A Survey on Understanding How Opinions about APIs can be Summarized Around API Aspects

Gias Uddin and Olga Baysal and Latifa Guerrouj

1 Introduction

Developers share their opinions about different aspects (e.g., performance, usability) of APIs in developer forums. While opinions about an entity (e.g, car, camera, hotels, restaurants) can be about diverse aspects of the entity, it is a common practice to identify most prevalent aspects of the entity around which comparison can be made (e.g., based on the aspect picture quality, which camera is the best?). We are aware of no such formal determination of aspects for the domain of API reviews. We conducted a survey of 64 professional software engineers and asked them how opinions about APIs can be grouped around aspects. We asked them two questions. Whether they agree to a list of predefined aspects and whether there can be other aspects of an API arond which they would like to aggregate the opinions about an API. In this paper, we discuss the methodology used to conduct the survey, and present an exploratory analysis of the responses.

2 Methodology

We asked developers about their perceptions on how opinions about APIs can be collected and summarized from forum posts. With regards to the API aspects, we asked developers two questions:

1) Given a list of 10 API aspects, whether they would like to summarize opinions about APIs around the aspects?
2) Besides the above 10 API aspects, are there any other API aspects about which developers would like to see opinions in the forum posts?

The first question was close-ended. Each of the 10 aspects contained a likert scale: 1) Strongly Agree, 2) Agree, 3) Neutral, 4) Disagree, and 5) Strongly Disagree. The second question was open-ended. The developers were encouraged to write as much as they can on their choice of other API aspects around which they would like to summarize opinions about APIs from forum posts.

To recruit the participants for the study, we sent emails to about 2500 Github users. Each email was personalized. An example of a partial email is shown in Figure 1. The survey was open for about seven weeks (from March 09 to April 30, 2016). In total, we received 64 responses. The first part of the survey asked demogrpahic questions about the participant: 78% of the respondents indicated to be professional software engineers, and 92% indicated that they are actively involved in softwarde development. The majority of the developers (64%) had more than 10 years of professional software development experience.

We analyzed the open-ended question using card sorting [1], by splitting each response to individual meaningful quotes. The quotes were then analyzed by both authors Uddin and Baysal.

- *Gias Uddin, School of Computer Science, McGill University.*
  E-mail: gias@cs.mcgill.ca
- *Olga Baysal, School of Computer Science, Carleton University.*
  E-mail: olga.baysal@carleton.ca
- *Latifa Guerrouj, Department of Software Engineering and IT, École de Technologie Supérieure.*
  E-mail: latifa.guerrouj@etsmtl.ca

3 Results

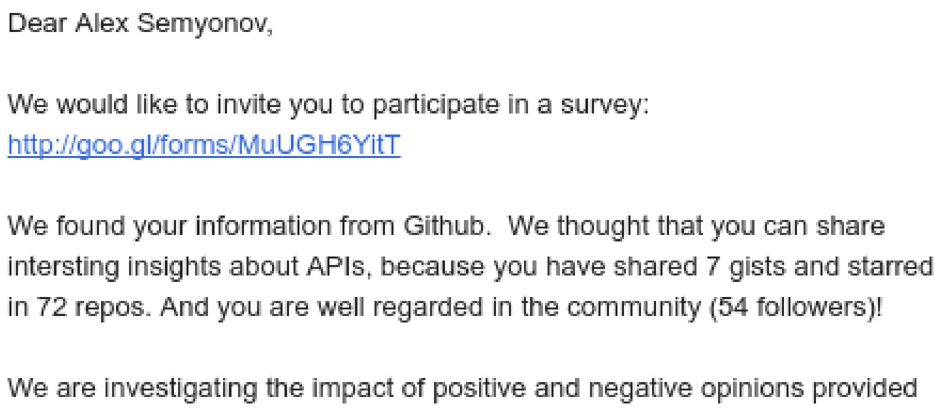
In Table 1, we present the the percentage distribution of the preferences of developers for each API aspect. The first two columns for each aspect show how many of the developers strongly or simply agreed to the fact that opinions about an API need to be summarized around that aspect. The third column shows the summation of the first two columns, i.e., how many of the all participants agreed to see summarized opinions about APIs around the given aspect. The next two columns how many of the total developers disagreed and the last column shows how many remained neutral.

Except 'Only sentiment', the developers agreed more than they disagreed to see opinion summaries about an API around the aspect. For the aspect 'Only sentiment', developers mostly disagreed, i.e., they considered that a simple presentation of mere opinions would not be sufficient.

Some of the additional items that developers look for in API opinions are: real-world examples and scenarios of usage, *"reputation of opinion provider"* (R61) and *"experience"* (R23), presence of good tests, code style, API maturity (*"how long it's been active for"* (R25)), *"extensibility of an API"* (R29), API's learning curve, and discussion of strong vs. weak points such as *"good at x, bad at y"* (R29).

4 Related Work

Natural language summaries have been investigated for software documentation [2], [3], [4], [5] and source code [6], [7], [8], [9]. Murphy [10] proposed two techniques to produce structural summary of source code. Storey et al. [11] analyzed the impact of tags and annotations in source code. The summarization of source code element names (types, methods) has been investigated by leveraging both structural and lexical information of the source code [12], by focusing on contents and call graphs of Java classes [7], based on the identifier and variable names in


Dear Alex Semyonov,

We would like to invite you to participate in a survey:
http://goo.gl/forms/MuUGH6YitT

We found your information from Github. We thought that you can share intersting insights about APIs, because you have shared 7 gists and starred in 72 repos. And you are well regarded in the community (54 followers)!

We are investigating the impact of positive and negative opinions provided

Fig. 1: An example email sent to a participant

TABLE 1: The distribution of preferences of developers for each aspect in the survey

Aspect	Strongly Agree	Agree	Total Agreed	Disagree	Strongly Disagree	Total Disagreed	Neutral
Performance	26.67	60.00	86.67	3.33	0	3.33	8.33
Usability	48.33	36.67	85.00	3.33	0	3.33	10.00
Security	36.67	48.33	85.00	0	0	0	13.33
Bug	41.67	45.00	86.67	1.67	1.67	0	10.00
Compatibility	20.00	55.00	75.00	1.67	0	1.67	21.67
Portability	10.00	40.00	50.00	6.67	0	6.67	35.00
Community	23.33	50.00	73.33	3.33	0	3.33	21.67
Legal	13.33	30.00	43.33	16.67	0	16.67	35.00
Documentation	38.33	50.00	88.33	0	0	0	10.00
Only Sentiment	3.33	11.67	15.00	21.67	0	21.67	40.00
Others	6.67	41.67	48.34	3.33	0	3.33	46.67

methods [6]. A more recent work proposed a technique to generate automatic documentation via the source code summarization of method context [8]. The notion of context is very important for developers because it helps understand why code elements exist and the rationale behind their usage in the software [13], [14], [15]. The selection and presentation of source code summaries have been explored through developer interviews [16], as well as eye tracking experiment [17]. Our findings confirm that developers also require different types of API opinion summaries.

Acknowledgements

We thank all developers who took part in our survey for their time, participation, and feedback.

References

[1] M. Miles and A. Huberman, *Qualitative Data Analysis: An Expanded Sourcebook*. SAGE Publications, 1994.

[2] G. M. Sarah Rastkar, Gail C. Murphy, "Automatic summarization of bug reports," *IEEE Trans. Software Eng*, vol. 40, no. 4, pp. 366–380, 2014.

[3] G. C. Murphy, M. Kersten, and L. Findlater, "How are java software developers using the eclipse ide?" *IEEE Softawre*, vol. 23, no. 4, pp. 76–83.

[4] R. Holmes, R. J. Walker, and G. C. Murphy, "Approximate structural context matching: An approach to recommend relevant examples," *IEEE Trans. Soft. Eng.*, vol. 32, no. 12, 2006.

[5] J. Anvik and G. C. Murphy, "Reducing the effort of bug report triage: Recommenders for development-oriented decisions." *ACM Trans. Softw. Eng. Methodol*, vol. 20, no. 3, pp. 952–970.

[6] G. Sridhara, E. Hill, D. Muppaneni, L. Pollock, and K. Vijay-Shanker, "Towards automatically generating summary comments for Java methods," in *Proc. 25th IEEE/ACM international conference on Automated software engineering*, 2010, pp. 43–52.

[7] L. Moreno, J. Aponte, G. Sridhara, M. A., L. Pollock, and K. Vijay-Shanker, "Automatic generation of natural language summaries for java classes," in *Proceedings of the 21st IEEE International Conference on Program Comprehension (ICPC'13)*, 2013, pp. 23–32.

[8] P. W. McBurney and C. McMillan, "Automatic documentation generation via source code summarization of method context," in *Proceedings of the 21st IEEE International Conference on Program Comprehension*, 2014, pp. 279–290.

[9] L. Guerrouj, D. Bourque, and P. C. Rigby, "Leveraging informal documentation to summarize classes and methods in context," in *Proceedings of the 37th International Conference on Software Engineering (ICSE) - Volume 2*, 2015, pp. 639–642.

[10] G. Murphy, *Lightweight Structural Summarization as an Aid to Software Evolution*. University of Washington.

[11] R. I. B. P. C. R. Margaret-Anne D. Storey, Li-Te Cheng, "Shared waypoints and social tagging to support collaboration in software development," in *CSCW*, pp. 195–198.

[12] A. M. S. Haiduc, J. Aponte, "Supporting program comprehension with source code summarization," in *In Proceedings of the 32nd International Conference on Software Engineering*, 2010, pp. 223–226.

[13] E. Duala-Ekoko and M. P. Robillard, "Asking and answering questions about unfamiliar APIs: An exploratory study," in *Proc. 34th IEEE/ACM International Conference on Software Engineering*, 2012, pp. 266–276.

[14] J. Sillito and G. C. M. K. D. Volder, "Asking and answering questions during a program change task," *Journal of IEEE Transactions on Software Engineering*, vol. 34, no. 4, pp. 434–451, 2008.

[15] S. E. Sim, C. L. A. Clarke, and R. C. Holt, "Archetypal source code searches: A survey of software developers and maintainers." in *IWPC*. IEEE Computer Society, pp. 180–.

[16] A. T. T. Ying and M. P. Robillard, "Selection and presentation practices for code example summarization," in *Proc. 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2014, pp. 460–471.

[17] P. Rodeghero, C. McMillan, C. McMillan, N. Bosch, and S. D'Mello, "Improving automated source code summarization via an eye-tracking study of programmers," in *Proc. 36th International Conference on Software Engineering*, 2014, pp. 390–401.