Speculative Analysis for Quality Assessment of Code Comments

Pooja Rani
Software Composition Group, University of Bern
Bern, Switzerland
🌐 scg.unibe.ch/staff

Abstract—Previous studies have shown that high-quality code comments assist developers in program comprehension and maintenance tasks. However, the semi-structured nature of comments, unclear conventions for writing good comments, and the lack of quality assessment tools for all aspects of comments make their evaluation and maintenance a non-trivial problem. To achieve high-quality comments, we need a deeper understanding of code comment characteristics and the practices developers follow. In this thesis, we approach the problem of assessing comment quality from three different perspectives: what developers ask about commenting practices, what they write in comments, and how researchers support them in assessing comment quality.

Our preliminary findings show that developers embed various kinds of information in class comments across programming languages. Still, they face problems in locating relevant guidelines to write consistent and informative comments, verifying the adherence of their comments to the guidelines, and evaluating the overall state of comment quality. To help developers and researchers in building comment quality assessment tools, we provide: (i) an empirically validated taxonomy of comment convention-related questions from various community forums, (ii) an empirically validated taxonomy of comment information types from various programming languages, (iii) a language-independent approach to automatically identify the information types, and (iv) a comment quality taxonomy prepared from a systematic literature review.

Index Terms—code comments, mining developer sources, developer information needs, comment quality assessment

I. INTRODUCTION

Well-documented code facilitates various software development and maintenance activities [1], [2]. Several studies show that high quality code comments help developers in program comprehension [3], suitable API selection [4], and bug detection [5]. However, comments are written using natural language sentences and their syntax and semantics are neither enforced by a programming language nor checked by the compiler. As a result, developers are free to use numerous means and conventions to write comments [6], and embed various types of information in them [7], thus making the quality evaluation of comments more complicated.

To guide developers in writing consistent and informative comments, programming language communities such as those for Java and Python, and large organizations such as Google and Oracle provide coding style guidelines. However, these guidelines only marginally cover aspects of commenting code such as content, style, and syntax. Furthermore, the availability of several guidelines for a language makes developers unsure about which comment conventions to use, which syntax to follow, and which type of information to write for what kinds of comments. Therefore, developers ask questions on mailing lists, and community platforms such as Stack Overflow (SO) and Quora to address these issues [8], [9]. Analyzing such developer concerns is valuable to understand their needs, and to identify challenges related to commenting practices. Similarly analyzing their actual commenting practices is essential to understand the information they embed in comments and to ensure the quality of that information.

Previous studies have characterized developer commenting practices in OOP languages by classifying comments based on the information that comments contain [7], [10]–[12]. Given the variety of comment types (class, method, or inline), not all comment types describe the source code at same levels of abstraction, therefore, the quality assessment tools need to be tailored accordingly. For example, class comments in Java should present high-level information about a class, whereas method comments should present implementation-level details [13]. These commenting conventions vary across programming languages. For instance, in comparison to Java, class comments in Smalltalk are expected to contain high-level design details and low-level implementation details. Given the increasing usage of multi-language software systems [14] and persistent concerns about maintaining high documentation quality, it is critical to understand what developers write in a particular comment type, and to build tools to extract and check the embedded information across languages. This can also help to ensure the extent to which a comment type adheres to a coding style guideline from the content aspect.

Even when a comment adheres to its coding style guidelines from all aspects such as content, syntax, and style, it is still possible that the comment is incomplete or inconsistent with the code, and thus lacks the desired high quality. Therefore, several other quality attributes that can affect comment quality need to be considered in the overall assessment of comments. Researchers have proposed numerous comment quality evaluation models based on a number of metrics [6], [15] and classification approaches [11]. However, a unifying comment quality taxonomy to express the purposes for which researchers evaluate comments, and which quality attributes they consider important and integrate frequently in their comment quality models or tools is still missing.

In summary, a good understanding of the existing practices

that developers follow, and of comment quality models researchers suggest is necessary to bridge the gap between the notion of quality and its concrete implementation. To gain this required understanding, we analyze code comments from various perspectives, of developers in terms of what they ask and what they write in comments, and of researchers in terms of what they suggest. In this exploration of semantics embedded in the comments, we use *speculative analysis*, by analogy with speculative execution (*e.g.*, branch prediction). Previous studies have also shown how speculative analysis can be used to develop tools that inform developers early and precisely about potential consequences of their actions [16], [17]. In our case, we are interested in supporting developers to ensure comment quality while writing or using comments for various development tasks.

The goal of this thesis is to investigate practices in code comment writing and evaluation in a stepwise manner to ultimately improve comment quality assessment techniques. I state my thesis as follows:

> *Understanding the specfication of high-quality comments to build effective assessement tools requires a multi-perspective view of the comments. The view can be approached by analyzing (1) developer concerns about comments, (2) their commenting practices within IDEs, and (3) required quality attributes for their comments.*

This dissertation will focus on three main questions:

- *What do developers ask about commenting practices?* Answering this can help in (i) identifying the key challenges developers face with current conventions and tools, and (ii) adapting their approaches accordingly.
- *What information do developers write in comments?* Understanding this can support the development of tools and approaches (i) to identify important information types and comment clones automatically, and (ii) to verify the adherence of comments to style guidelines from the content aspect.
- *How do researchers support comment quality assessment?* Answering this question can help in (i) identifying the limitation of the existing tools and techniques, and (ii) adapting the tools according to the proposed comment quality taxonomy.

II. THESIS VISION

This section presents the studies conducted to answer the main questions, as shown in Figure 1. The following subsections briefly describe the motivation, methodology, and preliminary findings of each study.

A. What do developers ask about commenting practices?

Previous studies have leveraged various online platforms to gain a deep understanding of developers needs and challenges [8], [9], [18]. We investigated the popular Q&A forums, SO and Quora, and Apache project-specific mailing lists to understand their commenting practices.

Methodology. To answer the question, we mined and preprocessed 11 931 posts extracted using 14 relevant tags on SO. The tags are selected using a hybrid approach combining
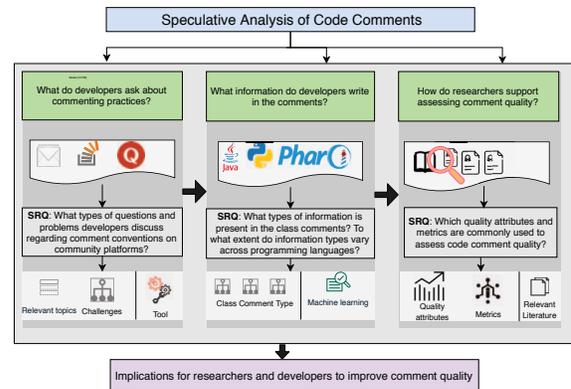

Fig. 1. Overview of my dissertation with all research questions, their methodology, and results.

a heuristics-based approach used by Yang *et al.* [9] and a keyword-based approach used by Aghajani *et al.* successfully in their work [18]. We used a semi-automated approach based on Latent Dirichlet Allocation (LDA) [19], an advanced and popular topic modeling technique, to identify topics from the selected posts. To uncover developer concerns in detail (mainly which type of questions they ask and about what tools and techniques), we manually analyzed a statistically significant sample set of posts from SO, Quora and mailing lists, and formulated a taxonomy of these concerns. The taxonomy offers an overview of the leading questions discussing commenting conventions in a more formal, structured, and possibly exhaustive way.

Findings. Our study results highlight that: (i) Developers ask questions about best practices to write comments (15% of the questions) and generate comments automatically using various tools and technologies. (ii) Among 14 topics identified by LDA, we found five irrelevant topics due to the generality and commonality of the tags (*e.g.*, "convention,", "commenting"). (iii) From our manual analysis, we found that developers are interested in embedding various kinds of information, such as code examples and media (*e.g.*, images) in their code comments but lack clear guidelines to write them. (iv) Developers post questions about documentation tools on SO, whereas no such questions are reported on Quora. In mailing lists, we did not find enough developer discussions about comment conventions.

Conclusion. This analysis shows that developers use various community platforms to raise concerns about code comments. Such concerns hint at the challenges developers face, and their needs from the programming language communities, tools, technologies, and researchers. Conveying clear guidelines to write good comments, and building tools to verify the adherence of comments to these guidelines indicate possible directions to support developers.

B. What information do developers write in comments?

Source code comments consist of several comment types (class comments, method comments, inline comments), but not all comment types contain the same types of information. We

start our analysis by first focusing on class comments, which play an important role in obtaining a high-level overview of classes in object-oriented programming languages [20]. Class commenting practices however vary across programming languages. For instance, a class comment in Java or Python contains high-level overview details and uses annotations (*e.g.*, @param,@return) to express specific types of information. In contrast, class comments in Smalltalk contain detailed design and implementation documentation, and they do not make use of any annotation. We first investigated class comments in Pharo (a modern Smalltalk environment), and identified the types of information developers embed in them by studying the research question RQ_1: *What types of information are present in Pharo class comments?* Then we measured the adherence of Pharo class comments to the class comment guidelines. To generalize our findings across languages, we extended our analysis to other programming languages, namely Java and Python. We systematically compared the commonalities and differences among class commenting practices with the research question RQ_2: *To what extent do information types vary across programming languages?* In order to automate the identification of information types from class comments across languages, we studied the research question RQ_3: *Can machine learning be used to identify class comment types according to our taxonomy automatically?*

Methodology. To answer RQ_1, we conducted a three-iteration-based analysis on a statistically significant sample set of 714 comments selected from internal and external projects of Pharo. Three authors analyzed the content of comments using open-card sorting and pair sorting to build and validate the comment taxonomy. In the case of Python and Java class comments, we used the initial comment taxonomy available from previous works [7], [12], and analyzed and validated the content using the closed-card sorting technique. Based on the constructed taxonomy *i.e.*, Class Comment Type Model (CCTM) and labelled data from each language, we answered RQ_2. To automatically classify class comment types according to CCTM for RQ_3, we used an approach that leverages two techniques namely Natural Language Processing (NLP) and TF-IDF. We use the TF-IDF technique as a baseline due to its successful adoption in recent work on classifying code comments [21]. We transform a multi-label classification into a set of single-label classification problems to balance one label at a time and avoid over-fitting the categories. We adopt a 10-fold cross-validation strategy with a standard probabilistic Naive Bayes classifier, the J48 tree model, and the Random Forest model based on the recent work [7]. We evaluate RQ_3 by measuring precision, recall, and F-measure of our approach against the TF-IDF baseline approach.

Findings. Our results highlight that: (i) Developers express different kinds of information (more than 15 information types) in class comments ranging from the high-level overview of the class to low-level implementation details across programming languages. (ii) Class comments contain various types of information but not all of these information types are suggested by coding guidelines, and this behaviour is observed across all the selected languages. In the case of Pharo, the information types suggested by the guidelines were observed more frequently than other information types. We are in still in the process of verifying this observation in other programming languages. (iii) The Random Forest algorithm fed by the combination of NLP+TF-IDF features achieves the best classification performance for the top six frequent categories over the investigated languages with relatively high precision (ranging from 78% to 92% for the selected languages), recall (ranging from 86% to 92%), and F-Measure (ranging from 77% to 92%) where Pharo achieves less stable results compared to Python and Java.

Conclusion. This analysis highlights the diverse types of information developers embed in class comments regardless of coding style guidelines about comments. Given the benefits of retrieving these information types automatically for various development tasks, it highlights the challenges in unifying retrieval approaches across languages.

C. How do researchers support comment quality assessment?

Software quality is frequently represented as a contextual concept. Therefore, it requires identification and quantification of important characteristics of high-quality software as a first step to measure it [22]. The main objective of our literature review is: to identify the quality attributes that are used to assess code comment quality and collect the metrics used to measure these quality attributes. Additionally, we are interested in which tools/models have been proposed by researchers to assess comment quality. To achieve these objectives, we plan to conduct a systematic literature review (SLR) to answer the following research questions. RQ_1: *What quality attributes are used to evaluate the quality of code comments, and what metrics are used to estimate the quality attributes?* RQ_2: *Which quality attributes and metrics do current assessment tools support?*

Methodology: We plan to conduct the SLR following the guidelines of Kitchenham [23]. We separate the study steps associated with the SLR-related phases planning, conducting the review, and reporting. In the planning phase, we identify the objectives of our SLR and specify the research questions. We plan to review the proceedings of the past ten years *i.e.*, 2010-2020 from the relevant SE conferences and journals according to the *Computing Research and Education Association of Australasia* (CORE) ranking.[1] We formulate the inclusion and exclusion criteria. Based on these criteria, we plan to systematically identify relevant studies.

Expected output. Insights from the SLR are expected to provide a detailed view of the tools and techniques proposed by researchers to assess the quality of comments. Based on these insights, we plan to prepare a comment quality taxonomy which can help researchers and developers in identifying various quality attributes suitable for a comment type and integrating the relevant measures in their tools as per their requirements.

---

[1]*CORE rankings portal*, accessed August 18, 2020, http://www.core.edu.au/

301

III. Preliminary and Expected Contributions

From each study, we present empirical insights, approaches, and tools to support developers and researchers in ensuring high-quality comments.

- For the first question *"What do developers ask about commenting practices?"*, we present: (i) an empirically validated taxonomy of comment convention-related questions from various community forums, and (ii) a tool to conduct a mining study on multiple sources or forums,
- For the second question *"What information do developers write in comments?"*, we provide: (i) an overview of the Pharo class commenting trends over seven major releases till 2019, (ii) an empirically validated taxonomy, called CCTM, characterizing the information types found in class comments written by developers in three different programming languages, and (iii) an automated approach (available for research purposes) able to accurately classify class comments according to CCTM
- For the third question *"How do researchers support comment quality assessment?"*, we expect to achieve: (i) a comment quality taxonomy to identify relevant quality attributes, and (ii) a review of existing tools and techniques that assess the quality of code comments.

IV. Proposed Timeline

I am a third-year PhD student and will be entering the final year of my PhD from January 2021. The expected timeline for the projects:

- A first study (II-A) has been submitted to the journal-first track at Transactions on Software Engineering and Methodology, 2020 (TOSEM'20).
- The RQ1 in the second study (II-B) is currently undergoing a minor revision in the journal-first track at Empirical Software Engineering (EMSE'19) [24].
- Other research questions (RQ2 and RQ3) in the second study (II-B) have been submitted to the Journal of Systems and Software (JSS'20).
- The third study (II-C) is currently planned to be submitted to Transactions on Software Engineering (TSE'21).

V. Related Work

Comment conventions (RQ1): Developers frequently use various web resources to satisfy their information needs. Recently, researchers have started leveraging these resources such as version control systems [25], Q&A forums [8], [9], and mailing lists [18]. In the context of software documentation, Aghajani *et al.* studied documentation issues on SO, Github and mailing lists [18] and formulated a taxonomy of these issues. However, they have focused on the issues related to project documentation, such as wikis, user manuals, and code documentation, and do not focus specifically on the issues of the convention of the code comments. Barua *et al.* found questions concerning coding style and practice to be amongst those most frequently appearing on SO [8], but did not investigate it further. Our first question (II-A) focuses specifically on the problems related to commenting practices developers discuss on SO, Quora, and mailing lists.

Identify information types from comments (RQ2): Code comments contain valuable information to help developers in various activities and tasks. Pascarella *et al.* identified the information types from Java code comments and presented a taxonomy [7]. Similarly, Zhang *et al.* identified information types from Python code comments [12]. We focused specifically on class commenting practices and used their taxonomy as an initial taxonomy to classify class comments. Compared to the work of Pascarella *et al.* and Zhang *et al.* [7], [12], we found several other types of information such as warnings, observations, and recommendations developers embed in class comments. To identify different kinds of information from comments automatically, several studies have explored numerous approaches based on heuristics or textual features [26], [27]. In contrast to these previous approaches, we extracted the natural language patterns (heuristics) automatically using a tool, combined them with other textual features, and tested our approach across languages.

Comments quality (RQ3): Apart from identifying information embedded in the comments, assessing comments from other perspectives has gained a lot of attention from researchers in the past years, for example, assessing comment quality [11], [15], detecting inconsistency between code and comments [28], [29], and examining co-evolution of code and comments [30]. The main aim is to keep comments consistent with the code and to maintain their high quality. Several recent works have proposed tools and techniques to automatically assess the comments using specific quality attributes and metrics [11], [15], [31]. However, a unifying model of comment quality attributes and metrics that are considered important for assessing comments is still missing. Previous literature reviews have provided the quality models for the software documentation [32], [33] but we focus specifically on the code comment aspect.

VI. Conclusion

To improve the state of comment quality assessment techniques, this thesis focuses on three main questions: what do developers ask about commenting practices, what do they write in comments, and how do researchers support assessment of comments. Our work draws insights from both empirical evidence mined from developer sources and research results (SLR). Our preliminary findings show that developers embed various kinds of information in comments. Still, they face several problems in locating the specific comment guidelines, verifying the adherence of their comments to the coding standards, and evaluating the overall state of the comment quality. Our empirical evidence also shows that Pharo developers follow commenting guidelines in writing class comments. These insights of developer commenting practices across languages can help researchers to improve comment quality assessment tools, and to evaluate comment summarization and comment generation approaches. We present initial approaches, tools,

and labelled dataset to facilitate the future comment analysis work on other languages and environment.

My future work will concentrate on exploring which tasks and activities require developers to search these information types in comments and how developers find these information types. Based on developer commenting practices, my objective would be to improve my prototype tools and reduce the efforts in assessing comment quality. I expect to finish the work for my dissertation in 2021.

REFERENCES

[1] S. C. B. de Souza, N. Anquetil, and K. M. de Oliveira, "A study of the documentation essential to software maintenance," in *Proceedings of the 23rd annual international conference on Design of communication: documenting & designing for pervasive information*, ser. SIGDOC '05. New York, NY, USA: ACM, 2005, pp. 68–75.

[2] F. A. Cioch, M. Palazzolo, and S. Lohrer, "A documentation suite for maintenance programmers," in *Proceedings of the 1996 International Conference on Software Maintenance*, ser. ICSM '96. Washington, DC, USA: IEEE Computer Society, 1996, pp. 286–295. [Online]. Available: http://dl.acm.org/citation.cfm?id=645544.655870

[3] U. Dekel and J. D. Herbsleb, "Reading the documentation of invoked API functions in program comprehension," in *2009 IEEE 17th International Conference on Program Comprehension*. IEEE, 2009, pp. 168–177.

[4] C. McMillan, D. Poshyvanyk, and M. Grechanik, "Recommending source code examples via API call usages and documentation," in *Proceedings of the 2nd International Workshop on Recommendation Systems for Software Engineering*, 2010, pp. 21–25.

[5] L. Tan, D. Yuan, G. Krishna, and Y. Zhou, "/* iComment: Bugs or bad comments?*/," in *Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, 2007, pp. 145–158.

[6] Y. Padioleau, L. Tan, and Y. Zhou, "Listening to programmers — taxonomies and characteristics of comments in operating system code," in *Proceedings of the 31st International Conference on Software Engineering*. IEEE Computer Society, 2009, pp. 331–341.

[7] L. Pascarella and A. Bacchelli, "Classifying code comments in Java open-source software systems," in *Proceedings of the 14th International Conference on Mining Software Repositories*, ser. MSR '17. IEEE Press, 2017, pp. 227–237. [Online]. Available: https://doi.org/10.1109/MSR.2017.63

[8] A. Barua, S. W. Thomas, and A. E. Hassan, "What are developers talking about? An analysis of topics and trends in Stack Overflow," *Empirical Software Engineering*, vol. 19, no. 3, pp. 619–654, 2014. [Online]. Available: https://doi.org/10.1007/s10664-012-9231-y

[9] X.-L. Yang, D. Lo, X. Xia, Z.-Y. Wan, and J.-L. Sun, "What security questions do developers ask? A large-scale study of Stack Overflow posts," *Journal of Computer Science and Technology*, vol. 31, no. 5, pp. 910–924, 2016. [Online]. Available: https://doi.org/10.1007/s11390-016-1672-0

[10] D. Haouari, H. Sahraoui, and P. Langlais, "How good is your comment? A study of comments in Java programs," in *2011 International Symposium on Empirical Software Engineering and Measurement*. IEEE, 2011, pp. 137–146.

[11] D. Steidl, B. Hummel, and E. Juergens, "Quality analysis of source code comments," in *Program Comprehension (ICPC), 2013 IEEE 21st International Conference on*. IEEE, 2013, pp. 83–92.

[12] J. Zhang, L. Xu, and Y. Li, "Classifying Python code comments based on supervised learning," in *International Conference on Web Information Systems and Applications*. Springer, 2018, pp. 39–47.

[13] E. Nurvitadhi, W. W. Leung, and C. Cook, "Do class comments aid Java program understanding?" in *33rd Annual Frontiers in Education, 2003. FIE 2003.*, vol. 1. IEEE, 2003, pp. T3C–T3C.

[14] F. Tomassetti and M. Torchiano, "An empirical assessment of polyglotism in GitHub," in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, 2014, pp. 1–4.

[15] N. Khamis, R. Witte, and J. Rilling, "Automatic quality assessment of source code comments: the JavadocMiner," in *International Conference on Application of Natural Language to Information Systems*. Springer, 2010, pp. 68–79.

[16] Y. Brun, R. Holmes, M. D. Ernst, and D. Notkin, "Speculative analysis: Exploring future development states of software," in *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research*, ser. FoSER '10. New York, NY, USA: ACM, 2010, pp. 59–64. [Online]. Available: http://doi.acm.org/10.1145/1882362.1882375

[17] K. Muşlu, Y. Brun, M. D. Ernst, and D. Notkin, "Making offline analyses continuous," in *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, 2013, pp. 323–333.

[18] E. Aghajani, C. Nagy, O. L. Vega-Márquez, M. Linares-Vásquez, L. Moreno, G. Bavota, and M. Lanza, "Software documentation issues unveiled," in *Proceedings of the 41st International Conference on Software Engineering, ICSE 2019, Montreal, QC, Canada, May 25-31, 2019*, J. M. Atlee, T. Bultan, and J. Whittle, Eds. IEEE / ACM, 2019, pp. 1199–1210. [Online]. Available: https://doi.org/10.1109/ICSE.2019.00122

[19] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet Allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[20] A. Cline, "Testing thread," in *Agile Development in the Real World*. Springer, 2015, pp. 221–252.

[21] V. Misra, J. S. K. Reddy, and S. Chimalakonda, "Is there a correlation between code comments and issues?: an exploratory study," in *SAC '20: The 35th ACM/SIGAPP Symposium on Applied Computing, online event, [Brno, Czech Republic], March 30 - April 3, 2020*, 2020, pp. 110–117. [Online]. Available: https://doi.org/10.1145/3341105.3374009

[22] J. Bansiya and C. Davis, "A hierarchical model for object-oriented design quality assessment," *IEEE Transactions on Software Engineering*, vol. 28, no. 1, pp. 4–17, Jan. 2002.

[23] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," 2007.

[24] P. Rani, S. Panichella, M. Leuenberger, M. Ghafari, and O. Nierstrasz, "What do class comments tell us? An investigation of comment evolution and practices in Pharo," *arXiv preprint arXiv:2005.11583*, 2020, to appear in Empirical Software Engineering.

[25] T.-H. Chen, S. W. Thomas, and A. E. Hassan, "A survey on the use of topic models when mining software repositories," *Empirical Softw. Engg.*, vol. 21, no. 5, pp. 1843–1919, oct 2016. [Online]. Available: https://doi.org/10.1007/s10664-015-9402-8

[26] N. Dragan, M. L. Collard, and J. I. Maletic, "Automatic identification of class stereotypes," in *Proceedings of the 2010 IEEE International Conference on Software Maintenance*, ser. ICSM '10. USA: IEEE Computer Society, 2010, p. 110. [Online]. Available: https://doi.org/10.1109/ICSM.2010.5609703

[27] Y. Shinyama, Y. Arahori, and K. Gondow, "Analyzing code comments to boost program comprehension," in *2018 25th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2018, pp. 325–334.

[28] I. K. Ratol and M. P. Robillard, "Detecting fragile comments," in *Proceedings of the 32Nd IEEE/ACM International Conference on Automated Software Engineering*. IEEE Press, 2017, pp. 112–122.

[29] F. Wen, C. Nagy, G. Bavota, and M. Lanza, "A large-scale empirical study on code-comment inconsistencies," in *Proceedings of the 27th International Conference on Program Comprehension*. IEEE Press, 2019, pp. 53–64.

[30] B. Fluri, M. Würsch, E. Giger, and H. C. Gall, "Analyzing the co-evolution of comments and source code," *Software Quality Journal*, vol. 17, no. 4, pp. 367–394, 2009.

[31] H. Yu, B. Li, P. Wang, D. Jia, and Y. Wang, "Source code comments quality assessment method based on aggregation of classification algorithms," *J. Comput. Appl.*, vol. 36, no. 12, pp. 3448–3453, 2016.

[32] W. Ding, P. Liang, A. Tang, and H. Van Vliet, "Knowledge-based approaches in software documentation: A systematic literature review," *Information and Software Technology*, vol. 56, no. 6, pp. 545–567, 2014.

[33] J. Zhi, V. Garousi-Yusifoğlu, B. Sun, G. Garousi, S. Shahnewaz, and G. Ruhe, "Cost, benefits and quality of software development documentation: A systematic mapping," *Journal of Systems and Software*, vol. 99, pp. 175–198, 2015.