Confusion Detection in Code Reviews

Felipe Ebert*†, Fernando Castor*, Nicole Novielli‡, Alexander Serebrenik†
*Federal University of Pernambuco, Brazil, {fe,castor}@cin.ufpe.br
†Eindhoven University of Technology, The Netherlands, {f.ebert, a.serebrenik}@tue.nl
‡University of Bari, Italy, nicole.novielli@uniba.it

***Abstract*—Code reviews are an important mechanism for assuring quality of source code changes. Reviewers can either add *general* comments pertaining to the entire change or pinpoint concerns or shortcomings about a specific part of the change using *inline* comments. Recent studies show that reviewers often do not understand the change being reviewed and its context.**

Our ultimate goal is to identify the factors that confuse code reviewers and understand how confusion impacts the efficiency and effectiveness of code review(er)s. As the first step towards this goal we focus on the identification of confusion in developers' comments. Based on an existing theoretical framework categorizing expressions of confusion, we manually classify 800 comments from code reviews of the Android project. We observe that confusion can be reasonably well-identified by humans: raters achieve moderate agreement (Fleiss' kappa 0.59 for the general comments and 0.49 for the inline ones). Then, for each kind of comment we build a series of automatic classifiers that, depending on the goals of the further analysis, can be trained to achieve high precision (0.875 for the general comments and 0.615 for the inline ones), high recall (0.944 for the general comments and 0.988 for the inline ones), or substantial precision and recall (0.696 and 0.542 for the general comments and 0.434 and 0.583 for the inline ones, respectively). These results motivate further research on the impact of confusion on the code review process. Moreover, other researchers can employ the proposed classifiers to analyze confusion in other contexts where software development-related discussions occur, such as mailing lists.

***Index Terms*—code review; confusion; machine learning.**

I. Introduction

Code reviews are an important mechanism for software quality assurance [1], [2], [3], [4], [5]. Active participation in code reviews has a tendency to decrease the number of post-release defects and improve software quality in open source projects [6], [7]. However, the time spent by a developer reviewing code is non-negligible [1] and may take up to 60% of the overall time spent on software development activities [8], [9]. Furthermore, performing code reviews can be difficult. Several studies [6], [10], [11], [12], [13] show that code reviewers often do not understand or are confused about the change under review and its context.

We do not make a distinction between lack of knowledge, confusion, or uncertainty. Lack of knowledge and confusion, which also encompass doubt and uncertainty, are strictly linked (*e.g.*, confusion could be determined by lack of knowledge) and are both actionable [14]. From now on, for simplicity we use the term "confusion" to refer to all these terms.

We conjecture that confusion acts as a mediating variable in the code review process: some attributes of the code being reviewed cause confusion, and confusion negatively affects efficiency and effectiveness of the review. To provide empirical evidence supporting or contradicting this conjecture, we need to identify the comments where the reviewers expressed confusion. This is the problem that this paper tackles. Once those cases have been identified, statistical techniques can be applied to confirm or refute the conjecture [15].

To identify the review comments expressing confusion we proceed in three steps. Since most of the code reviews do not express confusion, we employ a theoretical framework of confusion in computer-mediated discourse [16] to filter out the review comments that are likely not to express confusion. Next, we manually classify comments based on the opinions of four researchers. Finally, using the manually labeled data and features suggested in the aforementioned theoretical framework, we train a number of different classifiers to perform confusion identification automatically.

We apply our approach to 400 general and 400 inline code review comments randomly selected from the Android project. We found out that confusion can be reasonably well-identified by humans: the raters achieve moderate agreement (Fleiss' kappa 0.59 for the general comments and 0.49 for the inline ones). Using this manually-classified data, for each kind of comment we build a series of automatic classifiers that, depending on the goals of the further analysis, can be trained to achieve high precision (0.875 for the general comments and 0.615 for the inline ones), high recall (0.944 for the general comments and 0.988 for the inline ones), or substantial precision and recall (0.696 and 0.542 for the general comments and 0.434 and 0.583 for the inline ones, respectively). These results suggest that future statistical studies can be employed to analyze the causes and effects of confusion in code reviews. Furthermore, we observed that identification of confusion in inline comments is a more challenging task. It was evidenced in both manual labeling and the classifiers results.

The main contributions of this paper are: (1) to the best of our knowledge the first approach to identify confusion in code review comments; (2) an exploration of the application of 14 different classifiers to this problem; and (3) a gold standard dataset with the confusion annotations.[1]

II. Methodology

The main vehicle of this research is an explanatory case study [17] on confusion in code reviews. We *aim* at understanding how developers express confusion during code reviews. The *theoretical* framework for confusion identification

[1] https://github.com/spgroup/confusion

is based on the model of Jordan *et al.* [16], summarized in Section II-A. As a preliminary step towards building a classifier for confusion, we assess whether confusion can be identified by humans. Hence, our *research questions* are: (1) can human raters agree on the presence of confusion in code review comments?, and (2) is it possible to design a tool to recognize confusion in developers' comments?

A. Theoretical Framework

To the best of our knowledge, there exists no theoretical framework for identification of confusion in software engineering artifacts. Thus, we employ the framework created by Jordan *et al.* [16] for confusion in computer-mediated discourse. The framework contains seven categories of textual elements related to expression of confusion: *hedges*, *probables* and *hypotheticals* representing indirect expressions of uncertainty, *questions* requesting a solution, *I statements* and *nonverbals* describing direct psychological expression of uncertainty, and *meta* capturing the discussion of uncertainty. Each one of those categories is illustrated by several examples, *e.g.*, "sort of" and "maybe" are examples of hedges [16].

While the original framework [16] supports general reasoning about confusion, an obstacle to applying it in practice is challenged by the restricted list of linguistic features in each category provided by the authors [16]. Furthermore, Jordan *et al.* exclude words, *e.g.*, modal verbs, if they have other common uses besides expressing confusion. Therefore, with the exception of the Questions category, we augment the lists of features in each category based on additional linguistic sources [16], [18], [19], [20], [21], [22]. The complete list of features of each category is publicly available.[2]

B. Case Study Subject: Android

As the case study subject we select Android, a large and well-known open source project with a rigorous code review process and large number of publicly available code reviews.

Android uses Gerrit[3] as its code review system. Developers submit their changes into Gerrit, and invite others to review the changes. The change is merged only after being verified and approved by a senior developer. The web-interface provided by Gerrit allows reviewers to create *general* comments in the code review page, and *inline* comments in the source code file, referencing a word, a line or a group of lines.

C. Data Collection and Pre-processing

Using the Gerrit API we downloaded all Android code reviews until November 25, 2016 (the date we run the scripts). In total, we obtained data from 140,006 code reviews including 28,091 with inline comments. We have extracted 232,471 inline comments and 899,105 general comments. From the general comments, we have identified several bots, such as Treehugger Robot, Deckard Autoverifier, and Android Merger, and excluded comments authored by these bots from our dataset leaving 660,845 general comments for further analysis.

[2]https://github.com/spgroup/confusion

[3]http://www.gerritcodereview.com

Next using the extended framework we exclude the comments that do not contain features associated with confusion. We keep 91,658 general and 116,292 inline comments.

D. Manual Labeling

As a first step to identify comments expressing confusion, we evaluate the *hedges* category from the framework because they represent the majority of the comments: 97% for general and 87% for inline. We plan to evaluate the remaining categories in the future work.

A random sample of 25 comments was extracted from our data and manually classified as *confusion* or *no confusion*. The authors have agreed upon the labels of the sample and used it as a guideline during the labeling process. To obtain a confidence interval less than 5%, we randomly sampled 400 general and 400 inline comments. Four raters, having at least a master's degree in Computer Science, have manually classified the comments. The raters were instructed to consider as *no confusion* comments that employ uncertainty to express politeness, *e.g.*, "Maybe add checker tests to make sure you cover the cases you intended?". The raters worked individually.

We used Fleiss' kappa to measure the agreement for general and inline comments. The disagreement was resolved with online meetings with all four raters. In the rare cases when the agreement between the raters was not possible, we decided to discard the comments. Four general and four inline comments have been discarded, leaving 396 general and 396 inline comments that constitute the *gold standard set*.

E. Classifier

Before training the classifier we first remove the line breaks and replace urls, numbers, commits' Sha1 and user names with meta tokens (*e.g.*, @URL, USERNAME, COMMIT and NUMBER). To identify the user names we leveraged the name list collection by Vasilescu *et al.* [23]. We exploit machine learning techniques using our gold standard for training and validation. We experimented with several state-of-the-art classifiers using Weka [24]. To understand the impact of the feature choice on the classifier performance we run the classifiers over three different feature settings.

Features of the baseline model are uni- and bigrams extracted by the unsupervised StringToWordVector filter and selected based on the Term Frequency Inverse Document Frequency (TF-IDF). The "Baseline + 3" model extends the baseline by including i) the modal verbs count, ii) the count of the hedges from the extended framework and iii) the presence of a question mark. In this model we do not distinguish between different hedges or modal verbs. The "Baseline + hedges" model also adds to the baseline hedges from the extended framework but as opposed to "Baseline + 3" it considers different hedges as different features.

To assess the performance of different classifiers we compare them against the ZeroR classifier, which always predicts the majority class, and random guessing. We run our experiments in a 10-fold cross-validation setting, using stratified sampling as implemented by Weka.

III. Results

The manual labeling agreement between the four rates measured with Fleiss' κ is 0.59 for general and 0.49 for inline comments. We believe those results reveal that humans can, indeed, reasonably identify confusion in code review comments (κ between 0.41 and 0.60 is considered moderate [25]). After solving the disagreement, we observe that confusion is expressed both in the general (18%) and in the inline (21%) comments, and while there are slightly more confusion comments among the inline ones, the association between the kind of comments and presence of confusion is not statistically significant ($p \simeq 0.33$ for the Fisher's exact test.)

The results of the classifiers are shown in Table I. The best precision on confusion class is obtained by OneR for both general and inline comments albeit with different feature settings: for general comments, the best model is Baseline + 3 (0.875), and for inline ones—Baseline + hedges (0.615). For recall, the best classifier is Multinomial Naive Bayes, for both general (0.944) and inline comments (0.998). It achieves the same performance for all models, both for general and inline comments. Regarding the balance between precision and recall the best classifier is JRip for general (0.609) and Logistic for inline comments (0.497). In both cases the model with best performance is Baseline + 3.

IV. Discussions

We discuss the results IV-A, their implications IV-B, and threats to validity of our conclusions IV-C.

A. Detection of confusion

The interrater agreement suggests that identification of confusion can be reliably performed by human raters. This confirms the reliability of our annotation schema and the resulting golden sets. However, dealing with small, highly unbalanced dataset is something undesirable when training a classifier in a supervised machine learning setting [26]. Still, our preliminary results confirm that automatic detection of confusion in both general and inline comments is feasible.

More specifically, we observe that more advanced models Baseline + 3 and Baseline + hedges tend to outperform the Baseline model, and none of the best classifiers reviewed above makes use of the Baseline model. This means that addition of more specific features geared towards detection of confusion is indeed beneficial.

Highest recall is observed for both general and inline comments when using Multinomial Naive Bayes, regardless of the feature setting. This is consistent with previous evidence in literature showing how Multinomial Naive Bayes outperforms other approaches when dealing with small, unbalanced training set with few positive examples [27], as in our case.

While differences between general and inline comments affect the precision and recall figures, the same classifiers seem to perform best. This is, however, not the case for the F-measure. Overall, reasonably high precision and recall have been obtained, enabling future statistical studies of causes and effects of confusion in code reviews.

The best precision for the general comments is higher than the one for the inline comments, suggesting that identification of confusion in inline comments is a more challenging task. This observation concurs with the previous observation that identification of confusion in inline comments turned out to be more difficult to the human raters as well. Alternatively, one might need different predictors, *e.g.*, the context of the code change, to detect confusion in inline comments. However, OneR produces a very simplistic model, *i.e.*, a set of rules operating on a single predictor, which may be poorly generalizable on new unseen data.

B. Implications

As direct implications, the classifiers presented enable statistical studies of causes and effects of confusion in code reviews. Moreover, researchers now have evidence that inline comments are prevalent and relevant for future code review studies.

Later implications of our ultimate goal have larger scope. *Software developers* can learn from the reasons that make code changes hard to understand (and to accept), thus they will be able to write code changes and submit them in ways that attempt to avoid confusion. *Tool builders* can benefit by expanding static analysis tools so as to more comprehensively identify factors that make code changes hard to understand. *Educators* can train their students from the outset to write better code changes. *Researchers* will be able to propose solutions to overcome the problems that confusion in code reviews bring forth, *e.g.*, by including more context information in the code reviews. Researchers also can use our approach as a basis to identify confusion in a number of different contexts, with different implications, *e.g.*, bug reports and the associated discussions (commit messages, email discussions, and design and requirements documentation).

C. Threats to validity

The threats to *construct validity* are related to how properly a measurement reflects the concept being studied. Identifying confusion is not an easy task, and its has been being operationalized using keywords. We used an existing confusion framework [16] and also considered features from different sources [16], [18], [19], [20], [21], [22]. Considering the *internal validity* we note that none of the raters has been involved in Android development, so they might have misinterpreted certain comments as confusion or no confusion. However, all raters are computer scientists and two of the four have a substantial experience with labeling textual information. All disagreements were solved with a meeting with all raters. The threats to *external validity* are related to the generalizability of the study results. Our study targeted only Android, this means that other projects might have different results. Furthermore, rebalancing was not applied to counteract majority class bias that inherently affects our data. Learning from imbalanced data poses new emerging challenges that need to be addressed to build robust models of knowledge from raw data [26]. Replications are needed with larger datasets, using machine learning

Classifier	Class	General comments									Inline comments								
		Baseline			Baseline + 3			Baseline + hedges			Baseline			Baseline + 3			Baseline + hedges		
		P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
Naive Bayes	C	0.395	0.472	0.430	0.444	0.500	0.471	0.404	0.500	0.447	0.381	0.536	0.446	0.393	0.548	0.458	0.383	0.548	0.451
	NC	0.877	0.840	0.858	0.886	0.861	0.873	0.883	0.836	0.859	0.860	0.766	0.810	0.864	0.772	0.816	0.862	0.763	0.810
Multinomial Naive Bayes	C	0.209	0.944	0.342	0.209	**0.944**	0.342	0.211	0.944	0.345	0.234	0.988	0.378	0.234	**0.988**	0.378	0.234	0.988	0.379
	NC	0.943	0.204	0.335	0.943	0.204	0.335	0.946	0.216	0.352	0.976	0.128	0.227	0.976	0.128	0.227	0.976	0.131	0.232
Logistic	C	0.344	0.611	0.440	0.362	0.639	0.462	0.352	0.597	0.443	0.431	0.560	0.487	**0.434**	**0.583**	0.497	0.431	0.560	0.487
	NC	0.896	0.741	0.811	0.903	0.750	0.820	0.894	0.756	0.819	0.871	0.801	0.835	0.876	0.795	0.834	0.871	0.801	0.835
Simple Logistic	C	0.474	0.125	0.198	0.680	0.236	0.351	0.500	0.125	0.200	0.455	0.179	0.256	0.500	0.107	0.176	0.515	0.202	0.291
	NC	0.833	0.969	0.896	0.852	0.975	0.909	0.833	0.972	0.897	0.810	0.942	0.871	0.802	0.971	0.878	0.815	0.949	0.877
SMO	C	0.412	0.097	0.157	0.500	0.139	0.217	0.389	0.097	0.156	0.515	0.202	0.291	0.528	0.226	0.317	0.500	0.202	0.288
	NC	0.828	0.969	0.893	0.835	0.969	0.897	0.828	0.966	0.892	0.815	0.949	0.877	0.819	0.946	0.878	0.815	0.946	0.875
IBk	C	0.304	0.097	0.147	0.269	0.097	0.143	0.294	0.139	0.189	0.333	0.012	0.023	0.250	0.012	0.023	0.100	0.012	0.021
	NC	0.826	0.951	0.884	0.824	0.941	0.879	0.829	0.926	0.875	0.789	0.994	0.879	0.788	0.990	0.878	0.785	0.971	0.868
KStar	C	0.182	1.000	0.308	0.182	1.000	0.308	0.182	1.000	0.308	0.212	1.000	0.350	0.212	1.000	0.350	0.212	1.000	0.350
	NC	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
JRip	C	0.563	0.125	0.205	**0.696**	**0.542**	0.609	0.643	0.125	0.209	0.571	0.143	0.229	0.433	0.155	0.228	0.609	0.167	0.262
	NC	0.834	0.978	0.901	0.903	0.948	0.925	0.835	0.985	0.904	0.808	0.971	0.882	0.806	0.946	0.870	0.812	0.971	0.885
OneR	C	0.571	0.056	0.101	**0.875**	0.194	0.318	0.571	0.056	0.101	0.234	0.988	0.378	0.462	0.071	0.124	**0.615**	0.095	0.165
	NC	0.825	0.991	0.900	0.847	0.994	0.915	0.825	0.991	0.900	0.976	0.128	0.227	0.796	0.978	0.878	0.802	0.984	0.883
J48	C	0.364	0.167	0.229	0.595	0.347	0.439	0.316	0.167	0.218	0.303	0.119	0.171	0.429	0.250	0.316	0.314	0.131	0.185
	NC	0.835	0.935	0.882	0.867	0.948	0.906	0.832	0.920	0.874	0.796	0.926	0.856	0.818	0.910	0.862	0.798	0.923	0.856
Random Forest	C	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	NC	0.818	0.997	0.898	0.817	0.994	0.897	0.817	0.994	0.897	0.788	1.000	0.881	0.788	1.000	0.881	0.788	1.000	0.881
REPTree	C	0.364	0.056	0.096	0.636	0.486	0.551	0.385	0.069	0.118	0.424	0.167	0.239	0.500	0.214	0.300	0.394	0.155	0.222
	NC	0.823	0.978	0.894	0.891	0.938	0.914	0.825	0.975	0.894	0.807	0.939	0.868	0.817	0.942	0.875	0.804	0.936	0.865
Random Guessing	C	0.181	0.500	0.265	0.181	0.500	0.265	0.181	0.500	0.265	0.212	0.500	0.297	0.212	0.500	0.297	0.212	0.500	0.297
	NC	0.818	0.500	0.620	0.818	0.500	0.620	0.818	0.500	0.620	0.787	0.500	0.611	0.787	0.500	0.611	0.787	0.500	0.611
ZeroR (Majority)	C	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	NC	0.818	1.000	0.900	0.818	1.000	0.900	0.818	1.000	0.900	0.788	1.000	0.881	0.788	1.000	0.881	0.788	1.000	0.881

TABLE I

CLASSIFIER RESULTS: C—CONFUSION CLASS, NC—NO CONFUSION CLASS; P–PRECISION, R—RECALL, F—THE F-MEASURE.

tehcniques specifically designed to deal with skewness in class distribution, to further assess the generality of our models.

V. RELATED WORK

Code review has been extensively studied in recent works [2], [10], [11], [28], [29], [30], [31], [32], [33], [34]. Bacchelli and Bird [10] presented the concept of modern code review as a review that is informal, supported by tools, and happens frequently. They show the main challenge of code review is understanding the code change. Tao *et al.* [11] show that the rationale of the code change is the most important information for understanding the code review. Our work differs from those because we are focusing in confusion in code review comments. Furthermore, to the best of our knowledge, none of the studies that provide dataset for code reviews [30], [31], [32], [33] covers inline comments. In our dataset we present both general and inline comments.

Confusion detection is one of the examples of detecting mental states [35], [36]. Yang *et al.* [35] described a classification model using discussion forum behavior (comments) and clickstream data to automatically identify posts that express confusion. Their confusion model is based on i) users' click patterns, ii) users' linguistic features based on LIWC[4] words, and iii) questions features. They also tried to understand what users are confused about by looking to the recent click behavior. As their confusion model is not publicly available, we could not compare to ours. Jean *et al.* [36] proposed a new supervised and generic approach to automatically detect uncertain expressions within natural language. It is based on the statistical analysis of multiple lexical and syntactic features used to characterize sentences through vector-based representations. We could apply their uncertainty method to our dataset, however we tend to disagree with Jean *et al.*on what constitutes confusion. For instance, "Could anyone submit this?" and "I rebased could you review again please." are examples of non-comments from our dataset tagged as confusion by the method of Jean *et al.* [36].

A related but distinct problem is detection of affective states, a topic that has recently attracted substantial attention in the software engineering community [37], [38], [39], [40], [41]. Similarly to confusion detection, detection of affective states can be formulated as a classification problem; furthermore, similarly to our expectation that confusion is related to effectiveness of code review, affective states of developers have been related to effectiveness of their collaboration [42].

VI. CONCLUSIONS

In this paper we presented a framework for identifying confusion in textual comments and also analyzed whether confusion can be reasonable recognized by humans and automatically. The agreement among the raters shows that confusion can be indeed recognized by humans. Furthermore, confusion identification in inline comments has been shown to be more a difficult task than in general ones. We believe that is due to inline comments usually containing less information and also being more context dependent than the general comments as the former ones are directly related to certain parts of the code.

We construct the gold standard set of 396 general and 396 inline comments, and evaluate several classifiers. We observe that models with more specific features are beneficial towards

[4]https://liwc.wpengine.com

detection of confusion. Multinomial Naive Bayes presents the highest recall for both general and inline comments, regardless of the feature setting. OneR classifier is the best for high precision for both general and inline comments. However, to achieve substantial precision and recall, there results are different for general and inline comments. For the former, the best classifier is the JRip classifier, and for the latter, the best one is the Logistic classifier.

As the future work we plan to increment our model with the other categories from our framework and then train new classifiers. Thereafter, through statistical modeling and developer surveys we plan to study the reasons for confusion in code reviews, as well as the impact of confusion on the code review duration and outcome.

VII. Acknowledgment

This research was partially funded by CNPq/Brazil (304755 /2014-1, 406308/2016-0, 465614/2014-0), CAPES/Brazil (PDSE-88881.132399/2016-01), FACEPE/Brazil (APQ-0839-1.03/14, 0388-1.03/14, 0791-1.03/13), and by the project "EmoQuest", funded by the Italian Ministry for Education, University and Research under the SIR program.

References

[1] Y. Tao and S. Kim, "Partitioning composite code changes to facilitate code review," in *MSR*. IEEE, 2015, pp. 180–190.

[2] G. Bavota and B. Russo, "Four eyes are better than two: On the impact of code reviews on software quality," in *ICSME*, 2015, pp. 81–90.

[3] B. Boehm and V. R. Basili, "Top 10 list [software development]," *Computer*, vol. 34, no. 1, pp. 135–137, 2001.

[4] M. V. Mäntylä and C. Lassenius, "What types of defects are really discovered in code reviews?" *TSE*, vol. 35, no. 3, pp. 430–448, 2009.

[5] M. Barnett, C. Bird, J. Brunet, and S. K. Lahiri, "Helping developers help themselves: Automatic decomposition of code review changesets," in *ICSE*. IEEE, 2015, pp. 134–144.

[6] J. Cohen, S. Teleki, and E. Brown, *Best Kept Secrets of Peer Code Review*. Smart Bear Inc., 2006.

[7] S. McIntosh, Y. Kamei, B. Adams, and A. E. Hassan, "An empirical study of the impact of modern code review practices on software quality," *ESE*, pp. 1–44, 2015.

[8] B. Cornelissen, A. Zaidman, A. van Deursen, L. Moonen, and R. Koschke, "A systematic survey of program comprehension through dynamic analysis," *TSE*, vol. 35, no. 5, pp. 684–702, 2009.

[9] H. C. Benestad, B. Anda, and E. Arisholm, "Understanding software maintenance and evolution by analyzing individual changes: a literature review," *JSME*, vol. 21, no. 6, pp. 349–378, 2009.

[10] A. Bacchelli and C. Bird, "Expectations, outcomes, and challenges of modern code review," in *ICSE*. IEEE, 2013, pp. 712–721.

[11] Y. Tao, Y. Dang, T. Xie, D. Zhang, and S. Kim, "How do software engineers understand code changes?: An exploratory study in industry," in *FSE*. ACM, 2012, pp. 51:1–51:11.

[12] A. Sutherland and G. Venolia, "Can peer code reviews be exploited for later information needs?" in *ICSE-Companion*, 2009, pp. 259–262.

[13] T. D. LaToza, G. Venolia, and R. DeLine, "Maintaining mental models: A study of developer work habits," in *ICSE*. ACM, 2006, pp. 492–501.

[14] S. D'Mello and A. Graesser, "Confusion and its dynamics during device comprehension with breakdown scenarios," *Acta Psychologica*, vol. 151, pp. 106–116, 2014.

[15] R. M. Baron and D. A. Kenny, "The moderator–mediator variable distinction in social psychological research: Conceptual, strategic, and statistical considerations," *Journal of personality and social psychology*, vol. 51, no. 6, pp. 1173–1182, 1986.

[16] M. E. Jordan, D. L. Schallert, Y. Park, S. Lee, Y. hui Vanessa Chiang, A.-C. J. Cheng, K. Song, H.-N. R. Chu, T. Kim, and H. Lee, "Expressing uncertainty in computer-mediated discourse: Language as a marker of intellectual work," *Discourse Processes*, vol. 49, no. 8, pp. 660–692, 2012.

[17] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *ESE*, vol. 14, no. 2, pp. 131–164, 2009.

[18] J. Holmes, "Expressing doubt and certainty in english," *RELC Journal*, vol. 13, no. 2, pp. 9–28, 1982.

[19] M. E. Jordan and R. R. McDaniel Jr., "Managing uncertainty during collaborative problem solving in elementary school teams: The role of peer influence in robotics engineering activity," *Journal of the Learning Sciences*, vol. 23, no. 4, pp. 490–536, 2014.

[20] M. E. Jordan, A.-C. J. Cheng, D. Schallert, K. Song, S. Lee, and Y. Park, ""I guess my question is": What is the co-occurrence of uncertainty and learning in computer-mediated discourse?" *Journal of Computer-Supported Collaborative Learning*, vol. 9, no. 4, pp. 451–475, 2014.

[21] G. Lakoff, *Hedges: A Study in Meaning Criteria and the Logic of Fuzzy Concepts*. Springer, 1975, pp. 221–271.

[22] T. Varttala, "Hedging in the scientifically oriented discourse. exploring variation according to discipline and intended audience," Ph.D. dissertation, University of Tampere, 2001. [Online]. Available: https://tampub.uta.fi/handle/10024/67148

[23] B. Vasilescu, A. Capiluppi, and A. Serebrenik, "Gender, representation and online participation: A quantitative study," *Interacting with Computers*, vol. 26, no. 5, pp. 488–511, 2014.

[24] E. Frank, M. A. Hall, and I. H. Witten, "The WEKA workbench," 2016, appendix for "Data Mining: Practical Machine Learning Tools and Techniques". [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/Witten_et_al_2016_appendix.pdf

[25] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data," *Biometrics*, vol. 33, no. 1, 1977.

[26] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. on Knowl. and Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.

[27] G. Forman and I. Cohen, "Learning from little: comparison of classifiers given little training," in *PKDD*. Springer, 2004, pp. 161–172.

[28] O. Kononenko, O. Baysal, L. Guerrouj, Y. Cao, and M. W. Godfrey, "Investigating code review quality: Do people and participation matter?" in *ICSME*, 2015, pp. 111–120.

[29] M. Hentschel, R. Hähnle, and R. Bubel, "Can formal methods improve the efficiency of code reviews?" in *IFM*. Springer, 2016, pp. 3–19.

[30] M. Mukadam, C. Bird, and P. C. Rigby, "Gerrit software code review data from android," in *MSR*. IEEE, 2013, pp. 45–48.

[31] K. Hamasaki, R. G. Kula, N. Yoshida, A. E. C. Cruz, K. Fujiwara, and H. Iida, "Who does what during a code review? datasets of oss peer review repositories," in *MSR*. IEEE, 2013, pp. 49–52.

[32] P. Thongtanunam, X. Yang, N. Yoshida, R. G. Kula, A. E. C. Cruz, K. Fujiwara, and H. Iida, "Reda: A web-based visualization tool for analyzing modern code review dataset," in *ICSME*, 2014, pp. 605–608.

[33] X. Yang, R. G. Kula, N. Yoshida, and H. Iida, "Mining the modern code review repositories: A dataset of people, process and product," in *MSR*. ACM, 2016, pp. 460–463.

[34] P. van Wesel, B. Lin, G. Robles, and A. Serebrenik, "Reviewing career paths of the openstack developers," in *ICSME*, 2017.

[35] D. Yang, M. Wen, I. Howley, R. Kraut, and C. Rose, "Exploring the effect of confusion in discussion forums of massive open online courses," in *ACM Conference on Learning @ Scale*. ACM, 2015, pp. 121–130.

[36] P.-A. Jean, S. Harispe, S. Ranwez, P. Bellot, and J. Montmain, "Uncertainty detection in natural language: A probabilistic model," in *International Conference on Web Intelligence, Mining and Semantics*. New York, NY, USA: ACM, 2016, pp. 10:1–10:10.

[37] E. Guzman, D. Azócar, and Y. Li, "Sentiment analysis of commit comments in github: an empirical study," in *MSR*, 2014, pp. 352–355.

[38] A. Murgia, P. Tourani, B. Adams, and M. Ortu, "Do developers feel emotions? an exploratory analysis of emotions in software artifacts," in *MSR*, 2014, pp. 262–271.

[39] D. Pletea, B. Vasilescu, and A. Serebrenik, "Security and emotion: sentiment analysis of security discussions on github," in *MSR*, 2014, pp. 348–351.

[40] N. Novielli, F. Calefato, and F. Lanubile, "The challenges of sentiment detection in the social programmer ecosystem," in *SSE @ FSE*, 2015, pp. 33–40.

[41] D. Gachechiladze, F. Lanubile, N. Novielli, and A. Serebrenik, "Anger and its direction in collaborative software development," in *ICSE-NIER*, 2017, pp. 11–14.

[42] M. Ortu, B. Adams, G. Destefanis, P. Tourani, M. Marchesi, and R. Tonelli, "Are bullies more productive? empirical study of affectiveness vs. issue fixing time," in *MSR*, 2015, pp. 303–313.