Understanding FLOSS through community publications: Strategies for Grey Literature Review

Melissa Wen, Leonardo Leite, Fabio Kon
{wen,leofl,kon}@ime.usp.br
University of São Paulo, Brazil

Paulo Meirelles
paulo.meirelles@unifesp.br
Federal University of São Paulo, Brazil

ABSTRACT

Over the last decades, the Free/Libre/Open Source Software (FLOSS) phenomenon has been a topic of study and a source of real-life artifacts for software engineering research. A FLOSS project usually has a community around its project, organically producing informative resources to describe how, when, and why a particular change occurred in the source code or the development flow. Therefore, when studying this kind of project, collecting and analyzing texts and artifacts can promote a more comprehensive understanding of the phenomenon and the variety of organizational settings. However, despite the importance of examining Grey Literature (GL), such as technical reports, white papers, magazines, and blog posts for studying FLOSS projects, the GL Review is still an emerging technique in software engineering studies, lacking a well-established investigative methodology. To mitigate this gap, we present and discuss challenges and adaptations for the planning and execution of GL reviews in the FLOSS scenario. We provide a set of guidelines and lessons learned for further research, using, as an example, a review we are conducting on the Linux kernel development model.

KEYWORDS

Grey Literature, Literature Review, FLOSS, Linux, Methodology

ACM Reference Format:
Melissa Wen, Leonardo Leite, Fabio Kon and Paulo Meirelles. 2020. Understanding FLOSS through community publications: Strategies for Grey Literature Review. In *New Ideas and Emerging Results (ICSE-NIER'20), May 23–29, 2020, Seoul, Republic of Korea.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3377816.3381729

1 INTRODUCTION

The Free/Libre Open Source Software (FLOSS) phenomenon and its development process have been studied by both academia and industry. Their interest stems from the popularity of many FLOSS projects and their successful use of "peculiar" management and training practices to conduct a high-quality collaborative, geographically distributed, community-based software development effort.

Participants in the FLOSS ecosystem use and produce informative resources and artifacts to describe, prescribe, or proscribe what happens in a FLOSS project. Such community production includes artifacts as source code files, wikis, roadmaps, and mailing-list messages; and also publications not indexed by scientific repositories as working papers, white papers, annual/technical reports, blogs, videos, and web pages. This kind of information, not controlled by commercial publishers, is also known as Grey Literature (GL) [12], being produced by academia, industry, government, and hobbyists.

Alongside commercial and academic publications, GL can be used to broaden the understanding of how FLOSS developers interpret the environment of which they are part. Despite the informality of GL, it brings advantages of its own. Studies with negative or null results are easier to find in GL than in peer-reviewed academic literature, enabling a more critic perspective, with a potential reduction of bias and visualization of more balanced evidence [10]. GL is also a leading source for identifying topics and gaps not yet covered by academic literature. It also enables investigating more up-to-date and emerging information since academic production incurs long delays for publication due to the peer-review process.

Although it is still a nascent trend, a few software engineering reviews have already included GL material [2, 4, 13]. Nevertheless, we argue that the use of GL material is even more fruitful in FLOSS research, given the vast amount of public information resources produced by the FLOSS communities. In such a scenario, we claim that researchers should investigate such resources before taking the time of FLOSS contributors with surveys or interviews [5], especially considering that *i)* most of the needed information is already available with rich details and *ii)* top contributors usually do not have much time for extensive interviews.

Although there are recommendations for the inclusion of GL material in literature reviews, there is not yet a precise methodology or guidelines prescribing well-defined steps and restrictions for conducting Grey Literature Reviews (GLR). Some researchers have mitigated this problem by applying systematic review methods for examining GL. Nevertheless, conducting GLR still requires adaptations for handling the variety in quality, size, types, and structures of documents, besides the larger volume of available materials in comparison to a traditional literature review. Moreover, the reviewer must deal with the lack of robust search engines and with the challenges of textual and natural language analysis.

In this paper, we highlight the challenges, specificities, adaptations, and lessons learned from our experience in conducting a GLR in the context of a software engineering research on FLOSS. This paper is, thus, a first effort in defining a methodology for conducting GLRs for software engineering research. In particular, we present guidelines for supporting future FLOSS literature reviews.

Our experiences reported in this paper are based on our current endeavor in conducting a GLR about the Linux kernel development model. We are examining outputs of the Linux community to capture the attributes that practitioners use to describe the Linux development process, including project management characteristics, organizational structure, workflow, and decision-making process.

ICSE-NIER'20, May 23–29, 2020, Seoul, Republic of Korea

ACM ISBN 978-1-4503-7126-1/20/05…$15.00
https://doi.org/10.1145/3377816.3381729

Such review represents a suitable case of GLR on FLOSS for several reasons. The Linux project is one of the largest, most famous, and most consolidated FLOSS projects. Its development is connected in many ways to the history of the free software and open-source movements. The characteristics of its project inspired software development models; 21 years ago, it inspired Raymond for the definition of "The Cathedral and The Bazaar" models [11]. Its long history, large community, and the interest of software engineering studies in understanding its practices provide a great wealth of open access material available for investigation.

2 METHODS AND ADAPTATIONS

The inclusion of GL in systematic literature reviews has been increasingly widespread in the medical field [7, 10]. However, in software engineering, this procedure is not easily found yet. For that reason, we relied on guidelines from studies in both areas [6, 7, 10, 13] to structure a search plan suitable for FLOSS publications.

For addressing the lack of well-established methods, with defined steps and constraints, we adapted systematic review methods to examine community productions. Systematic Literature Review (SLR) is a research methodology that follows rigorous procedures to synthesize and evaluate the available evidence on a focused topic [3], supporting the development of evidence-based guidelines for practitioners [9]. Combining SLR methods with GLR helps shedding light on important industrial practices still not mapped by conventional software engineering studies [13].

As with SLRs, the methodological plan of a GL search provides guidelines, structure, and transparency to the search methods [7]. However, due to the characteristics of grey materials, GLR searches are often less methodical than traditional systematic searches for academic publications. To reduce searching bias, the researcher should map data sources and types, search terms, selection criteria, and boundaries. This approach also helps to manage the variety of search terms and the volume and diversity of "grey" materials available throughout the web.

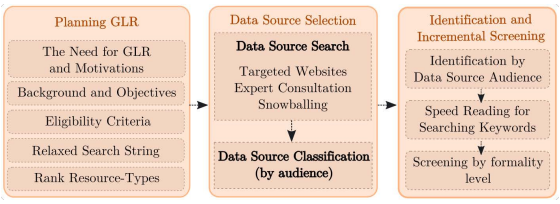

Figure 1: Our GLR Search Flow

From our experience, we recommend that the GLR planning should cover four essential aspects to ensure the quality of the selected documents: (1) outline the problem and define the research question, (2) define the inclusion and exclusion criteria, (3) develop a relaxed search string, and (4) define the resource-types to consider. In addition, we developed and applied additional methods to refine the screening step and complete the GL search: (5) data source evaluation and selection and (6) incremental screening process. We mapped challenges and adaptations for GLR on FLOSS, and summarized our review search flow in Figure 1.

Outline the problem and define the research questions. The first step in a typical GLR planning process is to establish the need for this type of review. Researchers should then describe the background and objectives of the study in a protocol [8] to delimit the scope of research and define research questions.

Our example, the community-based distributed development of the Linux kernel project, provides a wealth of alternative resources of information produced by professionals. This materials examination can ensure the most current picture of the Linux kernel development model and, consequently, improve understanding of FLOSS state of practice. As example, some organizations periodicaly provide reports about the State of Linux Kernel Development, which promptly discusses the Linux development statistics, who is working on the kernel, who is sponsoring the development, and so on. In this regard, the research question *"How do FLOSS development communities define the current Linux development model?"* guided our review case.

Combine Eligibility Criteria and Control Factors. Reviewing the GL poses many validity risks since these materials are not reviewed or provided by traditional publishers and channels. Besides this, evidence in them is often based on personal experience and opinion, at the risk of catching fake news and views from people outside the problem set. Based on the experiences reported by Soldani et al. [13] and Garousi et al. [6], we recommend defining inclusion and exclusion criteria and combining them with additional control factors to mitigate this problem.

For our review, we selected documents that satisfied all of the following eligibility criteria: (1) Published between 2009 and 2019. (2) Published online by institutions, industry-oriented magazines, and practitioners of the FLOSS area. (3) Available in English. (4) Most current version of the document. (5) Describes developments in the Linux kernel by: reporting practices; presenting statistics; discussing management and rules inside the project; or studying the project development and/or its community.

We also defined some control factors to assure sample quality: (1) The document is published by *i)* a reputable organization or magazine, *ii)* an individual author associated with such organizations and magazines, or *iii)* a practitioner with more than five years of FLOSS experience. (2) The document is publicly accessible. (3) Historical facts, statistics, or Linux kernel project artifacts support the statements in the document. (4) The content is not centered on technical issues / specific features.

Define a Relaxed Search String. Defining a flexible set of search terms, based on the problem definition and eligibility criteria, is essential to deal with the variety of terminologies used by persons and institutions for a given study object, since popular keywords in software engineering studies are still not commonly used by practitioners. Besides, a GL writer has little concern with standardization and structuring. Consequently, they usually use more informal and heterogeneous terms. Another issue is related to applying the search string in a website search engine. Much of these tools cannot address both diversity and lack of text structure. Therefore, even with the prediction of synonyms, we observed during the search process that broader and simple strings were essential to reach potentially relevant documents. Finally, the diversity of the audience of each data source required a warm-up round to understand

how an organization or person commonly refers to search string elements and evaluate the coverage of these terms accordingly.

For the Linux project case, we first defined a base search string, *Linux development model*. Then, we broke it into words and listed potential synonyms for each word, as follows: **Linux** → Linux Kernel; **Development** → Business, (empty string); **Model** → Scheme, Rules, Process, Guide, Community, Culture, (empty string).

Resource-types Outline. Considering the great diversity of GL resources, list the types of data to be considered for analysis. Then, separate this data into *in-text content* and *audiovisual content*, since the latter usually demands more effort to inspect than the former. Based on the categorization in "shades of GL" presented by Adams et al. [1], rank the resource types according to expected levels of formality. This rank should guide the document review order and should also support the development of appropriate stopping criteria for the screening phase.

In our case, we discarded audiovisual resources due to the effort required to examine this content and the difficulty of the transcription process. We also assessed the expected quality of textual contents from their formality levels. From this assessment, we decided not to include mailing-list emails, because these records often have opinion-based discussions. Nevertheless, mailing-list e-mails can be an interesting avenue for future research.

Extensive Selection of Information Sources. In a review of the traditional literature, data source selection often consists of a set of similar digital libraries from prestigious organizations in academic computer science publications or similar. However, for GL of FLOSS projects, data sources are not so obvious, and selecting them requires specific knowledge and consultation of experts.

A good data source selection minimizes GLR drawbacks such as time spent and omitting significant evidence. To select data sources of GL suitable for FLOSS studies, the researcher should consider carefully the initial search string and the content types. We also recommend four strategies for searching data sources:
(1) Search for targeted source: Conduct a Google search to identify organization webpages, magazines, and relevant blogs on FLOSS development. Use the advanced search tool to restrict results to English documents only. In this step, we evaluated the first ten pages of search hits and manually inspected each website for data source selection. We also evaluated blog posts that list relevant pages for those interested in tracking Linux project updates.
(2) Consult experts: Consultation with experts can help discover other relevant data sources. We consulted four of them and identified additional data sources not previously covered.
(3) Apply snowballing: Add data sources if links or references to their contents repeatedly appear during the sample selection phase.
(4) Classify the data sources: The data sources should be classified according to their proximity to the object of study, in our case, the Linux kernel development. We divided data sources into sources that deal specifically with the Linux kernel and references that deal more broadly with IT or FLOSS subjects. The publication collection did not follow any order of priority; however, to improve the performance of the review process, a researcher should first evaluate contents in the specific data sources, and use data from the other sources as complementary information.

Since search engines use unique algorithms to generate their relevance ranking schemes, the researcher must be prepared to address the singularities of each data source. Accordingly, we recommend a prior evaluation of data sources in terms of audience, text structure, search tools, quality of search results display, and updates. This step helps to identify the main subject addressed, potential search pitfalls, and also the level of formality of documents.

For our review, the mentioned data source search strategies reached 15 data sources: eight magazines, three developer blogs, two practitioners-oriented news websites, one institutional website, and one wiki page. We evaluated each of them and discarded one for not allowing open access to publications. We flagged two others to receive proper procedures in the evidence collection and analysis phase. One had a restricted time interval (it was deactivated in 2015) and the other had its content only partially screened, as not everything was openly available.

Some data sources did not provide a search tool, and others had varying quality of features. The main issues faced were: (1) site pagination only worked until page 10; (2) the date of the indexed article in the search was not the same as the date of publication; (3) it was not possible to filter or sort by date; (4) results were shown in blocks rather than lists; (5) there was no pagination; (6) did not present any results ordering; (7) did not allow filter searching; (8) did not enable search by an expression, only search by any of the words of the string; (9) search for all words in the string, but in arbitrary order; (10) limit on the number of documents returned.

To overcome weak search mechanisms, we recommend the following strategies: *(1) Choose a suitable order of search hit:* by date, when it is possible to search by an exact sequence of words; by relevance when the tool only seeks for any of the terms in the string. *(2) Define stopping criteria* influenced by weak search engines or large volumes of data, such as page number restriction, a suitable number of hits, the decline in quality, and availability of evidence. *(3) Consider searching by tags/categories*, whenever possible.

A researcher should perform a manual inspection of publication titles and dates when a data source does not provide any search engines but had great data potential (such as some blogs of experienced developers or important personages).

Incremental Screening Process. Even in systematic reviews, changes in a review protocol are sometimes necessary to adjust methods to unanticipated circumstances [8]. Despite all efforts to adhere to the search plan, we used some strategies to overcome unexpected limitations and difficulties during document screening. We have developed an incremental method for handling the number of search occurrences and the lack of standard indexing. This method consists of dividing the screening process into data source audiences and iteratively conducting the sample selection. In each iteration, the inspection becomes more detailed and follows the ascending order of document formality.

For our review of Linux kernel publications, we divided the selection process into two steps according to data source audience: evaluate the most specific Linux source documents first and then the most widespread for the IT industry. We documented this screening process in a spreadsheet[1] divided into two tabs: (I) Linux-oriented

[1] Data available on gitlab.com/ccsl-usp/glr/-/raw/master/linux-case-data.pdf

publishers (8 data sources); and (II) IT-Industry-oriented publishers (7 data sources). In the first tab, we identified 236 documents of publishers and practitioners experts on Linux. In the second tab, we collected 104 documents published in magazines and blogs of general IT industry subjects. Each spreadsheet row represents a potentially relevant document. Seven columns index the document data: (1) date of last update or access, (2) title, (3) authoring organization or person, (4) category of the data source, (5) URL, (6) search strategy that located it, (7) links or references in the document.

GL rarely presents standard indexing and controlled vocabulary, with a good quality title, abstract, or keywords to assist in the inclusion or exclusion of the document. Hence, we also developed an incremental exclusion process. First, remove duplications or documents with only a link for another one, bringing the linked material to the sample. Second, try to evaluate titles, looking for any relation to the object under study. However, as titles in GL are often not accurate or are "attractive" but misleading, this process was not very productive. Because of this, we opted for a speed reading of the full-text, searching for the keyword *Linux* to quickly understand how the project was addressed in that document and evaluate the exclusion criteria. When the eligibility criteria were not clear, we kept the document in the list. Finally, we classified the material according to the categorization of GL of Adams et al. [1] and performed the full-text screening following this order of priority: (1) less informal literature first; (2) maintainers blog posts; (3) reports of project landmarks and emblematic cases; (4) documents from the snowballing process.

3 CONCLUSION

Software engineering literature still presents little of what is performed by FLOSS practitioners, what activities they do, and how often they happen. This distancing from the practice leads academic works to lack a well-understanding of the FLOSS phenomenon, carrying a homogeneous and biased point of view. Through this preliminary investigation, we claim that software engineering research must accept and analyze knowledge artifacts produced by practitioners to enrich knowledge about FLOSS development. The strategies reported here could boost the use of GLR to address FLOSS matters and favor future FLOSS investigations. On the other hand, GL search methods do not have the same high standards of transparency and reproducibility as the search in academic libraries.

Our need for a GLR came from a case study of the Linux kernel project, which aimed to describe the Linux kernel development model from the FLOSS community. Linux is both a typical case of a successful FLOSS project and an extreme case of vigor in the publications made by its community. The Linux kernel software and its community-project have peculiarities compared to other FLOSS projects. Due to the software type, its complexity, longevity, and also project community size, the Linux kernel is the topic of a large number of publications and has a high availability of materials produced by practitioners, unlike many other FLOSS projects. Therefore, obtaining information and performing some GLR steps can vary in complexity when applied to investigations of other FLOSS projects. Nevertheless, due to the Linux project relevance and protagonism in the last two decades of FLOSS research, an accurate mapping of its current state of development practices could improve the execution and relevance of software engineering research on FLOSS.

In the past two decades, the FLOSS ecosystem has diversified its forms of development. Also, the development of the Linux kernel undergone notable transformations. At the GLR's screening stage for the Linux case, we already found aspects that diverge from an anarchical, bazaar-like development and converge to dictatorship or even feudalism. These include the existence of a leader with abusive and disrespectful behaviors, a pyramidal structure, an upper cadre, the allotment of the source code to small teams of maintainers, and work performed mainly by paid developers. Besides, we observed issues that, to the best of our knowledge, are little discussed in traditional literature. First, topics related to code maintenance organization: models of maintainership, the hierarchy of permissions, as well as problems like maintainer's scalability, such as signs of stress, burn-out, fears, and paralysis in the face of innovations. Second, the community subsistence: concerns about the difficulty in transferring knowledge that results in community aging and missing new voluntary contributors; companies developing new contributors in-house; and the toxic environment. Finally, topics related to the continuous growth of the source code, the non-regression rule, and the code change authorship concentrated in top contributors that are usually maintainers and employees of large companies. All these issues deserve further scrutiny by the research community.

This research was supported by CNPq proc. 465446/2014-0 and 163525/2018-8 and FAPESP proc. 15/24485-9.

REFERENCES

[1] R. J. Adams, P. Smart, and A. S. Huff. 2017. Shades of Grey: Guidelines for Working with the Grey Literature in Systematic Reviews for Management and Organizational Studies. *International Journal of Management Reviews* 19, 4 (2017).
[2] J. Bailey, D. Budgen, M. Turner, B. Kitchenham, P. Brereton, and S. Linkman. 2007. Evidence relating to Object-Oriented software design: A survey. In *First International Symposium on Empirical Software Engineering and Measurement.*
[3] J. C. de A. Biolchini, P. G. Mian, A. C. C. Natali, T. U. Conte, and G. H. Travassos. 2007. Scientific research ontology to support systematic review in software engineering. *Advanced Engineering Informatics* 21, 2 (2007), 133–151.
[4] B. B. N. de França, H. Jeronimo Junior, and G. H. Travassos. 2016. Characterizing DevOps by Hearing Multiple Voices. In *Proceedings of the 30th Brazilian Symposium on Software Engineering (SBES '16).* ACM, 53–62.
[5] V. Garousi, M. Felderer, and M. V. Mäntylä. 2016. The Need for Multivocal Literature Reviews in Software Engineering: Complementing Systematic Literature Reviews with Grey Literature. In *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering.*
[6] V. Garousi, M. Felderer, and M. V. Mäntylä. 2018. Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. *Information and Software Technology* (2018).
[7] K. Godin, J. Stapleton, S. I. Kirkpatrick, R. M. Hanning, and S. T. Leatherdale. 2015. Applying systematic review search methods to the grey literature: a case study examining guidelines for school-based breakfast programs in Canada. *Systematic Reviews* (2015).
[8] Julian P. T. Higgins and James Thomas (Eds.). 2019. *Cochrane handbook for systematic reviews of interventions* (2nd ed.). Wiley-Blackwell.
[9] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman. 2009. Systematic Literature Reviews in Software Engineering - A Systematic Literature Review. *Inf. Softw. Technol.* 51, 1 (2009), 7–15.
[10] Arsenio Paez. 2017. Gray literature: An important resource in systematic reviews. *Journal of Evidence-Based Medicine* 10, 3 (2017), 233–240.
[11] Eric Raymond. 1999. The cathedral and the bazaar. *Knowledge, Technology and Policy* 12, 3 (1999), 23–49.
[12] Hannah. Rothstein, A. J. Sutton, and Michael. Borenstein. 2005. *Publication bias in meta-analysis : prevention, assessment and adjustments.* Wiley.
[13] Jacopo Soldani, Damian Andrew Tamburri, and Willem-Jan Van Den Heuvel. 2018. The pains and gains of microservices: A Systematic grey literature review. *Journal of Systems and Software* 146 (2018), 215–232.