

Supplemental Material

Enriching API Documentation with Code Samples and Usage Scenarios from Crowd Knowledge

Jingxuan Zhang, He Jiang, Zhilei Ren, Tao Zhang, and Zhiqiu Huang

This document provides the supplement materials for the paper “Enriching API Documentation with Code Samples and Usage Scenarios from Crowd Knowledge”. It mainly consists of four sections. In the first section S1, we compare the programming efficiency of developers when they are assigned with the ADECK-enriched API documentation and when they are allowed to access to the Internet. In the second section S2, we conduct statistical hypothesis testing in RQ3 to verify whether different programming tasks, developer groups, and API documentation are significant different. In the third section S3, we explore what kind of API types can ADECK cover by conducting a small-scale manual annotation. Finally, in the last section S4, we explore the kind of API types that can be covered by ADECK but not by eXoaDocs and vice versa.

■ S1. Comparison against the situation in which developers are allowed to access to the Internet

To investigate whether developers with the ADECK-enriched API documentation are more efficient than that of the developers who are allowed to access to the Internet, we conduct an additional small-scale experiment. Similarly, we recruit 7 master students to participant in the experiment, and we have investigated their programming skills to guarantee that they have similar skills with the developers in the manuscript. The number of newly recruited developers is equal to one developer group in RQ3. In such a way, we can compare different developer groups with two evaluation metrics, i.e., the task completion frequency and the completion time, and further show which developer group is more productive.

Before conducting the experiment, these newly recruited developers are provided with the same experimental guideline, in which we introduce the related concepts and workflow. They are required to fully understand this experimental guideline before starting to complete the programming tasks. Then, the newly recruited developers try to finish the same three programming tasks as the manuscript. When they conduct the programming tasks, they are allowed to access to the Internet. Thus, they can search something in the search engine and Stack Overflow. When they complete one programming task or the programming time reaches to 30 minutes, they continue to do the next following programming task. During they are completing the programming tasks, the screen capture tool records the programming behaviors of developers. After all the developers complete the programming tasks, we analyze the videos to obtain the two evaluation metrics and further compare different developer groups.

Table 1 shows the comparison results of the new developer group accessing to the Internet and the developer groups with the ADECK-enriched API documentation. We can see that the developers accessing to the Internet complete less programming tasks within more average completion time than the developers with the ADECK-enriched API documentation. For example,

the new developer group accessing to the Internet completes 12 programming tasks. In contrast, the other developer groups with the ADECK-enriched API documentation complete no less than 14 programming tasks. From the perspective of the complete time, the new developer group accessing to the Internet complete programming tasks within an average completion time of 22.67 minutes. Whereas, the other groups with the ADECK-enriched API documentation achieve an average completion time of less than 20.88 minutes. From Table1, we can see that the developers with the ADECK-enriched API documentation are more productive.

Table 1. comparison between different developer groups

Different developer groups	The task completion frequency	The completion time
New group accessing to the Internet	12	22.67
Group 1 with the ADECK-enriched API documentation	14	19.25
Group 2 with the ADECK-enriched API documentation	14	20.88
Group 3 with the ADECK-enriched API documentation	15	19.97

After analyzing the programming behaviors of newly recruited developers by inspecting their screen capture videos, we find two reasons that may influence their efficiency. First, there is too much irrelevant information can be retrieved by the search engine and the Stack Overflow search engine, if developers search the programming tasks. In such a way, developers waste a lot of time to scan and find their wanted information. Second, even if they focus on only one webpage or one question in Stack Overflow, they still need to read the content of the retrieved webpage or read all the answers to this question and select the suitable one in Stack Overflow. In addition, they also need to adapt their correct code samples to their programming situations and contexts. Hence, the developers accessing to the Internet may not complete their programming tasks within limited time.

■ S2. Statistical hypothesis testing in RQ3

We conduct the paired Wilcoxon signed rank test to verify whether the programming tasks, the developer groups, and the API documentation are statistically significant in RQ3. Since there are two evaluation metrics, i.e., the task completion frequency and the completion time, we conduct the statistical hypothesis testing between each pair of the programming tasks, the developer groups, and the API documentation from the perspective of the two evaluation metrics separately. Similarly, the p-value is set to 0.05 by default. If the calculated p-value is less than 0.05, a significant difference can be detected. The p-values are presented in Table 2, Table 3, and Table 4.

Table 2 shows the comparison results between different programming tasks. We make the background of a textbox to gray, if its p-value is less than 0.05. We can see that most of the p-values are less than 0.05. That is to say, the programming tasks are statistically different, which is consistent with our design purpose that the programming tasks should have different level of difficulty. For example, there is a statistically significant difference between Task 1 and Task 3 in terms of the task completion frequency and the task completion time, since the p-values are 0.0033 and 0.0004 respectively.

Table 2. Comparison between different programming tasks

	The task completion frequency	The task completion time
Task 1 vs Task 2	0.0369	0.0464
Task 1 vs Task 3	0.0033	0.0004
Task 2 vs Task 3	0.0719	0.0011

Table 3. Comparison between different developer groups

	The task completion frequency	The task completion time
Group 1 vs Group 2	1	0.5067
Group 1 vs Group 3	0.8016	0.7228
Group 2 vs Group 3	0.7768	0.6319

Table 4. Comparison between different types of API documentation

	The task completion frequency	The task completion time
Raw vs eXoaDocs	0.3458	0.3152
Raw vs ADECK	0.0369	0.0003
eXoaDocs vs ADECK	0.1489	0.0004

Table 3 shows the comparison between different developer groups. We can see that all the p-values are larger than 0.05. It means that each pair of developer groups is not statistically different. For example, when comparing Group1 and Group 3, the p-value of the task completion frequency is 0.8016 and the task completion time is 0.7228. Hence, the three developer

groups achieve similar results so that the developers are evenly distributed into groups with similar programming skills.

The comparison results between different types of API documentation are shown in Table 4. We can see that a half of p-values are less than 0.05. According to Table 4, the developers with the raw API documentation are statistically different with the developers with the ADECK-enriched API documentation in terms of the task completion frequency and the completion time. In addition, the developers with the eXoaDocs-enriched API documentation are significantly different with the developers with the ADECK-enriched API documentation in terms of the completion time. Considering that developers using the ADECK-enriched API documentation can complete more programming tasks within less time than those using the raw API documentation, we can say that leveraging the ADECK-enriched API documentation could boost the productivity of developers.

■ S3. Exploring the kind of API types that can be enriched by ADECK

To explore what kind of API types that can be enriched by code samples using ADECK and eXoaDocs, we conduct an additional experiment. Figuring out this question could give more insights into ADECK.

As suggested by the Apatite tool and the Jadeite tool, we rank the Java SE API types sequentially based on their popularities (the number of Google hits). Then, we randomly select 10 API types from the top 10%, middle 10%, and bottom 10% ranked API type list respectively. In such a way, we obtain 30 API types with different popularities in total. Since annotating the quality of the enriched code samples for the selected API types is time consuming and labor intensive, we randomly select 30 API types to be evaluated. The 10 API types sampling in the top 10% in the ranked API type list are regarded as the commonly used API types. The 10 API types from the middle 10% in the ranked API type list are treated as the moderately used API types. In contrast, the 10 API types from the bottom 10% in the ranked API type list are regarded as the rarely used API types. Comparing the quality and quantity of the enriched code samples for these selected API types could uncover what kind of API types can ADECK and eXoaDocs find code samples for.

Table 5. The number of API types that can be enriched by code samples

	ADECK	eXoaDocs
10 API types in top 10%	5	5
10 API types in middle 10%	4	5
10 API types in bottom 10%	3	3
In total	12	13

We repeat the same procedures in RQ2 in the manuscript to conduct the experiment. After the volunteers finished the annotation, we collect and analyze the results from them. We compare ADECK and eXoaDocs from two aspects, i.e., the number of API types that can be enriched by code samples and

the quality of the enriched code samples.

Table 5 shows the number of API types which can be enriched by code samples using ADECK and eXoaDocs. We can see that neither ADECK nor eXoaDocs can enrich all the selected API types, and they can only obtain code samples for a part of API types. The number of code sample enriched API types using ADECK and eXoaDocs are 12 and 13 respectively. From the commonly used API types to rarely used API types, both the number of code sample enriched API types by ADECK and eXoaDocs decreases. As for the popular API types, ADECK and eXoaDocs achieves the same result, i.e., 5 API types can be enriched. In terms of the moderately used API types, eXoaDocs achieves better results than ADECK. From the perspective of the rarely used API types, ADECK and eXoaDocs achieve the same result. On the whole, eXoaDocs can enrich more API types with code samples than ADECK, especially for the moderately used API types.

Fig. 1, Fig. 2, and Fig. 3 show the quality of the enriched code samples for the selected API types. The same as the manuscript, we measure the quality of code samples with three metrics, i.e., correctness, conciseness, and usability.

We can see from the figures that, the quality of code samples enriched by ADECK is better than that of eXoaDocs for both the top 10% and bottom 10% ranked API types. For example, the average correctness of the code samples by ADECK for the API types in top 10% is 4.01. In contrast, it is only 3.66 by eXoaDocs in the same condition. In terms of the API type in the bottom 10%, ADECK is better than eXoaDocs in some metrics, i.e., correctness and usability, but not in the other metric, i.e., conciseness.

In summary, eXoaDocs can enriched more API types with code samples than ADECK, especially for the moderately used API types. However, the quality of the code samples enriched by ADECK is better than eXoaDocs.

The API types in top 10%

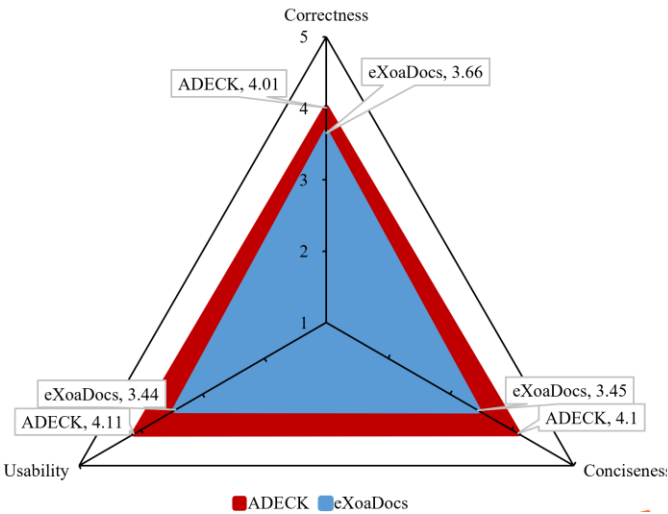


Fig. 1. The quality of code samples for the API types in top 10%

The API types in middle 10%

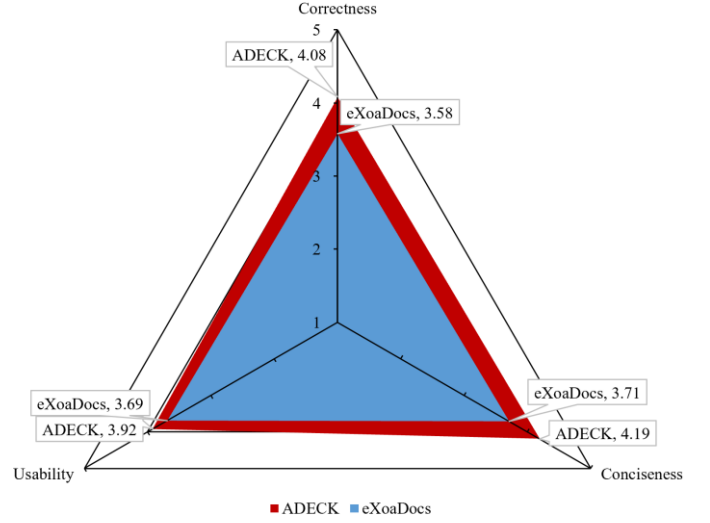


Fig. 2. The quality of code samples for the API types in bottom 10%

The API types in bottom 10%

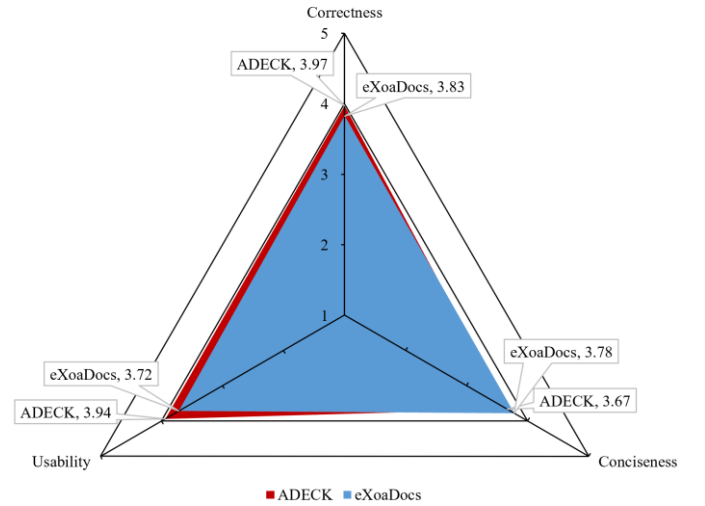


Fig. 3. The quality of code samples for the API types in bottom 10%

We can guess the reason from the design and implementation of ADECK and eXoaDocs. eXoaDocs relies on Google search to obtain code samples, whereas ADECK relies on Stack Overflow. Hence, eXoaDocs can enriched more API types. ADECK select typical code samples according to their use scores, but eXoaDocs does not have such a mechanism. Thus, the quality of code samples enriched by ADECK is better.

■ S4. Exploring the API types covered by ADECK but not by eXoaDocs and vice versa

We conduct an additional experiment to explore what are the API types that can be covered by ADECK but not by eXoaDocs and vice versa. Figuring out this research question could deepen our understanding to the characteristics of ADECK and eXoaDocs.

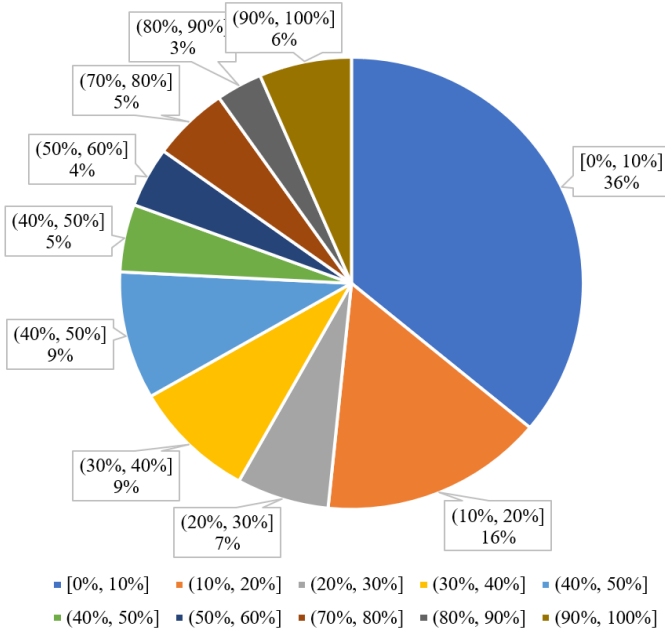


Fig. 4. The distribution of A1.

We first obtain the API types that can be enriched by ADECK but cannot enriched by eXoaDocs (we call these API types A1). Then, as suggested by the Apatite tool and the Jadeite tool, we investigate the distribution of A1. We rank all the API types based on their popularities, and divide them into 10 equally sized intervals, each of which takes up 10%. For example, the all the API types are distributed in [0%,10%], (10%,20%], (20%,30%], ..., (90%,100%]. Among them, [0%,10%] means the API types that ranks in top 10%, (10%, 20%] means the API types that ranks between top 10% to top 20%, and so on. Finally, we obtain the distribution of A1 on the 10 intervals and present the results in Fig. 4. Similarly, we call those API types A2, which can be enriched by eXoaDocs but cannot enriched by ADECK. We also obtain the distribution of A2 on the 10 intervals and show the results in Fig. 5. By presenting the results like this, we can know which kind of API types that A1 and A2 belong to.

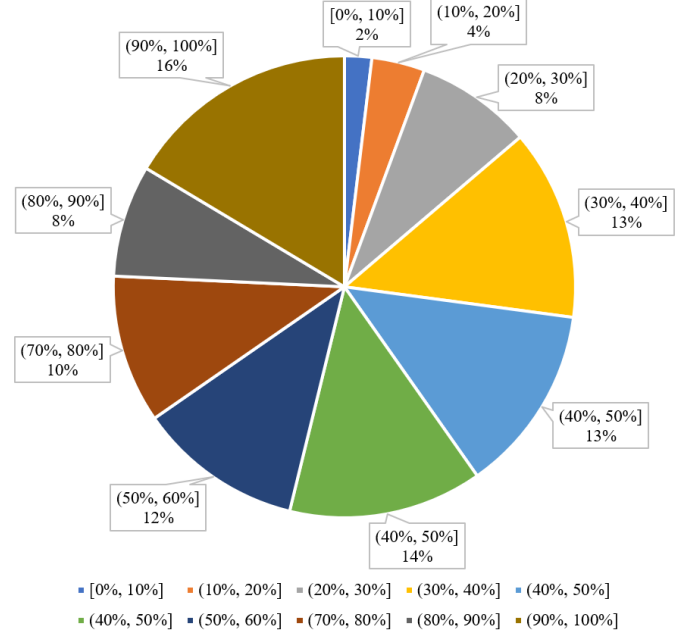


Fig. 5. The distribution of A2.

We can see from Fig. 4 that more than a half of A1 are ranked in top 20%. For example, 36% of A1 are distributed in the interval [0%, 10%], and 16% of A1 are distributed in the interval (10%, 20%]. It means that ADECK can cover more popular or commonly used API types than eXoaDocs. In terms the distribution of A2 presented in Fig. 5, we can see that most of A2 are moderately used and rarely used API types. For example, 86% of A2 are ranked below 30%.

In summary, the more than a half of the API types that covered by ADECK but not by eXoaDocs are commonly used API types. In addition, most of the API types that covered by eXoaDocs but not by ADECK are moderately used and rarely used API types.