *"We also find that functional languages are somewhat better than procedural languages"*
4. *"The data indicates that functional languages are better than procedural languages"*
5. *"it suggests that disallowing implicit type conversion is better than allowing it"*
6. *"that static typing is better than dynamic"*
7. *"that managed memory usage is better than unmanaged"*

These are causal arguments. They make for good copy, but are not supported. In other words, use of causal language by the FSE authors invited readers to draw causal inferences.

2.10 Language classes

The FSE authors disagreed with TOPLAS S3.2.2:L300 where, during repetition, we reclassify languages. Their argument is: (a) this belongs in reanalysis, and (b) we confirm FSE's conclusions.

> *"After the reclassification they got qualitatively the same results as ours: comparing tables 4c in TOPLAS and Table 7 in CACM, it is clear they both imply that the functional language categories are associated with (very slightly) fewer bugs, and that those findings are statistically significant." – Rebuttal*

> *"We note that this should have been called a reanalysis. Berger et al. called it repetition" – Rebuttal*

Addressing errors in data labelling does not fit neatly in the repetition/reanalysis framework. It is not reanalysis as we are not changing the analysis or cleaning the data, but it is not quite repetition either.

The classification of languages is wrong: consider Scala. In FSE, it is lumped with Clojure, Erlang & Haskell under the "Functional Paradigm". For this to be meaningful, there must exist some shared attribute these languages have that makes programs written in them similar. Referential transparency and higher order functions could be that. But, while Scala has higher-order functions, it is imperative. So, it is not a perfect match. Worse: Java also has higher-order functions, yet it isn't in that group.

As to the claim that we confirm the *conclusions* of FSE: our reclassification simply says that with a better classification, you get slightly different results. It does not validate the FSE authors' conclusion because the FSE analysis suffers from the same statistical problems as the rest of the paper. To validate their conclusions, we'd have to run with cleaned data and proper statistics. If curious, the FSE authors can adapt our scripts.

The FSE authors argued that it is not possible to compare models with different categories:

> *"Their model has different variables than ours, they derived two categories from one of ours. Thus, the models are not directly comparable, only the implications of those models are comparable." – Rebuttal*

We agree. The takeaway is that the FSE categories do not match any meaningful partition of languages according to features and that reclassification yields different but uncomparable results.

Finally, the FSE authors argue against the use of the FSE paper for comparison:

> *"We note that they compared their results to FSE, the superseded study. In the latter, CACM, we revised our classification of languages, e.g. with TypeScript." – Rebuttal*

This is a misstatement. The groupings of languages in categories are *exactly* the same in FSE and CACM; they just have different names. We repeat our findings from 2.2 above:

Table 3. Different types of language classes.
The only difference is the name of the categories. The grouping of languages has not changed.

Table 7. Functional languages have a smaller relationship to defects...
The tables are identical across CACM and FSE.

So, not only was there no meaningful change in the groupings, but the models are also *exactly* identical.

2.11 Repetition of RQ3 and RQ4

There is confusion about the status of repetition for RQ3 and RQ4. Did we repeat the FSE result? No.

> *"They reproduce our RQ3 results, and they acknowledge this in S.3.2.3. They implemented their own methods for RQ3, different than ours (in the data and scripts package we sent the TOPLAS authors, by mistake we had omitted the scripts to reproduce our RQ3 and RQ4; they did not follow up to ask us for them). ... From their results, they conclude the same as we do from ours in CACM: no evidence is found of a correlation between domain and defect proneness. Thus, this is a confirmatory reproduction study of ours. They did not perform a repetition of our RQ4 as they did not have our scripts, see previous paragraph." – Rebuttal*

S3.2.3 was a best effort guess of what FSE did. It is neither a successful repetition, nor a reanalysis.

2.12 Approaching reanalysis

We focus on RQ1 as it is the most important of the questions, and one we could repeat to our satisfaction.

> *"This is where TOPLAS is most misleading, on multiple accounts. First, their reanalysis is only an RQ1 reanalysis. They did not do a reanalysis of our RQ2-RQ4. They gathered their own data, for the same projects we did. For various reasons they couldn't mine all the projects we did. They also could get more data for some projects than we did" – Rebuttal*

Our reanalysis was done entirely using the FSE data with cleaning applied as described above.

The confusion comes from a misreading of issue 7 (S4.1:L364) where, in order to find missing commits, we downloaded projects from GitHub to compare them with the FSE data. The downloaded data was not used for our corrections. We did not find all the projects because the data lacked owner names. We compensated by heuristically matching projects (see S4.1:L360).

2.13 Multi-hypothesis testing

While the FSE authors concede that adjustment for multiple hypothesis testing is required, they disagree with our use of Bonferroni to correct for the error.

> *"They correct p-values for multiple hypothesis testing, though whether to correct or not in such a way is a matter of debate, especially p-values of coefficients within the same regression model. Still,we recognize that some may argue that a balanced correction like the false discovery rate is appropriate." – Rebuttal*

> *"In spite of them showing FDR results in Table 6, as discussed above, for their conclusions they use the very conservative, and problematic Bonferoni correction." – Rebuttal*

> *"they defer to the Bonferroni correction, and not FDR, in their final analyses in TOPLAS. They end up with only 5 significant results after the Bonferroni correction." – Rebuttal*

As the FSE authors mention, Table 6, column (c) shows the results of FDR and Bonferroni. The difference between the two is that one language, namely Ruby, loses statistical significance under Bonferroni. As a side note, column (e) where we apply the bootstrap, makes Ruby significant again.

2.14 Controlling for uncertainty, and the Bootstrap

The FSE authors argue that our use of the Bonferroni correction in the bootstrap is unnecessarily harsh:

> *"While bootstrap is potentially useful vis-a-vis labelling in the presence of uncertainty, unfortunately, they used the Bonferroni correction with the bootstrap. As discussed above, that correction is too conservative, with an inappropriately deleterious effect on significant findings. We posit that much of the information in the data was lost in TOPLAS after the application of Bonferroni." – Rebuttal*

The bootstrap-based analysis used the Bonferroni correction for methodological simplicity. Briefly, any FDR adjustment requires the calculation of p-values, which in turn requires the calculation of the reference distribution under the null hypothesis. Yet, it is not possible to resample the existing data under the null. Therefore, procedures for bootstrap-based hypothesis testing make additional assumptions – such as the assumption that the sampling distributions of the parameters are pivotal (i.e., the null hypothesis shifts the mean of the sampling distribution, without changing the variance). We decided against adding assumptions.

Instead, we opted for reporting bootstrap-based confidence intervals for the parameters of the Negative Binomial regression, for which the confidence level can be easily adjusted with Bonferroni. There is no generally accepted FDR-based adjustments for the width of confidence intervals.

Labeling uncertainty is a major issue in the original work, one that cannot be swept under the rug or ignored.

2.15 Zero sum contrasts

The use of zero-sum contrast is questioned:

> *"Berger et al. also apply a different contrasting technique, zero-sum, than the one we used, which they claim may be more appropriate in this setting, though they give no evidence for it. In CACM we have justified the use of weighted contrasts and provided a reference. TOPLAS doesn't directly compare their contrasting technique to ours. Due to the lack of objective evidence either way, we are not swayed by their argument. At best this point is debatable, if not unnecessary." – Rebuttal*

We recommend Chapter 8.4 of the 2004 book by Kutner, Neter, Nachtsheim, and Li, "Applied Linear Statistical Models" as a good starting point on the benefits and drawbacks of alternative coding strategies of categorical variables.

2.16 Bug labelling false positives

The FSE authors take issue with our methodology for determining bug labelling accuracy. Bug labelling takes each commit and labels that commit either as a bug-fixing-commit or not. We reported labelling (S4.1:L421) has a 36% false positive rate. This was done with the help of independent developers. The FSE authors claim that a sampling of twelve commits revealed 10 true positives.

> *"When examining a subset (numbering 12) of the buggy commits which they claim are false positives, considering the commit logs, the actual changes, linked issue numbers (when available) and discussions (when available) we found that 1 was a false positive, but 10 were in fact true positives, with one other one being debatable. We are therefore skeptical of their claimed 36% FP?" – Rebuttal*

Our reanalysis has 197 commits deemed buggy in FSE. Three independent developers evaluated them, and found that 71 of those were in fact not buggy. In the appendix we provide a full list of those 71 commits, giving the original URL, part of the commit message, and our assessment. Unlike suggested by the comment above, we find only 6 cases out of 71 in which we disagree with the developers.

Among these allegedly buggy commits are *many obvious* non-bugs. Clearly, the accuracy of the automated process used in FSE/CACM is questionable.

There is a further misunderstanding:

> *"Ours was not shown wrong: we reported 84% precision in both CACM and FSE (See S.2.4, CACM). Our method is automatic, which has pluses and minuses, as discussed in our paper." – Rebuttal*

Unfortunately, this has little to do with the point at hand as it refers to bug *classification*, i.e. once a bug-fixing commit has been identified, to which class (such as Algorithm, Concurrency, ...) does it belong: "*To evaluate the accuracy of the bug classifier, we manually annotated 180 randomly chosen bug fixes, equally distributed across all of the categories*." [CACM S2.4] This answers a different question.

2.17 Small Perl project

One of our talks was inaccurate:

> *"Dr. J. Vitek, in his many talks, publicly mocks us for including a 16-line Perl project in a table of the "Largest Projects" in Github. Hilarity understandably ensues from the audience. That would indeed be a laughable error, if we had done that. Three points here: first, the table, as described in our paper, shows the most-starred projects. Second, at the time of study, the Perl file mysqltuner.pl had 784 lines of code. Finally, that particular highly-starred Perl project got filtered out of our analyzed subset, for having insufficient commit history." – Rebuttal*

The accurate statement is "one of the three top-starred Perl projects is a 784 line script that does not have enough commits to be considered for inclusion in the analysis" .

3. Conclusion

We reviewed the points made by the FSE authors in their rebuttal. None of their issues invalidates our approach. Thus, our conclusions stand:

- The FSE paper found six languages had a positive association with bugs (C, C++, Objective-C, PHP, Python, JavaScript) and four had a negative association (Typescript, Clojure, Haskell, Scala, Ruby).
- Our reanalysis, based on the same data after data cleaning and improvement to the statistical methodology, shows only one language with a positive association to errors (C++) and three with a negative association (Clojure, Haskell, and Ruby).

All of our data and code is publicly available.

We look forward to the FSE authors publishing their CACM artifact.

Appendix: Reviewing bug labelling false positives

Our reanalysis examined 197 commits that were deemed buggy in FSE. Three independent developers evaluated them, and found that 71 of those were in fact not buggy. We summarize those 71 commits here. For each, we give its URL, the commit message, and our explanation of the decision. As can be seen, we disagree with our developers on 6 out of 71 commits. Some commits have been deleted since we performed our analysis. We note that among these allegedly buggy commits are *many obvious* non-bugs. Clearly, the accuracy of the automated process used in FSE/CACM is questionable.

1. http://github.com/0x43/DesignPatternsPHP/commit/2ca016fb66b16ae1591baee64c7c5f1c1cf9986d
   "some fixes on comments"
   Changes to comments. Not a bug.

2. http://github.com/19hz/nsq/commit/49ecef58e0e8ba0a1875e7830e344c7482beeebb
   Commit deleted

3. http://github.com/19hz/nsq/commit/a79206e1f3f89047acb6d17d9bd80f10437bb56d
   Commit deleted

4. http://github.com/AlexMeliq/less.js/commit/1199ce41b094def9b90edb56afcf8761d2294521
   "Make rhino error support better"
   Not a bug, changed the way errors are reported.

5. http://github.com/Arcank/nimbus/commit/53ce240d76add2e2fe5417fdafafd4337ea3b0ec
   "Removed networking operations from Nimbus. AFNetworking is now the default. . . ”
   Not a bug, change to features.

6. http://github.com/Chenkaiang/XVim/commit/032a1abe48bd2b92645876f986a70724196c93f9
   "Fix color for static text field"
   Set the color of a field.

7. http://github.com/clojure/core.logic/commit/756771aa990c5b6284595036de231398ee4819ca
   "allow ‘fixc‘ to take rands. x can be a vector values in this case.”
   Functionality change. Not a bug.

8. http://github.com/clojure/core.logic/commit/979308570786c30291feb43b111306e5ffa8c0a1
   ” note on how we intend to fix walk-term for IPersistentMaps"
   Changes to comments. No code modified.

9. http://github.com/docpad/docpad/commit/697f4185757b4b5aecd054d9514504c4c32714f4
   "v6.42.3. Improvement. . . . Fixed DocPad version number undefined ..."
   Functionality change. Not a bug.

10. http://github.com/faylang/fay/commit/974fff51e0d0c98b667700d2c517af4d40a2adee
    "Add lazyness to infix operators (∗ + - etc) In the test case I’ve replaced. . . ”
    Functionality change. Not a bug.

11. http://github.com/GeertJohan/gorp/commit/071ca908b6e5049351dfbd6242d800e53463cce6
    "Add named query parameter binding from map or struct. Fixes go-gorp# 61"
    The issue is "[Feature Request] Named query arguments # 61". Functionality change. Not a bug.

12. http://github.com/GeertJohan/gorp/commit/af8337d4a1d0d35911c3cd1a8d4be58bf518fa03
    "Add sqlite3 and PostgreSQL Dialects. Fix issues in tests that let err. . . "
    The commit adds more functionality, and in tests, calls panic if an errno is reported. This is a bug.

13. http://github.com/lfe/lfe/commit/81b48ac826176b9899d0d34f672390d29bf657a9
    "Extend ? macro to handle optional timeout and default value."
    Functionality change. Not a bug.

14. http://github.com/lfe/lfe/commit/9ae33afc8aff8a8cdbc02ba1f4a7a4643c33854b
    "Better error checking in guards."
    Functionality change. Not a bug.

15. http://github.com/magicalpanda/MagicalRecord/commit/b11a797379b358448e499e5ae9529f953765d23f
    "Converted CoreDataRecipes sample to MagicalRecordRecipes sample appli. . . "
    Changes 75 files and 9K lines. Not a bug.
16. http://github.com/magicalpanda/MagicalRecord/commit/e048f91c71798d10c8709f19687faed01895d8ca
    "fix comments, add new context helper, made helpers more consistent"
    Not a bug. Comments and code additions.
17. http://github.com/mpeltonen/sbt-idea/commit/0395e37d2cde8732ca09b6f53e4313e88f3b3c22
    "Include classes jars with classifier in idea config (issue # 145)"
    The issue is "Test JAR dependency not included in "external libraries" in IntelliJ project # 145". Not a bug.
18. http://github.com/mpeltonen/sbt-idea/commit/18804a91672b28cb9bf7a38fa1fe28d91f4ba30a
    "Tabs to spaces formatting fixes"
    Not a bug – change spacings.
19. http://github.com/mpeltonen/sbt-idea/commit/831cd1666015ed07bd218995e8ab9b760739a380
    "Make 'no-sbt-classifiers' to be the default With lastest sbt version, . . . "
    Not a bug – at least not a programming issue.
20. http://github.com/mpeltonen/sbt-idea/commit/fc13c0ebcb184a5fde3ae0eb17fe97e91555c107
    "set "deprecation" and "unchecked" scalac settings according . . . fix for # 120"
    The issue is "Set deprecation and unchecked scalac options # 120" Functionality change. Not a bug.
21. http://github.com/MythTV/mythtv/commit/04828d18b995d7f033f60e6f98ff2f792ffcfe9c
    "Group items now have signals to indicate that they're about to go . . . "
    This adds more events to the UI and the scaffolding around it, changes formatting (removes extra { }), not a bug.
22. http://github.com/plumatic/plumbing/commit/d34194b3987d2d527e78369744cfa46a0ecaea96
    "better lazy test – make sure that lazy error checking happens"
    Change to a test file. Not a bug.
23. http://github.com/sinclairzx81/typescript.api/commit/52c7b57392633d4c01d10b99585230d2b048caad
    "added declarations to compiled unit object. additional updates on d.ts file. . . "
    No evidence of a bug.
24. http://github.com/0x43/DesignPatternsPHP/commit/0d35ab368bfb71f2a72c538e9a4d708d51b2f98d
    "fix a typo"
    The field isDirty is consistently misspelled: 1 def and 2 uses. The code lived like this for over a year. Then all uses were changed in single commit. Not a bug.
25. http://github.com/19hz/nsq/commit/043b79acda5fe57056b3cc21b2ef536d5615a2c2Deleted commit
26. http://github.com/19hz/nsq/commit/2a6ed0dd5a24e6f9f73b6742007f1b2cf99fb31b Deleted commit
27. http://github.com/Arcank/nimbus/commit/620d9580c9e401168996b17ea00e2dc77e43e245
    "Fix up vertical alignment of images in attributed strings. Now suppor. . . "
    Not a bug.
28. http://github.com/Chenkaiang/XVim/commit/0245026e393d0f95d8f8534a753f68c4ffe5d80f
    "Fix XVimProject# 189.∗ XVimWindow is no longer an NSView ..."
    The issue is: "Use objc_setAssociatedObject instead of making XVimWind..." Functionality change. Not a bug.
29. http://github.com/clojure/core.logic/commit/65bdedafa827a17eccc33f497d166810fbd162f2
    "Disequality that does not depend on unify. ... uses the :prefixc property. . . "
    Functionality change. Not a bug.
30. http://github.com/docpad/docpad/commit/1c3640ffb9b8a1727d41625a6a80588f27f680ff
    "Fixed unit tests :)"
    Changes 16 files in 586 lines.This adds tests and makes scaffolding around them robust. Not a bug.
31. http://github.com/faylang/fay/commit/712bfd41d426c767a5a413ecc24911e93bcf0b54
    "Fix console example (closes # 201):"
    The issue is "Example Console.hs should use putStrLn instead of print? # 201" changes how error messages are printed. Both print and printStrLn could have been reasonably used. Not a bug.
32. http://github.com/faylang/fay/commit/d527586ee7ac5646068ffb885198814e877d8fca
    "Make error messages a tiny bit friendlier, and some general house-keeping."
    Not a bug.

33. http://github.com/faylang/fay/commit/d7340a55f0c5ee903ae19bc3a733907127eb4d99
"Fix TODO"
Remove a "TODO" and some lines. Not a bug.

34. http://github.com/GeertJohan/gorp/commit/3aa59f87d2d4ba141c81f0d892fc041a25ef649d
"Add support for time.Time objects . . . Thi sresolves gorp issue go-gorp# 14"
The issue is "I'm not sure if this is something you want to support, but Sqlite (and presumably other databases) does not support a formal datetime type. . . . ." Functionality change. Not a bug.

35. http://github.com/K2InformaticsGmBH/proper/commit/57c3226147127fc860fa7dec9ec60ec98bb01a79
"PropEr now catches exits by default."
No indication of a bug.

36. http://github.com/lfe/lfe/commit/14180d6f839760ce071fb49d07f21dfaaa612795
"Better record handling, internal fixes, documentation."
A cumulative update, but the comment "% Ensure Return" suggests a genuine bug.

37. http://github.com/lfe/lfe/commit/e163397cdf515d653060758d8c8ae593e0ce9104
"A small fix in the docs and update copyrights."
Not a bug.

38. http://github.com/mpeltonen/sbt-idea/commit/1125a8b760810f682a3c929534e3d235a27e1245
"support for classes dirs in unmanagedClasspath, in addition to jars Issue 181"
The issue is "unmanagedClasspath in Runtime # 181" Functionality change. Not a bug.

39. http://github.com/plumatic/plumbing/commit/a0149aba501833dafe57cd35962bb6802fb87fda
"better error messages for graph"
Not a bug.

40. http://github.com/sinclairzx81/typescript.api/commit/654256c47a45764f5356c71b9fd00ae8ae8a897f
"... tried to solve the missing lambda, '_this' bug by way..."
This is a bug. Buried down in huge refactoring commit.

41. http://github.com/sinclairzx81/typescript.api/commit/8541d7ca595449570852428edbe647aa00f0d435
"fix on register()"
This looks like a bug!

42. http://github.com/0x43/DesignPatternsPHP/commit/55017fb43f936aa230be65447f5e69a38dda7f10
"fix PSR-0"
This is not a program but a set of design patterns. Not a bug.

43. http://github.com/0x43/DesignPatternsPHP/commit/ceb6a3eeb958576c5229a057be802cca976031ec
"fixed typo in Singleton"
Change to a comment. Not a bug.

44. http://github.com/AutoMapper/AutoMapper/commit/2a9361439752ceb0daa852aa86e9b8c078096147
"Tests passing for WinRT"
Does not look like a bug.

45. http://github.com/AutoMapper/AutoMapper/commit/6ff6c50bd27bb33076fb3196f4b409507a33a1d1
"Modified data reader mapper to support . . . Fixing compile error; closes # 254"
The issue is "Support for yielding records from DataReaderMapper rather than using a list # 254" and it is a feature request. The compile error mentioned is not relevant to study as the goal is to look at dynamic errors.

46. http://github.com/clojure/core.logic/commit/2406f2d31ebebcdaca4b3dd9c302b9b9688c6e0c
"‘fixc‘ as suspected ‘treec‘ is not quite flexible enough. Instead we now . . . "
Functionality change. Not a bug.

47. http://github.com/docpad/docpad/commit/610525940ab53c8fe0f9567c8a2c3f0be6d0adad
"...Thanks to [Ashton Williams] for issue # 595"
This is a bug.

48. http://github.com/GeertJohan/gorp/commit/d0f66577307520646053 2ba54158979deb4f56ec
"...If a table uses a reserved word for a column name (silly but happens) then there will be a SQL error when trying to fetch a mapped struct."
Functionality change. Not a bug.

49. http://github.com/MerlinDMC/gocode/commit/f36ed6ec9caf15cc7cf7fe8ec8d631ee34748d97
    "Add test.0012. No import statements issue. For some reason local parsing fails if there are no import statements."
    Not a bug fix, but a new test.

50. http://github.com/mpeltonen/sbt-idea/commit/6a707e2887bb385a49d6ddd5d1991f95ff4675e4
    "Use full path to fix Idea complaining about the Null import (Idea Scala plugin bug?)."
    Not a bug in code, but a bug in a plugin.

51. http://github.com/mpeltonen/sbt-idea/commit/b7c3895a7b3a1dd7489c9f084e4d9e7366152747
    "Download classifiers by default unless no-classifiers or . . . Fixes issue # 63"
    The issue is "Make with-classifiers and with-sbt-classifiers enabled by default # 63". this is a feature request.

52. http://github.com/0x43/DesignPatternsPHP/commit/623ad063305ce5240e47ebfdaad2bc17419fe75f
    "fix tiny typo"
    Change to comment. Not a bug.

53. http://github.com/AutoMapper/AutoMapper/commit/2eb1339249cc3cdeb734e8d93b34659fe9920d34
    "PrimitiveArrayMapper implementation Fixes # 279"
    The issue is "byte array to byte array is very slow. # 279" Functionality change. Not a bug.

54. http://github.com/Chenkaiang/XVim/commit/16a3d5d298bd0427e74e8f283c00a994b97b53ce
    "A handful of additional fixes to the way register playback their macros"
    . Not a bug.

55. http://github.com/docpad/docpad/commit/894b56a6ffe67825ef7d8e2282ae5752c27f59f7
    "Fix tests for importer rewrite. Caterpillar optimisation attempt. . . ."
    New feature and changes to tests. Not a bug.

56. http://github.com/faylang/fay/commit/bdc1aebb04e58d44a537d52573e63a4c63a6e1b0
    "Remove unused CompileErrors and do some renaming"
    Not a bug.

57. http://github.com/lfe/lfe/commit/0e6eca5d06f99d3cefd278eb4e7ca705b499b663
    "Handle compiler option warnings_as_errors We do this in the samw way as . . . "
    Functionality change. Not a bug.

58. http://github.com/magicalpanda/MagicalRecord/commit/45d764a58d4b11c0cbdeb10fa442a18680e492ff
    "Cleanup code is now debug-only"
    Not a bug.

59. http://github.com/MerlinDMC/gocode/commit/1863540842c9a5bb532d8fd75de6ede6cf5068ac
    "Fix semantic_rename/test.0002."
    This is likely a bug, the previous commit added tests. This is the fix in non-test code.

60. http://github.com/MerlinDMC/gocode/commit/975a2fc86646f4afae2b8414a8457f5f52c935b5
    "Make a global universeScope variable. Also fixes all TODOs regarding. . . ”
    Not a bug.

61. http://github.com/MerlinDMC/gocode/commit/bd72d4bc5941f538c3f789fb9a3d54a43043800b
    "Mutlifile packages support, few misc fixes."
    Functionality change. Not a bug. over 6 files and 401 locations.

62. http://github.com/mpeltonen/sbt-idea/commit/4f7acd829eb4b330ae263404dd56c8fb27b85993
    "Make classifiers of sources and javadocs configurable (issue # 97)"
    The issue is "Support configurable source/javadoc classifiers # 97" Functionality change. Not a bug.

63. http://github.com/0x43/DesignPatternsPHP/commit/6d5f72beec06cc9e1b4eef1d1b06fe9af9428fdb
    "Fix PHPDoc in FactoryMethod"
    Change to a comment.

64. http://github.com/Arcank/nimbus/commit/0c2a43ca2922cbda6916a97cee4bef8cfbfc67c2
    "[overview] Fix strict warnings."
    Not a bug.

65. http://github.com/Arcank/nimbus/commit/8691a0a009fe39872c756c8aeb06887a584dc18b
    "[core] Add NIError.h/m."
    Functionality change. Not a bug.

66. http://github.com/AutoMapper/AutoMapper/commit/20054717a055d7ced4e5daa66151b7cf0ab6a0df
"Win RT version compiling including unit tests"
73 changes files in 3K location. Functionality change. Not a bug.

67. http://github.com/yu19930123/ngrok/commit/22537c01364b867d0ba335e6024242752a9bda47
"add bug reporting instructions on crash"
Not a bug., changed the way errors are printed.

68. http://github.com/Chenkaiang/XVim/commit/29e8a92d6b2a54212113cb7e69957cec660eeac2
"Handful of fixes and better visual support Reorganized the way ..."
Refactoring, Not a bug.

69. http://github.com/K2InformaticsGmBH/proper/commit/13dbb5687464c8311de3be3e092d86d8bfb650c3
"Permit caller to "prime" PropEr's typeserver and random generator prior . . . "
Feature change not a bug.

70. http://github.com/0x43/DesignPatternsPHP/commit/f2074443076d0e2100014142ac3bce5c3aa080f6
"Fix typo"
Not a bug. Change to comments.

71. http://github.com/sinclairzx81/typescript.api/commit/109ef9b4c0515442cfc979271b3709e43748edad
"fix up to lib.d.ts"
7,897 additions and 1,107 deletions. Looks like catching up to changes.

References

[1] Baishakhi Ray, Prem Devanbu, and Vladimir Filkov. *Rebuttal to Berger et al., TOPLAS 2019*. 2019. arXiv: 1911.07393 [cs.SE].

[2] Emery D. Berger, Celeste Hollenbeck, Petr Maj, Olga Vitek, and Jan Vitek. "On the impact of programming languages on code quality: A reproduction study". In: *ACM Trans. Program. Lang. Syst.* 41.4 (Oct. 2019), 21:1–21:24. DOI: 10.1145/3340571. URL: http://doi.acm.org/10.1145/3340571.

[3] Baishakhi Ray, Daryl Posnett, Vladimir Filkov, and Premkumar T. Devanbu. "A large scale study of programming languages and code quality in GitHub". In: *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, (FSE-22), Hong Kong, China, November 16 - 22, 2014*. Ed. by Shing-Chi Cheung, Alessandro Orso, and Margaret-Anne D. Storey. ACM, 2014, pp. 155–165. DOI: 10.1145/2635868.2635922. URL: https://doi.org/10.1145/2635868.2635922.

[4] Baishakhi Ray, Daryl Posnett, Premkumar T. Devanbu, and Vladimir Filkov. "A large-scale study of programming languages and code quality in GitHub". In: *Commun. ACM* 60.10 (2017), pp. 91–100. DOI: 10.1145/3126905. URL: https://doi.org/10.1145/3126905.