Text Analysis

Michael Lundquist

June 9, 2019

In “Creating and Evolving Developer Documentation: Understanding the Decisions of Open Source Contributions” Barthélémy Dagenais and Martin P. Robillard discuss the aspects of effective documentation. Using historical analysis and interviews, the authors determined how documentation is written and best practices for writing documentation. The historical analysis was primarily used to determine how documentation is currently written. Interviews were primarily used to determine pros and cons of different types of documentation. Although documentation methods differ for different types of projects, all projects benefit from good documentation in similar ways. Documentation can be a huge competitive advantage for a project. According to one of the interviewed contributors,” even if his project launched a year after a competing project, the user base grew quickly because the competing project had no documentation.” (Dagenais & Robillard, 2010, p.6)

Although this study provided insights into effective documentation practices, the study's qualitative methods were insufficient to draw solid conclusions. In the study's historical analysis, the authors ”manually inspected more than 1500 revisions of 19 documents selected from 10 open source projects.”(Dagenais & Robillard, 2010, p.1). Nineteen documents are hardly enough to draw serious conclusions from but can suggest patterns in how documentation is made that warrant future research. The project's authors interviewed 12 contributors to and 10 users of open source projects. These interviews were about a half-hour long. Again, half-hour interviews don't prove anything, but they do give insights into some expert's experiences developing documentation.

1 Types of Documentation

The study analyzed the pros and cons of different types of documentation.

The first type of documentation the authors analyzed are wiki pages. Wiki pages, like Wikipedia, are web pages that anyone can edit. As you would expect, allowing anyone to edit a wiki page can be a bad idea. First, because anyone can edit the wiki, they ”lack authoritativeness” (Dagenais & Robillard, 2010, p.5). Furthermore, because wiki pages become ”less concise” (Dagenais & Robillard, 2010, p.5) and even suffer from SPAM. According to the article, ”24.1% of the revision in Firefox”(Dagenais & Robillard, 2010, p.5) were SPAM. Because of these problems, ”all of the projects we surveyed that started on a wiki (4 out of 12) moved to an infrastructure where contributions to the documentation are controlled”(Dagenais & Robillard, 2010, p.5). Despite these problems, wiki pages are extremely effective at encouraging community evolvement, which is critical to open source projects. To prevent anonymous actors from SPAMing the documenta-

tion, but still allow community involvement, the authors suggested using a comment section and encouraging feedback through various channels.

The second type of documentation analyzed was getting started documentation. Getting started documentation is designed to get users to a base level of competency with a tool as quickly as possible. If getting started documentation is well written, it can also serve as marketing material. Unfortunately, "Finding a good example on which to base the getting started documentation, an example that is realistic but not too contrived, is difficult" (Dagenais & Robillard, 2010, p.6 Contributor 11). Good getting started documentation is useful when learning frameworks because it gives the user a simple example that's consistent the framework's principles but doesn't overload the user.

The final type of documentation is reference documentation. Good reference documentation should describe a single function in simple terms. According to the authors for "atomic functions, reference documentation is the most appropriate documentation type to begin with because, as contributor C11 mentioned, it can be difficult to create getting started documentation that shows examples calling many functions" (Dagenais & Robillard, 2010, p.6). Libraries of atomic functions, for example rest APIs, are very different from frameworks that require a conceptual understanding of the framework to be useful. Fortunately, writing reference documentation is much easier than writing getting started documentation because it only requires documenting small parts of a code base.

2 Documentation Management

The paper also discussed some miscellaneous observations about the documentation process.

The paper discussed how useful documentation tools can be. Tools cause many syntax errors in documentation. According to the authors, they "were responsible for an important amount of changes and that better tool support could probably mitigate this problem: 55.4% in Eclipse (HTML), 11.4% in Django (Sphinx), 11.1% in GTK (SGML), and 6.7% in WordPress (wiki)" (Dagenais & Robillard, 2010, p.4). Despite all these errors caused by documentation tools, they are still invaluable. Without documentation tools, documentation becomes scattered and unstandardized.

Although having a dedicated documentation tool is critical, having a dedicated documentation team can be disastrous. When a separate team exclusively writes documentation, "code contributors outnumbered documentation contributors so the documentation lagged behind the

implementation"(Dagenais & Robillard, 2010, p.7). Additionally, the documentation team could misunderstand how the contributors' code works and mis-document it.

An excellent way to avoid having a dedicated documentation team is to require all code contributions to include related documentation. Then, if users then have questions about a developer's documentation, it can be revised as needed. Having a policy that enforces developers to include documentation with their contributions requires the developers to think about how their contributions actually work. When developers think how their contributions work, they often catch mistakes. In "Creating and Evolving Developer Documentation: Understanding the Decisions of Open Source Contributions", the authors call this proof-reading process "embarrassment-driven development" (Dagenais & Robillard, 2010, p.6).

Embarrassment-driven development can be useful on the macro scale also. The historical analysis discussed in the paper revealed that projects' documentation often gets bursts of activity over a short period of time. When interviewed, the projects' lead contributors said some of these bursts of activity were due to book deals. These book deals required the lead contributors describe how their project work which forced them to re-evaluate much of their documentation.

3 Conclusion

Although this paper doesn't make any strong conclusions, it is still useful to readers. First, it gives an understanding of what type of documentation (wiki, getting started or reference docs) is appropriate for a situation. Then it discusses the tools, protocols and general patterns involved when writing documentation. This paper would be very useful to anyone starting an open-source project.

References

Dagenais, B., & Robillard, M. P. (2010). Creating and evolving developer documentation: Understanding the decisions of open source contributors. In *Proceedings of the eighteenth acm sigsoft international symposium on foundations of software engineering* (pp. 127–136). New York, NY, USA: ACM. Retrieved from `http://doi.acm.org/10.1145/1882291.1882312` doi: 10.1145/1882291.1882312