Social Health Cues Developers Use when Choosing Open Source Packages

Andrew Head
UC Berkeley
Berkeley, CA, USA
andrewhead@berkeley.edu

ABSTRACT

Developers choose open source packages from many alternatives. One increasingly important factor when choosing a package is its "social health", or a developer's ability to get help on communication channels. We conduct a study to understand how developers learn about the social health of open source packages before using them. We offer preliminary results of the cues developers find.

CCS Concepts

•**Information systems** → *Social networks;* Web log analysis; Internet communications tools; •**Software and its engineering** → *Documentation; Open source model;* Search-based software engineering;

Keywords

Social health; community; packages; open source

1. INTRODUCTION

How should a developer choose a package from among many choices? This question gets more difficult each day as new open source packages are published. We want to build visualizations and search tools to help developers make such choices more quickly and effectively.

We start by asking what information matters to developers. Typically, developers choose open source packages based on functionality and quality of documentation. We suggest that as social communication channels become intertwined with the development process [14], developers need to understand the "social health" of a package's community before they use it. We define a package as *socially healthy* when a developer can reliably get helpful answers from a package's forums, mailing lists, and other communication channels.

We ran a study to find out how developers currently learn about the social health of packages. We contribute preliminary results from this study, including the following: (1) Developers seek out common cues when answering the same questions about a package's social health; (2) They rely on texts of issue reports, documents, and conversations to assess social health.

2. OUR APPROACH

We invited 10 developers to a study where they compared the social health of two packages. Each participant chose a pair of packages, out of three pairs. Then we asked each participant to compare the two packages, using six questions about each package's community, documentation, and developers. The questions were based on related work and conversations we had with software developers, and included:

- Which community will be more welcoming when responding to questions you ask?
- Which package's documentation will be more up-to-date with the code?
- Which package's developers can you better trust to make reliable, usable software?

These three questions (out of six) were based on observations of anti-social behavior in developers' communication channels [14], concerns about how up-to-date documentation is [7, 10, 12, 14], and how developers assess code example quality based on author reputation [12].

Participants answered the questions using the web. We note developers often get answers face-to-face [6, 14], over email [5, 6], or by trying out code [4]. However, we focused on information on the web, as we believed this would reveal signals we could incorporate into future search tools.

We collected three measurements of cues participants used to answer social questions: a timestamped log of visited URLs; self-reported ratings of web pages' "helpfulness" for answering each question; and open-ended responses of what evidence on the web was most helpful for comparing the social health of the two packages for each question. We report a view of our current results here. Our on-going analysis has focused on two research questions: (1) What sites do participants use to learn about a package's social health? (2) What challenges do developers face when learning about a package's social health?

3. PRELIMINARY RESULTS

We observed several noteworthy strategies participants took to learn about a package's social health. First, participants read relevant text to assess social health. In particular, participants skimmed text from conversations to determine how welcoming communities were. Participants also

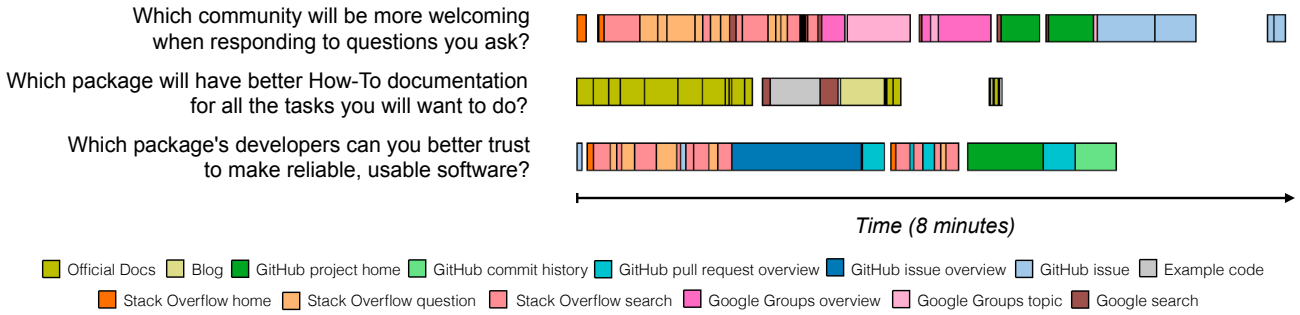

Figure 1: We asked developers to answer questions about the social health of open source projects, specifically their community, documentation, and developers. This plot shows the search behavior of a single participant for three of the six questions. This participant viewed many documents to judge the difference in social health between two packages. To determine which package had a more welcoming community, they viewed seven questions on Stack Overflow, four issues on GitHub, and three conversations on Google Groups.

read texts from documentation about a project's API and development philosophy to determine if a package was designed for users with their backgrounds and goals. Existing package comparison tools do not support reading relevant conversations and documentation. Instead, such tools only show summary statistics (e.g. [1, 3]).

Participants sometimes looked for social health cues in the form of advice from current users of a package. Participants queried the web through search engines to find prebuilt summaries describing the community and documentation. Participants found explicit comparisons of packages on Reddit, and user testimonials on Quora. One participant consulted issue reports to determine how up-to-date documentation was. When participants looked at texts of questions, documents, and issue reports, they chose a handful of pages from dozens or even hundreds.

Our URL log data suggests common places where participants found social health cues. Participants relied on different types of web pages when answering each social health question. We have seen participants find: (1) whether a community is welcoming by viewing Q&A sites and discussions on Reddit and Google Groups; (2) documentation recency by viewing issue reports, code contribution histories, and pull request contents; (3) the trustworthiness of developers by viewing issue reports, and profiles on code hosting sites. Figure 1 shows a cross-section of the URLs one participant visited when answering three of the six questions. We leave quantitative analysis of these trends and a full exploration of specific cues from these sites for future work.

Our preliminary results confirm a need for search tools that reveal obscure information about social health quickly. Participants frequently realized they missed important information after twenty or thirty minutes of learning about a package. One participant wrongly assessed a package had no community at all. Another participant failed to find a newer version of a package under a different name. And another participant missed large repositories of example code written by the package's developers. Such incorrect judgments could be costly to reverse: One participant read source code before realizing that one package's programming API was preferable to one he favored before.

We believe these incorrect judgments are a product of developers' habits and the limitations of current information interfaces. With attention to both, we can design more powerful search tools to help developers quickly and effectively learn about packages' social health when choosing between them. For now, these preliminary results suggest that package consumers should not trust their initial instincts, and should inspect comparisons, discussions, and Q&A for a well-informed picture of a package's social health. Developers should know that cues about the social health for their projects could exist on dozens of domains, and potential consumers may need help learning where to look for help.

4. RELATED WORK

Developers work in an immense online network. Developers benefit from this network by leveraging social media and other channels to stay informed and connect with other developers [13, 15]. Today there is a proliferation of socially-enabled channels [14] through which developers answer each other's questions [9], stay up-to-speed with rapidly changing software [8], help each other overcome bugs and learn new tools [11], and share information in many different forms at many different speeds. The knowledge and conversations of programmers are increasingly distributed across the web.

Some community-developed tools aim to help developers compare packages. Such community tools only show a fraction of the metrics used by developers. Typical metrics for these tools (e.g., [1, 2, 3]) tend to be based on counts and rates of code contributions, issue resolutions, downloads, and "stars" on code hosting sites like GitHub. Developers may be concerned with how much they respect the authors of code [12], how up-to-date the documentation is [7, 10, 12, 14], and whether the community is anti-social [14]. We propose that carefully selected samples from communication channels can help developers make more informed judgments based on social health. The current work presents a study to help us pick these samples.

5. CONCLUSION

In this extended abstract, we propose that new tools can help developers make sense of a sprawl of social information for packages they choose. By presenting preliminary results from a study with developers, we show evidence of what social signals exist on the web and where. Our results will inform the design of search tools to help developers understand packages and their social health.

6. REFERENCES

[1] Awesome Python. https://python.libhunt.com/.

[2] package-quality. https://github.com/alexfernandez/package-quality.

[3] Ruby Toolbox. https://www.ruby-toolbox.com/.

[4] J. Brandt, P. J. Guo, J. Lewenstein, M. Dontcheva, and S. R. Klemmer. Two studies of opportunistic programming: Interleaving web foraging, learning, and writing code. CHI '09.

[5] A. J. Ko, R. DeLine, and G. Venolia. Information Needs in Collocated Software Development Teams. ICSE '07.

[6] T. D. LaToza, G. Venolia, and R. DeLine. Maintaining mental models: A study of developer work habits. ICSE '06.

[7] T. C. Lethbridge, J. Singer, and A. Forward. How software engineers use documentation: the state of the practice. *IEEE Software*, 20(6), 2003.

[8] M. Linares-Vásquez, G. Bavota, M. Di Penta, R. Oliveto, and D. Poshyvanyk. How do API changes trigger stack overflow discussions? a study on the Android SDK. ICPC '14.

[9] L. Mamykina, B. Manoim, M. Mittal, G. Hripcsak, and B. Hartmann. Design lessons from the fastest q&a site in the west. CHI '11.

[10] J. Nykaza, R. Messinger, F. Boehme, C. L. Norman, M. Mace, and M. Gordon. What programmers really want: results of a needs assessment for sdk documentation. SIGDOC '02.

[11] C. Parnin, C. Treude, and M. A. Storey. Blogging developer knowledge: Motivations, challenges, and future directions. ICPC '13.

[12] M. P. Robillard and R. Deline. A field study of API learning obstacles. *Empirical Software Engineering*, 16(6), 2011.

[13] L. Singer, F. Figueira Filho, and M.-A. Storey. Software engineering at the speed of light: how developers stay current using twitter. ICSE '14.

[14] M.-A. Storey, L. Singer, B. Cleary, F. Figueira Filho, and A. Zagalsky. The (R) Evolution of social media in software engineering. FOSE '14.

[15] M. A. Storey, A. Zagalsky, F. Filho, L. Singer, and D. German. How Social and Communication Channels Shape and Challenge a Participatory Culture in Software Development. *IEEE Transactions on Software Engineering*, PP(99), 2016.