Do Comments follow Commenting Conventions? A Case Study in Java and Python

Pooja Rani*, Suada Abukar*, Nataliia Stulova *, Alexander Bergel†, Oscar Nierstrasz*

*Software Composition Group, University of Bern, Bern, Switzerland
†Department of Computer Science, University of Chile, Santiago, Chile
scg.unibe.ch/staff

***Abstract*—Assessing code comment quality is known to be a difficult problem. A number of coding style guidelines have been created with the aim to encourage writing of informative, readable, and consistent comments. However, it is not clear from the research to date which specific aspects of comments the guidelines cover (*e.g.*, syntax, content, structure). Furthermore, the extent to which developers follow these guidelines while writing code comments is unknown.**

We analyze various style guidelines in Java and Python and uncover that the majority of them address more the content aspect of the comments rather than syntax or formatting, but when considering the different types of information developers embed in comments and the concerns they raise on various online platforms about the commenting practices, existing comment conventions are not yet specified clearly enough, nor do they adequately cover important concerns. Our results highlight the mismatch between developer commenting practices and style guidelines, and provide several focal points for the design and improvement of comment quality checking tools.

***Index Terms*—Comment analysis, Software documentation, Coding Style Guidelines**

I. Introduction

Developers use several kinds of software documentation, including design documents, wikis, and code comments, to understand and maintain programs. Studies show that developers trust code comments more than other forms of documentation [1]. As code comments are usually written in a semi-structured manner using natural language sentences, and they are not checked by the compiler, developers have the freedom to write comments in various ways ([2], [3]).

To encourage developers to write consistent, readable, and informative code comments, programming language communities and several large organizations, such as Google and Apache,[1] provide coding style guidelines that also suggest comment-related conventions ([4], [5], [6]). These conventions cover various aspects of comments, such as syntactic, stylistic, or content-related aspects. For instance, *"Use 3rd person (descriptive), not 2nd person (prescriptive)"* is an example of a stylistic comment convention for Java documentation comments.[2] However, to what extent these aspects are covered within different style guidelines and languages is not known. Having an understanding of various commenting conventions can help developers and researchers know what our current style guidelines lack. To gain this understanding, we formulate the research question: ***RQ*$_1$***: Which type of comment conventions are suggested by various style guidelines?*

As high-quality comments support developers in understanding and maintaining their programs, it is essential to ensure the adherence of their comments to the style guidelines to evaluate the overall comment quality. Rani *et al.* have investigated class comments of Smalltalk and their adherence to the commenting conventions provided by a default template [7]. They found that Smalltalk developers follow writing style and content-related comment conventions more than 50% of the time, but they use inconsistent structure and formatting of comment content. As Java and Python are among the most popular languages in use, several research works have focused on studying comments in Java and Python ([8], [9]), some especially focusing on class comments [10]. However, it remains largely unknown whether Java and Python developers adhere to the commenting conventions suggested by the style guidelines or not. To obtain this understanding, we formulate another research question: ***RQ*$_2$***: To what extent do developers follow commenting conventions in writing code comments in Java and Python?*

Our initial results show that the majority of style guidelines propose more content-related conventions than other types of conventions, but compared to the different types of content developers actually embed in comments ([8], [9], [10]), and the concerns they raise on online platforms (*e.g.*, StackOverflow or Quora) regarding comment conventions [11], it is clear that existing conventions are neither adequate, nor precise enough. On the other hand, these style guidelines often include conventions that are not relevant or applicable in many cases, leading developers to ignore them.

When the conventions are applicable, developers often follow the writing style and content conventions (80% of comments), but violate structure conventions in Java and Python class comments (nearly 30% of comments), confirming the previous results for Smalltalk by Rani *et al.* [7]. Although the project-specific guidelines provide very few additional class comment

[1] Code style guide section in https://spark.apache.org/contributing.html
[2] https://www.oracle.com/technical-resources/articles/java/javadoc-tool.html

conventions, these conventions are followed more often compared to the conventions suggested by the standard guidelines both in Java and Python class comments. The data related to RQ_1 and RQ_2 is given in the replication package.[3]

Our results highlight the mismatch between conventions suggested by the style guidelines but not followed by developers in their projects and vice versa. Verifying this mismatch is currently not well-supported by tools or linters. The linters currently support very limited comment conventions [12], [13], for instance, they are often limited to checking the presence or absence of documentation comments, or their formatting with code. Our results indicate the need to conduct extensive studies on (i) which comment-related conventions linters provide, (ii) how well linters cover comment conventions from various style guidelines, and (iii) building tools and techniques to reduce the mismatch between developer commenting practices and style guidelines.

II. Study Design

Data collection. Rani *et al.* have previously investigated the adherence of Smalltalk class comments to the conventions provided by a default template [7]. To verify their results across other languages, we analyze class comments in Java and Python. Rani *et al.* also investigated class comments of diverse Java and Python projects that vary in terms of size, domain, and contributors [10]. We use their dataset of Java and Python comments for our analysis (RQ1 and RQ2). Table I shows the list of these projects in the column *Project* for the selected languages shown in the column *Language*.

*a) Comment conventions in Style Guidelines (**RQ_1**):* The goal of this RQ is to investigate the type of comment conventions (or rules) various style guidelines suggest. The term *comment convention* refers to suggestions or rules about various aspects of comments, such as syntax, formatting, content, or writing style. As a first step, we identify the standard and project-specific guidelines that the selected projects recommend for code documentation. Then we extract the comment-related rules from these guidelines. Table I shows each project in the column *Project* and if it supports project-specific guidelines (✓) or not (×) for comments in the column *Project guideline*. The column *Standard guideline* describes the standard guidelines that the project mentions on their project page dedicated to writing comments.

As comment conventions can be scattered across multiple paragraphs in a style guideline, we scan all sentences and select those that mention any convention about comments. A rule can target various types of comments, such as class, method, package, or inline comments, or part of comments (*e.g.*, summary, parameters). In case a sentence targets multiple comment types, we split the rule for each type. In total, we collected 600 comment-related rules. We organized all rules into a taxonomy of five main categories: *Content*, *Structure*, *Formatting*, *Syntax*, and *Writing Style*. If a rule does not fit any of these categories, we put it into the *Other* category. The rationale behind the taxonomy is that the approaches evaluating comment quality can focus on a specific aspect of comments they want to evaluate and improve. Categories such as *Content* and *Writing Style*, are considered important by Rani *et al.* in their work on evaluating Smalltalk comments [7].

[3] https://doi.org/10.5281/zenodo.5153663

TABLE I
Overview of the selected projects and their style guidelines.

Language	Project	Project guideline	Standard guideline
Java	Eclipse.cdt	✓	Oracle
	Hadoop	✓	Oracle
	Vaadin	✓	Oracle
	Spark	✓	Oracle
	Guava	×	Google
	Guice	×	Google
Python	Django	✓	PEP8/257
	Requests	✓	PEP8/257
	Pipenv	×	PEP8/257
	Mailpile	×	PEP8/257
	Pandas	✓	Numpy
	iPython	✓	PEP8/257, Numpy
	Pytorch	✓	Google

According to Rani *et al.*, the *Content* category contains the rules that describe which type of information the comment should contain while the *Writing Style* category contains natural-language specific rules, such as grammar, punctuations, and capitalization. We added three more categories, following their methodology, to cover other aspects of comments. The *Formatting* category deals with the rules related to indentation, blank lines, or spacing. It often complements *Structure* category conventions. The category *Structure* contains the rules about organizing the text, or location of the information in comments. For example, how the tags/sections/information should be ordered in the comments. The *Syntax* category focuses on the syntax to write a specific type of comments, for instance, which symbol to use to denote comments. Once each rule is assigned to a category, we analyze the frequency of various types of rules in the style guidelines to answer the RQ_1.

*b) Adherence of comments to conventions (**RQ_2**):* The goal of this RQ is to verify whether or not developers follow the comment conventions, *i.e.*, the rules identified in the previous RQ, in practice in their projects. Currently, there are no tools available that automatically check comments against all types of rules, therefore we manually validate a sample of class comments against the rules applicable to class comments (270 rules out of 600 rules). For the scope of this work, we first focus on the rule types that require manual validation due to limited tool support *i.e.*, all types of rules other than *Formatting*.

We use the dataset by Rani *et al.* that provides sample class comments of the selected projects from Java and Python [10].

They selected a statistically significant sample of class comments from each project with 95% confidence and 5% margin of error, resulting in a total of 700 class comments for both languages. In case a comment follows a particular rule, we label the rule as *followed*, otherwise as *not followed*. There are often cases where a rule is not applicable to the comment due to the unavailability of that information in the comment. For instance, the rules verifying syntax, content, or style of the version information in a class comment cannot be checked if the version information is not mentioned in the comment. For such cases, we label such rules as *not applicable* to the comment. We exclude a few rules for now that cannot be verified with the current dataset due to the abstract nature of a rule, the unavailability of the symbols that denotes the class comment, or code associated with the class, *e.g.*, to verify the Oracle rule *"for the @deprecated tag, suggest what item to use instead"*, class comment alone is not enough and require code of the class to verify the replacement item.

We measure how many comments follow a particular rule and how many do not. One author labels the comments, and the second author reviews the labeled comments. In cases where they do not agree, the third author is consulted, and conflicts are resolved using the majority voting mechanism (Cohens kappa=0.80).

III. Early Results

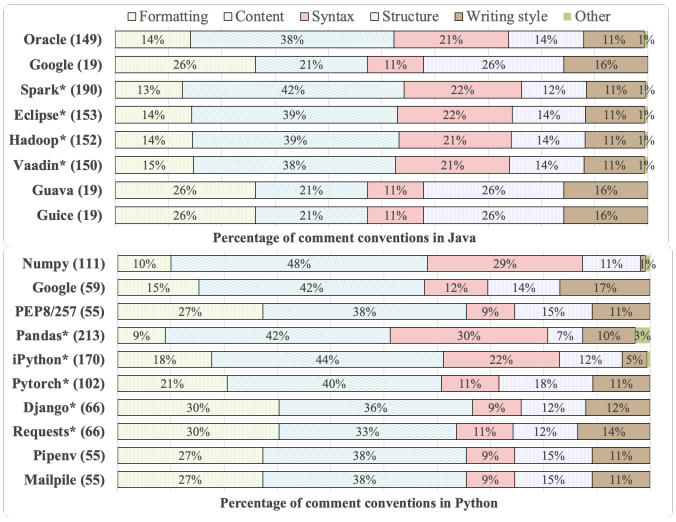

Fig. 1. Types of conventions in Java and Python guidelines

*a) Comment conventions in Style Guidelines (**RQ**$_1$):* Figure 1 shows the total number of conventions for each standard guideline (Oracle and Google) and project-specific guidelines (including conventions from the standard guidelines) on the y-axis. The x-axis indicates the ratio of conventions belonging to a particular category from our taxonomy. Our results show that the majority of style guidelines present more rules about the content to write (*Content*) in comments, except for the Google style guideline in Java, which contains more rules on how to format and structure comments (*Formatting*, *Structure*). Since the Oracle guideline is used as a baseline in several Java projects, project-specific guidelines suggest few additional comment conventions, and these conventions often either conflict with or clarify the standard guidelines. For example, the conventions, such as *line length limit* and *indentation with two spaces, four spaces, or tab*, are often among such additional and conflicting rules across projects. Identifying such rules and ensuring they are configured properly in tools can help developers in following them automatically.

Figure 1 also shows the distribution of rule types for Python style guidelines. Numpy and the projects following it (Pandas and iPython) contain the most rules about what type of information to write in comments and how to write it, compared to other standard guidelines, such as those of Google, PEP, and Oracle. For example, the Numpy guideline suggests writing a short and extended summary of the class, usage examples, notes and warnings in a class comment, and provides syntax and style conventions to write these types of information. Class comments of iPython and Pandas contain all of these information types and follow the syntax conventions to write them. Interestingly, developers write such types of information in all other projects ([9], [10]) regardless of if the project guideline suggest or not, but they are writing these information types in inconsistent ways. Previous comment analysis studies in Java and Python also show that developers embed other, different types of information in comments, such as *Usage*, *Expand*, *Rationale*, or *Pointer*, but we do not find conventions in the corresponding style guidelines (Google, PEP, Oracle) to write such types of information ([8], [9]).

We observe that even though style guidelines are intended to encourage and help developers to write good comments for all code entities, comment conventions are scattered across multiple sources, documents, and paragraphs. Thus, it is not always easy to locate conventions particular to one entity (class, function, inline), causing developers to seek conventions using online sources[11].

Finding. The majority of the style guidelines propose more content-related conventions than other types of conventions, but they are not easy to locate in the style guidelines, and do not always match developer commenting practices.

Finding. The Numpy style guideline provides more rigorous content conventions compared to other style guidelines, such as Oracle, PEP257, or Google.

*b) Comment conventions in Style Guidelines (**RQ**$_2$):* Figure 2 shows the distribution of comments within each project that follows rules, do not follow them, or to which the rules are not applicable. For example, in Eclipse on average 27% of the comments follow the rules, whereas 3% of comments violate the rules, and 70% of the comments do not have enough relevant information within them to check them against a rule. The majority of the class comment rules in Java are not applicable to selected comments (50-70%) whereas in

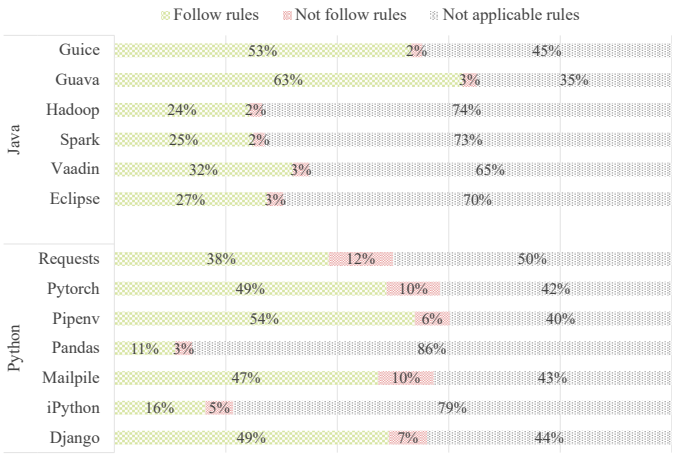

Fig. 2. Percentage of comments that follow rules, do not follow them, or to which rules are not applicable.

Python the ratio is much less. The number of *not applicable* rules shows that the style guidelines suggest various comment conventions, but developers rarely follow them while writing comments. For instance, the Oracle rules in Java, such as *"use FIXME to flag something that is bogus or broken"*, or *"use @serial tag in class comment"* are rarely followed. Similarly, some rules in Python, such as *"Docstrings for a class should list public methods and instance variables"* are also rarely followed.

Finding. Style guidelines suggest various comment conventions, but developers do not or rarely follow them while writing comments.

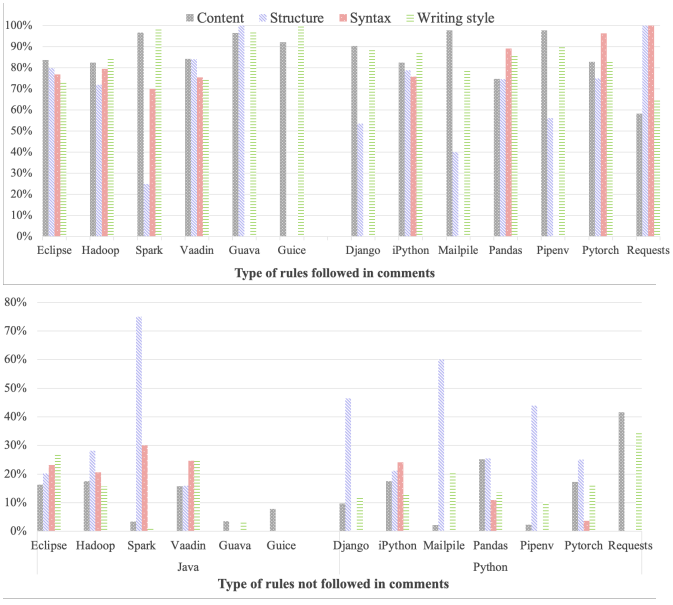

Fig. 3. Types of rules followed or not in Java and Python projects

Of the rules that are applicable as shown in Figure 3, writing style and content rules were more often followed than syntax and structure rules, confirming previous results for Smalltalk [7]. It shows that developers are interested in writing informative comments.

Finding. Compared to Python, Java class comments violate rules less often (as shown in Figure 3).

Finding. Class comments in Java and Python often follow writing style and content conventions (80% of comments), but violate structure conventions (30% of comments).

As discussed before, some rules are often followed, while there are others that are frequently violated. For instance, the syntax rule *"separate the paragraphs with a <p> paragraph tag"* in Spark is violated often. Similarly, in Pandas the rule *"a few sentences giving an extended summary of the class or method after the short (one-line) summary"* is often followed but the rule *"there should be a blank line between the short summary and extended summary"* is often violated. Such conventions can be further investigated by surveying developers to know the specific factors, such as the usage of linters for comments, team strictness, or developer awareness behind these explicit instances of rule adherence or violation. Although the project-specific guidelines provide few additional conventions, these conventions are followed more often than the conventions provided by the standard guidelines. For example, the rule *Do not use @author tags* is specific to Hadoop and in contrast with the Oracle style guideline, but it is always followed in Hadoop comments.

Finding. Project-specific class comment conventions are followed more often than the conventions suggested by the standard guidelines.

IV. Implication & Related Work

Impact of commenting conventions. Coding style guidelines impact program comprehension and maintenance activities. However, not all conventions from the guidelines have the same impact on program comprehension activities. Smit *et al.* [12] ranked 71 code conventions that are most important to maintainable code. However, they accounted only for missing or incomplete Javadoc comments on public types and methods, and did not account for other comment-related conventions, especially about their content.

Similarly, most previous work has focused on building tools for formatting and naming conventions for code entities, while being very limited on comment conventions [2], [14]. We provide a dataset of 700 labeled class comments and 600 conventions (taxonomy) for Java and Python that developers often follow or violate in their comments. It can help researchers rank the specific comment conventions to find out their importance and impact on the program comprehension and maintenance activities.

Comment generation. To reduce developer effort in writing comments, various researchers have proposed to generate

comments automatically. Moreno *et al.* proposed a template-based approach to generate class comments in Java [15]. Given the importance of including developer commenting practices in such approaches, and the impact of a template on developers [7], our results can help researchers design such templates more carefully.

Adherence of comment conventions. Bafatakis *et al.* evaluated the compliance of Python code snippets posted on StackOverflow to Python style guidelines [16]. Simmons *et al.* investigated the extent to which Data Science projects follow code standards in Python [13]. However, covering other important comment conventions and automatically verifying the adherence of comment conventions to the coding standards is not yet fully achieved. In our study, we find various comment conventions, such as grammar rules, the syntax of writing different types of information that developers often follow, but which are not covered in such studies. Rani *et al.* measured the adherence of Smalltalk class comments to the default comment template and found that developers follow the writing conventions of the template [7]. Java and Python do not provide any default template to write comments but support multiple style guidelines by organizations and projects. We studied diverse projects in Java and Python that follow various style guidelines. We found that developers follow writing and content conventions more than other types, thus confirming the results of Rani *et al.* for Smalltalk.

V. Conclusions

Given the importance of code comments and consistency concerns in projects, we study various style guidelines and big open-source projects in the context of comment conventions. We highlight the mismatch between what the style guidelines suggest for comments, and how often developers follow them, and what conventions developers follow in their comments but which are not suggested or mentioned by the style guidelines. Our results indicate the need to conduct extensive studies on various linters or quality assessment tools to know the extent they cover comment conventions. We also provide several focal points for the design and improvement of comment quality assessment tools.

VI. Acknowledgement

We gratefully acknowledge the financial support of the Swiss National Science Foundation for the project “Agile Software Assistance” (SNSF project No. 200020-181973, Feb 1, 2019 - Apr 30, 2022).

References

[1] W. Maalej, R. Tiarks, T. Roehm, and R. Koschke, “On the comprehension of program comprehension,” *ACM TOSEM*, vol. 23, no. 4, pp. 31:1–31:37, Sep. 2014. [Online]. Available: http://mobis.informatik.uni-hamburg.de/wp-content/uploads/2014/06/TOSEM-Maalej-Comprehension-PrePrint2.pdf

[2] M. Allamanis, E. T. Barr, C. Bird, and C. Sutton, “Learning natural coding conventions,” in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ser. FSE 2014. New York, NY, USA: ACM, 2014, pp. 281–293. [Online]. Available: http://doi.acm.org/10.1145/2635868.2635883

[3] 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.

[4] “Oracle documentation guidelines,” 2020. [Online]. Available: https://www.oracle.com/technetwork/java/javase/documentation

[5] W. S. Jr. and E. White, *The Elements of Style*, 4th ed. Allyn and Bacon, 2000.

[6] “Google style guidelines,” 2020, verified on 10 Jan 2021. [Online]. Available: https://google.github.io/styleguide/

[7] 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 Smalltalk,” *arXiv preprint arXiv:2005.11583*, 2020, to be Published in Empirical Software Engineering.

[8] 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

[9] J. Zhang, L. Xu, and Y. Li, “Classifying python code comments based on supervised learning,” in *Web Information Systems and Applications - 15th International Conference, WISA 2018, Taiyuan, China, September 14-15, 2018, Proceedings*, ser. Lecture Notes in Computer Science, X. Meng, R. Li, K. Wang, B. Niu, X. Wang, and G. Zhao, Eds., vol. 11242. Springer, 2018, pp. 39–47. [Online]. Available: https://doi.org/10.1007/978-3-030-02934-0_4

[10] P. Rani, S. Panichella, M. Leuenberger, A. Di Sorbo, and O. Nierstrasz, “How to identify class comment types? A multi-language approach for class comment classification,” *Journal of Systems and Software*, vol. 181, p. 111047, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121221001448

[11] 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

[12] M. Smit, B. Gergel, H. J. Hoover, and E. Stroulia, “Maintainability and source code conventions: An analysis of open source projects,” *University of Alberta, Department of Computing Science, Tech. Rep. TR11*, vol. 6, 2011.

[13] A. J. Simmons, S. Barnett, J. Rivera-Villicana, A. Bajaj, and R. Vasa, “A large-scale comparative analysis of coding standard conformance in open-source data science projects,” in *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2020, pp. 1–11.

[14] M. Arai, “Development and evaluation of eclipse plugin tool for learning programming style of java,” in *2014 9th International Conference on Computer Science & Education*. IEEE, 2014, pp. 495–499.

[15] L. Moreno, J. Aponte, G. Sridhara, A. Marcus, L. L. Pollock, and K. Vijay-Shanker, “Automatic generation of natural language summaries for Java classes,” in *IEEE 21st International Conference on Program Comprehension, ICPC 2013, San Francisco, CA, USA, 20-21 May, 2013*, 2013, pp. 23–32.

[16] N. Bafatakis, N. Boecker, W. Boon, M. C. Salazar, J. Krinke, G. Oznacar, and R. White, “Python coding style compliance on stack overflow,” in *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE, 2019, pp. 210–214.