Communicative Intention in Code Review Questions

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

***Abstract*—During code review, developers request clarifications, suggest improvements, or ask for explanations about the rationale behind the implementation choices. We envision the emergence of tools to support developers during code review based on the automatic analysis of the argumentation structure and communicative intentions conveyed by developers' comments. As a preliminary step towards this goal, we conducted an exploratory case study by manually classifying 499 questions extracted from 399 Android code reviews to understand the real communicative intentions they convey. We observed that the majority of questions actually serve information seeking goals. Still, they represent less than half of the annotated sample, with other questions being used to serve a wider variety of developers' communication goals, including suggestions, request for action, and criticism. Based on our findings we formulate hypotheses on communicative intentions in code reviews that should be confirmed or rejected by follow-up studies.**

***Index Terms*—questions; communicative intention; code reviews; exploratory case study; Android.**

I. Introduction

During a code review, developers might identify a defect, suggest a better solution, or ask about the rationale behind the implementation choices. As such, code review discussions represent an invaluable source of information ready to be mined for i) extracting information about a software project and its evolution [1] and ii) understanding how developers conduct code review, *i.e.*, which understanding needs they try to fulfill [2] and what comments they perceive as useful [3], [4]. We believe that the identification of communicative intentions expressed by developers in their comments, such as making a direct suggestion, requesting a clarification, and expressing disagreement, is essential to this latter perspective. In line with this view, Viviani *et al.* [1] recently proposed to mine developers' discussions in pull requests to extract design information that explicitly documents design decisions [5], based on the analysis of the dialogue argumentative structure [1].

We envision the emergence of tools to support developers during code review based on the automatic analysis of the communicative goals conveyed by developers' comments.[1] To this aim, we present the first study of the communicative intentions expressed in code review questions. We focus on questions as they have been recently described as triggers of useful conversation excerpts in code review, *i.e.*, in design discussions [1], [5] or in knowledge-sharing [2].

[1]The first author was also affiliated to the Eindhoven University of Technology, The Netherlands, during this study.

We performed an exploratory study [6] on the questions asked by developers in the inline comments in Android code reviews. We manually classified 499 questions derived from 399 code reviews in Gerrit. Our findings suggest that questions in code review serve diverse communicative goals, *e.g.*, requesting clarifications, discussing hypothetical scenarios, and suggesting improvements. We observed that the majority of questions serve information seeking goals. Still, they represent less than half of the annotated sample, providing evidence that questions may convey a wider variety of developers' communicative intention. A large category is represented by the questions aimed at eliciting an action of the collaborators, *i.e.*, developers usually employ politeness when making suggestions by using questions instead of affirmative sentences.

Code change authors interviewed by Bosu *et al.* [3] consider clarification questions as "not useful" as they do not immediately contribute to code improvement. However, such questions have been reported as useful to improve the reviewer's understanding of the change under review, which in turn can lead to improvement suggestions being formulated later on. Furthermore, those clarification questions can be useful to trigger knowledge transfer discussion between the contributors of a software project. This was mentioned as a reason for conducting code reviews, beyond finding defects, by all but one of the interviewees in the study of Bacchelli and Bird [2].

Gachechiladze *et al.* argued that identification of anger can be used to design tools and interventions supporting developers [7]. Complementarily, identification of communicative intention can provide a starting point in recommendations for good practices. If, *e.g.*, polite requests for action rather than direct suggestions are more common among code reviews deemed to be useful [3], then developers can be encouraged to embrace politeness (cf. discussion of how to ask for help on StackOverflow [8] and envisioned guidelines on how to communicate efficiently in code reviews [4]). Identifying requests for clarification may also benefit development of a recommender system identifying lack of reviewer expertise and triggering expert intervention [3]. Moreover, requests for rationale or suggestions of alternative solutions are steps towards identification of design discussion in code review [1].

Our findings, while preliminary, suggest research hypotheses worth investigating in future work. Following the guidelines of Runeson and Höst [6], we formulate a set of hypotheses that should be confirmed or refuted by follow-up studies.

II. Background

Communicative Intention. People proceed in their conversations through a series of *speech acts* to yield a specific communicative intention: they ask for information, agree with their interlocutor, state facts and express opinions [9]. Speech acts constitute the basic unit of verbal communication and are well studied in linguistics and computational linguistics [10], [11]. For the communication to be effective, all persons involved in the conversation must agree on the intention of the message [12]. Indeed, overt communication can be seen as an error-prevention strategy [13] and misinterpretations can arise from misunderstanding of the communicative intention [14]. In this study, we focus on the communicative intentions conveyed by developers during code review. Specifically, we analyze their communicative goals, *i.e.*, the *illocutionary force* [10] of the questions asked during code review.

Code Reviews. Code review is a widely employed practice of software quality assurance where developers inspect the code changes before they are integrated into the main repository [15], [16]. When reviewing code changes, developers might provide either general or inline comments. General comments are posted in the code review page itself, which presents the list of all general comments as threads of messages. Inline comments are posted directly in the source code file and can reference a word, a line or a group of lines. They are intended to be a dialogue between the reviewer and the code change author, as recognized by developers themselves, *e.g.*, Alan Fineberg, software engineer at Yelp: "*The reviewer then goes through the diff, adds inline comments on review board and sends them back. [...] The reviews are meant to be a dialogue, so typically comment threads result from the feedback*"[2]. As such, we focus on questions extracted from inline comments.

III. Methodology

We seek to answer the following research question: *what are the communicative intentions expressed in the developers' questions in code reviews?*

A. Annotation Sample from Android Code Reviews

We conduct an explanatory case study [6] on Android because it is a large and well-known open source ecosystem that adopts a rigorous code review process using Gerrit. We extracted our annotation sample from the dataset of inline comments we previously collected from the entire Android ecosystem [17]. Our annotation sample contains 499 questions extracted from randomly selected 399 inline comments (corresponding to the confidence interval of less than 5% and confidence level of 95% from a population of 10,965 questions). The questions are extracted by running the StanfordNLP API, a state-of-the-art NLP toolkit [18], on the 399 selected inline comments. As result, it could identify 442 questions.[3]

[2]Quoted by Marty Stepp in their slides https://courses.cs.washington.edu/courses/cse403/13sp/lectures/10-codereviews.pdf

[3]$57 = 499 - 442$ additional questions were not identified by StanfordNLP API but added during the manual labeling.

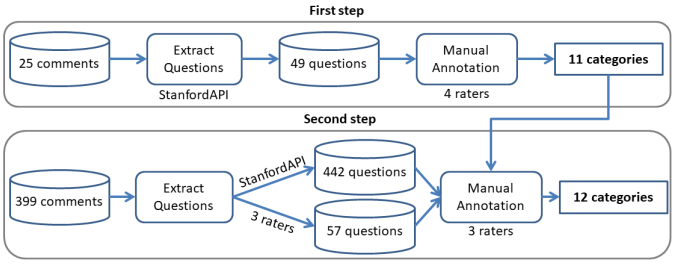

Fig. 1. Methodology adopted in the manual labeling.

B. Manual Labeling

We performed manual annotation of questions extracted from code review comments in two steps involving four and three raters, respectively (see Fig. 1). The first step aimed at defining the coding schema using an open coding methodology, *i.e.*, without predefined categories [19]. During this step, the four raters individually annotated the 49 questions extracted from 25 randomly sampled comments. The annotation was performed at the sentence level, *i.e.*, the individual questions rather than comment as a whole are used as unit of analysis. However, the whole comment was presented to provide context for the annotation. The raters were requested to assign a single label to each question indicating the communicative intention it conveyed. The disagreements were solved through online discussions. Based on the first annotation step we designed an initial taxonomy including 11 question categories that were used as guidelines for annotation during the second step. Once again, the raters were requested to individually label the communicative intention of 442 questions. They were instructed to perform the labeling based on the 11 categories in the initial taxonomy. However, they could suggest any missing categories they found relevant. Once the individual annotation was completed, we compared the labels from each rater and compiled them into one final set.

As a result, one new category was added to the initial ones, *i.e.*, criticism, leading to the final list of 12 categories. We used Fleiss' kappa to measure the agreement [20] and majority voting—to resolve the disagreements. If all the votes were different, the raters hold an online discussion.

During the manual annotation of the 442 questions, the raters found 57 additional questions not identified by the StanfordNLP API. These questions were included in our analysis of communicative intentions, thus resulting in 499 questions overall in our manually annotated dataset. However, the annotation of those 57 questions did not include any new question category. The raters discarded 2 declarative sentences that were erroneously classified by the StanfordNLP API as questions. Furthermore, 1% (5) of the questions were discarded given the inability of the raters to reach an agreement on the label to assign to them. In the end, our sample included 492 questions. The dataset of questions with corresponding communicative intention labels is publicly available for research purposes.[4]

[4]https://goo.gl/Bpoqj6

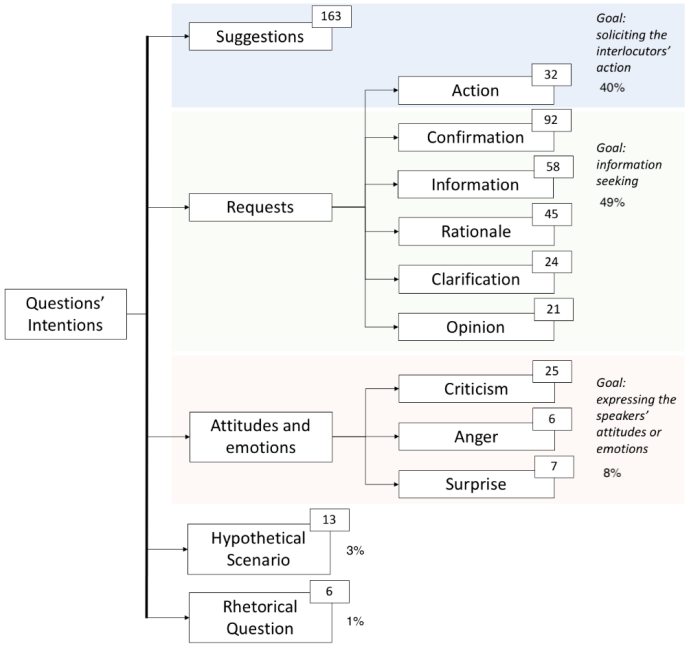

Fig. 2. Questions' intention classification tree.

IV. Results

The manual labeling agreement between the three raters measured with Fleiss' κ is 0.40, indicating a moderate agreement [21]. Figure 2 presents the questions' category classification tree, which we discuss in the following.

1) ***Soliciting the interlocutor's action****:* A considerable amount of questions (40%) is used to elicit an action of the interlocutor (see Figure 2). Specifically, 33% (163) of questions actually are **suggestions** for an alternative solution. The use of a question rather than an affirmative sentence to recommend a different implementation strategy might indicate the reviewer's attempt of being polite towards the change author, as in "*Maybe introduce an additional line between 'abc' and 'def'?*". In 7% (32) of cases, reviewers rather make a direct **request for action**, *i.e.*, they directly ask the change authors to perform a specific action as in "*Can you make these different? [...]*". As opposed to suggestions, requests for action explicitly refer to the individual that is expected to take action.

2) ***Information seeking****:* The majority of questions (49%) serve information seeking goals. Specifically, developers try to fulfill different information needs while reviewing code changes, *i.e.*, they ask questions to clarify their understanding of the code change. More than 18% (92) of questions are **request for confirmation**, *i.e.*, the reviewer has a certain degree of understanding but expresses doubt and seeks approval from the code change author as in "*shouldn't this just be a failure? [...]*". In 11% (58) of cases, reviewers perform **requests for information**, *i.e.*, reviewers have partial understanding of the problem and ask a question aimed at obtaining missing technical details required to complement the understanding, as in "*When can this be null? [...]*". In 9% (45) of cases, developer performs a **request for rationale** to understand the reason behind an implementation choice (*e.g.*, "*Why is this included? [...]*"). **Requests for clarification** occur when the reviewers miss overall understanding of the problem, *e.g.*, "*What's happening here?*". Requests for clarification occur in 4% (24) of the questions. Finally, developers also make comments aimed at eliciting others' opinions (**request for opinion**), as in *e.g.*, "*Which name do you suggest?*".

3) ***Expressing attitudes and emotions****:* In 8% of questions, the actual developers' communicative intention is to express their attitudes, opinions or emotions. Developers express their attitude of doubt through criticisms and even share such attitude of perplexity or disagreement by communicating their emotions with different degrees of strength, from surprise to anger. In all these cases, the final goal of the speaker is to express their own cognitive and emotional state to induce critical reflection in the interlocutor. About 5% (25) of the questions contain some level of negativity but could not be classified as expressing anger, they rather express **criticism** towards an implementation choice made by the interlocutor (*e.g.*, "*Do you really want to return the address of a local variable here?*"). We also observe few cases where developers expressed their emotions through questions. They represent 2% (13) of our question dataset, of which 1.22% (6) were classified as **anger** and 1.42% (7) as **surprise**. As an example of anger: "*wtf? you _really_ want reflection here.*", and of surprise: "*is this true? that seems mildly surprising. [...]*".

4) ***Hypothetical scenarios****:* Questions describing hypothetical scenarios (2%) may serve different communication purposes related to *what-if* scenarios, from information seeking to expression of criticism or doubt. Since context is needed to disambiguate the speakers' intention, we decided not to map *a priori* the hypothetical scenarios to categories discussed above. Developers tend to use wh-questions in combination with conditional conjunctions to describe hypothetical scenarios, *e.g.*, "*What about if an already Jack server is running?*".

5) ***Rhetorical questions****:* These are usually used when developers raise a question to answer it later themselves. Such questions serve the communicative goal of providing evidence and argument to support the claim made right after the question. They constitute only 1.22% (6) of the questions, *e.g.*, "*Isn't the case that you illustrated (0.9ms being decremented as 0) applicable in both solutions? Yes, [...]*".

V. Discussion

In line with findings from previous research suggesting that code review also serves knowledge-transfer purposes [2], we observe that the majority of questions are information seeking requests, including requests for confirmation, information, rationale, clarification, and opinion. This suggests a rich variety of information needs experienced by developers during code review. While representing the majority of questions, information seeking requests are actually less than half of the annotated sentences in our dataset (49%). In particular, the second most frequent goal addressed by developers' questions is to elicit a reaction of the interlocutor (40%). Finally, developers use questions to express doubt or disagreement with different

degrees of intensity, *e.g.*, with surprise, criticism expressions, and anger. This confirms previous evidence that developers express emotions during daily programming tasks [22].

Overall, our findings suggest that questions in code reviews do not seek exclusively to obtain information, *i.e.*, developers use questions to serve a wide variety of communication goals. In the following we attempt to suggest avenues for possible implications for practitioners as well as further research.

A. Implications for Practitioners

In line with the recommendations for good practices provided by Efstathiou and Spinellis [4], we advocate that tools should support the automatic analysis of fine-grained communicative intentions in code review comments. For example, understanding the information needs of reviewers and the way authors and reviewers interact during code review can provide empirically-driven guidelines for change authors to anticipate the reviewers information needs, thus making the code review process faster and more effective. Such practices can be complemented by the use of automatic tools providing feedback for improvement during code review. For example, we envision tools for augmented writing of comments to support neurodiverse developers to achieve an efficient and explicit communication during code review.

Furthermore, fine-grained analysis of the argumentation structure of code review discussions can be leveraged to identify long-lasting value knowledge to be shared among the contributors of a project. For example, the correct identification of requests for rationale may help identify design-related discussion, as proposed by Viviani *et al.* [1].

This paper also provides further evidence of expressions of negative emotions in collaborative development [7], [23]. Early detection of negative emotions might benefit community management and reduce undesired turnover, *i.e.*, by preventing burnout and loss of productivity [24] or timely addressing code of conduct violations and community smells [25].

Finally, real-time identification of the communicative goal of questions in code review could enhance the effectiveness of the code review process itself. Requests for clarification or rationale could be detected in order to support programmers experiencing confusion, *i.e.*, by soliciting the author of the change to provide the required information [17] or by triggering an expert intervention. Conversely, requests for action or suggestions might be leveraged to notify the colleagues, *i.e.*, by notifying them that a change in the code is recommended.

B. Implication for Researchers

As befitting the exploratory case study [6], we propose three hypotheses about the intention of developers' questions.

H1 (cf. Section IV-1). *Questions from the inline comments are frequently used to trigger an action of the interlocutor (rather than to satisfy information needs).* Suggestions and requests for action jointly form the second most frequent communication goal: elicit an action of peers. We would like to investigate whether this same trend occurs in the general comments. Furthermore, this evidence indicates the intention of adopting a polite style of communication by performing requests to perform code changes through indirect questions, rather than direct imperative statements. It would be interesting to enhance the evidence provided by related research about politeness and productivity in bug fixing [26] by investigating this relationship also in code review.

H2 (cf. Section IV-2). *While the largest group of questions from the code reviews aim at satisfying information needs, such as request for information, rationale, or clarification, this group constitutes less than half of all questions.* One could further investigate to what extent the various types of requests are related to the presence of confusion in code reviews [17].

H3 (cf. Section IV-3). *Developers use questions to express their own cognitive and emotional state in order to induce critical reflection in the interlocutor.* They express criticism and emotions (*e.g.*, anger or surprise), and convey doubt, perplexity and disagreement with different degrees of strength.

Beyond the specific hypotheses, a replication of this study with a broader set of questions, general comments and, possibly, with an automatic approach to speech act analysis, will bring full understanding of the role played by questions in code reviews. We believe that such understanding will shed new light on the informational needs of developers, the antecedents to confusion they might experience, and the way they communicate with each other to make suggestions or requests. Furthermore, questions expressing different communicative intentions might have different impact on software quality, development time or project evolution: *e.g.*, hypothetical scenarios might be successful in revealing hidden bugs. A yet another perspective would be to combine the study of communicative intentions with social aspects of developers' communities, *e.g.*, who is asking questions with different communicative intentions and who is being asked. One might also wonder whether the observation of Gachechiladze *et al.* that anger in Jira issues is usually directed at objects rather than the speaker themselves or the interlocutors [7] is also valid for the code reviews. Finally, we would like to verify relation between directionality of emotions and the outcome of code reviews (merge or abandon).

C. Threats to Validity

As for construct validity, we used a state-of-the-art NLP tool, *i.e.*, the StanfordNLP API [18], to extract questions included in our sample set. During the manual annotation, we observed that two sentences were erroneously classified as questions (0.4%) by the tool. Thus, we still consider its results precise enough for our exploratory study. As for internal validity, manual annotation is an error-prone activity whose output depends on the subjective evaluation of the raters. To mitigate such threats, the raters were recruited based on their background in computer science and expertise with labeling. We solved disagreements either through majority voting or discussions, to address threats due to subjectivity in annotation. Finally, due to the exploratory character of our study we do not claim generalizability of our observations but use them to derive hypotheses (Section V-B) for future studies.

VI. Related Work

The communicative intentions of questions have been the object of dedicated studies of ordinary conversations [27], talk shows [28], and community-based Question & Answer (Q&A) sites [29]. To the best of our knowledge, this is the first study of communicative intention within the context of code reviews.

Thanks to the popularity of Stack Overflow, which has made available an enormous number of natural language interactions, researchers in software engineering also started to investigate how developers formulate their requests in information seeking tasks. Treude *et al.* [30] analyzed which kinds of questions are asked on Stack Overflow. Bajaj *et al.* [31] investigated the questions and answers from Stack Overflow to understand the challenges and misconceptions among web developers. Calefato *et al.* [8] studied how information seekers can increase the chance of eliciting a successful answer to their questions. While relevant for the software development domain, the aforementioned studies focused either on the topics or the writing style of the questions from Q&A websites, where the intention of questions is assumed to be the same for all Stack Overflow posts, *i.e.*, asking for technical help [30].

Other studies aimed at categorizing the type of information need expressed by developers' questions. Fritz and Murphy [32] conducted interviews with eleven developers and identified 78 questions they ask in their daily jobs, and among the information needs identified, there are code change specific, people specific, and work-item progress. Sillito *et al.* [33] conducted a study with developers performing code changes to understand what information they need to know about a code base and how they find it.

VII. Conclusions

In this paper, we conducted a case study to investigate the communicative intention of developers' questions during the code review process. We manually labeled 499 questions from 399 Android code review comments. We found that, while representing the majority of requests, information seeking questions are still less than half of all questions in our sample. This evidence suggests that questions are actually used by developers in code review to serve a wider variety of communicative purposes. Specifically, we found that questions are extensively used by developers to convey suggestions. Developers also express their cognitive and affective states in code review comments, such as attitude of doubts and criticisms or emotions like anger and surprise. Based on this case study, we formulate three hypotheses about the intention of developers' questions in the code review process.

VIII. Acknowledgment

This research was supported by CNPq/Brazil (304220/2017-5), CAPES/Brazil (PDSE 88881.132399/2016-01), FACEPE/Brazil (APQ 0839-1.03/14, APQ 0388-1.03/14, 0791-1.03/13), and by the project "EmoQuest", funded by MIUR under the SIR program.

References

[1] G. Viviani, C. Janik-Jones, M. Famelis, and G. Murphy, "The structure of software design discussions," to appear in CHASE, 2018.
[2] A. Bacchelli and C. Bird, "Expectations, outcomes, and challenges of modern code review," in *ICSE*, 2013, pp. 712–721.
[3] A. Bosu, M. Greiler, and C. Bird, "Characteristics of useful code reviews: An empirical study at microsoft," in *MSR*, 2015, pp. 146–156.
[4] V. Efstathiou and D. Spinellis, "Code review comments: language matters," in *ICSE-NIER*, 2018, pp. 69–72.
[5] G. Viviani, C. Janik-Jones, M. Famelis, X. Xia, and G. C. Murphy, "What design topics do developers discuss?" in *ICPC*, 2018.
[6] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *ESE*, pp. 131–164, 2009.
[7] D. Gachechiladze, F. Lanubile, N. Novielli, and A. Serebrenik, "Anger and its direction in collaborative software development," in *ICSE-NIER*, 2017, pp. 11–14.
[8] F. Calefato, F. Lanubile, and N. Novielli, "How to ask for technical help? evidence-based guidelines for writing questions on stack overflow," *IST*, vol. 94, pp. 186–207, 2018.
[9] L. Albright, A. I. Cohen, T. E. Malloy, T. Christ, and G. Bromgard, "Judgments of communicative intent in conversation," *Journal of Experimental Social Psychology*, vol. 40, no. 3, pp. 290 – 302, 2004.
[10] J. L. Austin, *How to do things with words*, ser. William James Lectures. Oxford University Press, 1962.
[11] J. R. Searle, *Speech Acts: An Essay in the Philosophy of Language*. Cambridge, London: Cambridge University Press, 1969.
[12] V. Žegarac and B. Clark, "Phatic interpretations and phatic communication," *Journal of Linguistics*, vol. 35, no. 2, pp. 321–346, 1999.
[13] J. Crook, "On covert communication in advertising," *Journal of Pragmatics*, vol. 36, no. 4, pp. 715–738, 2004.
[14] M. Haugh, "On understandings of intention: A response to Wedgwood," *Intercultural Pragmatics*, vol. 9, no. 2, pp. 161–194, 2012.
[15] 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.
[16] M. V. Mäntylä and C. Lassenius, "What types of defects are really discovered in code reviews?" *TSE*, pp. 430–448, 2009.
[17] F. Ebert, F. Castor, N. Novielli, and A. Serebrenik, "Confusion detection in code reviews," in *ICSME*, 2017, pp. 549–553.
[18] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *ACL System Demonstrations*, 2014, pp. 55–60.
[19] A. Strauss and J. Corbin, *Basics of Qualitative Research: Grounded Theory Procedures and Techniques*. Sage Publications, 1990.
[20] J. Fleiss, "Measuring nominal scale agreement among many raters," *Psychological Bulletin*, vol. 76, pp. 378–382, 1971.
[21] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data," *Biometrics*, vol. 33, no. 1, 1977.
[22] 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.
[23] D. Ford and C. Parnin, "Exploring causes of frustration for software developers," in *CHASE*, 2015, pp. 115–116.
[24] M. Mäntylä, B. Adams, G. Destefanis, D. Graziotin, and M. Ortu, "Mining valence, arousal, and dominance: Possibilities for detecting burnout and productivity?" in *MSR*, 2016, pp. 247–258.
[25] P. Tourani, B. Adams, and A. Serebrenik, "Code of conduct in open source projects," in *SANER*, 2017, pp. 24–33.
[26] M. Ortu, G. Destefanis, M. Kassab, S. Counsell, M. Marchesi, and R. Tonelli, "Would you mind fixing this issue?" in *XP*, 2015, pp. 129–140.
[27] T. Stivers and N. J. Enfield, "A coding scheme for question-response sequences in conversation," *Journal of Pragmatics*, pp. 2620–2626, 2010.
[28] C. Ilie, "Question-response argumentation in talk shows," *Journal of Pragmatics*, pp. 975–999, 1999.
[29] L. Chen, D. Zhang, and L. Mark, "Understanding user intent in community question answering," in *WWW*, 2012, pp. 823–828.
[30] C. Treude, O. Barzilay, and M.-A. Storey, "How do programmers ask and answer questions on the web?" in *ICSE-NIER*, 2011, pp. 804–807.
[31] K. Bajaj, K. Pattabiraman, and A. Mesbah, "Mining questions asked by web developers," in *MSR*, 2014, pp. 112–121.
[32] T. Fritz and G. C. Murphy, "Using information fragments to answer the questions developers ask," in *ICSE*, 2010, pp. 175–184.
[33] J. Sillito, G. C. Murphy, and K. De Volder, "Questions programmers ask during software evolution tasks," in *FSE*, 2006, pp. 23–34.