Topology of the Documentation Landscape

Marco Raglianti
marco.raglianti@usi.ch
REVEAL @ Software Institute – USI, Lugano, Switzerland

ABSTRACT

Every software system (ideally) comes with one or more forms of documentation. Beside source code comments, other structured and unstructured sources (*e.g.,* design documents, API references, wikis, usage examples, tutorials) constitute critical assets. Cloud-based repositories for collaborative development (*e.g.,* GitHub, Bitbucket, GitLab) provide many functionalities to create, persist, and version documentation artifacts. On the other hand, the last decade has seen the rise of rich instant messaging clients used as global software community platforms (*e.g.,* Slack, Discord). Although completely detached from a specific versioning system or development workflow, they allow developers to discuss implementation issues, report bugs, and, in general, interact with one another.

We refer to this evolving heterogeneous collection of information sources and documentation artifacts as the *documentation landscape*. It is important to have tools to extract information from these sources and integrate them in a topological visualization, to ease comprehension of a software system. How can we automatically generate this topology? How can we link elements in the topology back to the source code they refer to?

The goal of this PhD research is to automatically mine the documentation landscape of a system by disclosing pieces of information to aid, for example, in program maintenance tasks. We present our classification of possible documentation sources. The long term vision is to provide a domain model of the documentation landscape to build, visualize, and explore its instances for real software systems and evaluate the usefulness of the metaphor we propose.

CCS CONCEPTS

• **Software and its engineering → Collaboration in software development**; • **Information systems** → *Internet communications tools*; Data mining.

KEYWORDS

software documentation, communication platforms, visualization

ACM Reference Format:
Marco Raglianti. 2022. Topology of the Documentation Landscape. In *44th International Conference on Software Engineering Companion (ICSE '22 Companion), May 21–29, 2022, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3510454.3517068

1 INTRODUCTION

Software documentation is critical in several development activities [2]. Development tasks can be extracted and linked to documentation sections to provide on-demand support for those tasks and ease documentation navigation for developers [38]. Software documentation from heterogeneous sources (mailing lists, StackOverflow, issues, and pull requests) has been investigated to produce a taxonomy of possible issues [3]. Some of these sources are non-authoritative. More or less knowledgable users on StackOverflow can contribute in a Q&A format where other users can vote on the quality of answers. Popular frameworks and languages can generate a lot of non-authoritative documentation that can be exploited, for example, by recommender systems [28]. Also in Agile contexts, where documentation efforts are traditionally kept to a minimum, there is a perceived need for specific forms of activity supporting documents [27]. We refer to this faceted and heterogeneous multitude of possible sources as the **documentation landscape** of a software system. Although a complete list of contributions related to software documentation is out of scope, see the survey by Zhi *et al.* for 69 papers summarizing costs, benefits, and quality of software development documentation [43].

Correct and up to date documentation is useful [7, 9, 13, 14, 31, 34] but these attributes are often lacking [30, 31]. The problem of coevolution with code [3, 39, 40] has been increasing with the expansion of the documentation landscape. New sources emerged in the last two decades. A few examples are developers' blog posts [24], software engineers' microblogging on Twitter [37], rich media instant messaging applications [11, 21, 26, 33], news aggregators [4], and feature-rich forums [16]. By mining these sources it is possible to complement and fix traditional documentation [6, 28].

Crowd-curated documentation [25] shifted the ratio between documentation producers and consumers, but the coevolution problem remains relevant [40]. Cloud-based repositories support collaboration via tools tightly coupled with the repository itself [8, 36]. In the context of GitHub projects, although Issues have been investigated [5, 12, 15, 17, 18], the amount and nature of links to other communication platforms is still unclear and fairly unexplored.

A new perspective on software documentation comes in the form of *on-demand developer documentation* [32]. The idea is to produce, on-demand, just enough documentation to fulfill the necessary task. Such documentation can be custom-tailored for the task at hand and more closely align producers' efforts with consumers' needs. Automatic generation techniques try to reduce the gap between documentation and code [1, 22, 23]. Some of the shortcomings identified by Aghajani *et al.* in software documentation [3] (*e.g.,* inconsistency, outdatedness) can be partially mitigated or completely overcome by automatic derivation of documentation via program analysis [41], source code summarization [19, 42], method level documentation [22], and automatic comment updating [20]. This is also part of the documentation landscape.

2 RESEARCH FOCUS

The main goal of our research is to automatically extract heterogeneous documentation from different sources in the documentation landscape of a software system (Fig. 1). We envision the application of visualization techniques, in particular to *large* systems, to obtain a topological representation supporting program comprehension and maintenance tasks.

Our research tries to address the following questions:

- How many different documentation sources are there and how are they characterized?
- How can we automatically mine a software project to generate a topology of those sources?
- Which characterizing metrics can be mapped to visual attributes for different program comprehension and maintenance tasks?

Finally, we also aim at linking the topology back to source code artifacts. An interactive visualization of the documentation landscape will allow switching between bottom-up (from source code to architecture and high-level concepts) and top-down (from domain-based hypotheses to code) approaches to program comprehension [35] thus better supporting an opportunistic strategy. Our hypothesis is that a topological representation of documentation sources and artifacts can contribute to an efficient navigation of the documentation landscape in specific tasks. Controlled experiments in the so far outlined framework, with the support of the proposed tools, should bring empirical evidence of how it can benefit multiple stakeholders (*e.g.,* developers to support everyday activities, team leaders and project managers for planning and progress status assessment).

3 DOCUMENTATION LANDSCAPE

In the literature there is lack of a systematic approach to classification and integration of multiple documentation sources. We identified thirteen possible sources that might be relevant and classified them according to their nature – the *archetype* – and to different metrics for each archetype. There are four archetypes: Documents, Code, Multimedia, and Community. These surround the central point of source code in a version control system (see Fig. 1). For sources in each archetype, metrics are spanning the horizontal or vertical axis. For example in the *Community* archetype there are slower (*e.g.,* mailing lists) and faster (*e.g.,* instant messaging) sources as well as more volatile or more persistent ones. Finally, for each source, there are multiple possible instances. For example, there are many different instant messaging applications that are possible documentation sources (*e.g.,* Gitter, Discord, Slack). In a similar fashion, different types of documents can be related to a specific source depending on their origin or format (*i.e.,* requirements documents shared via Wiki, printed user manuals).

4 RESEARCH AGENDA

We initially explored Discord[1], an instance of the instant messaging source, for unstructured forms of documentation shared between Pharo[2] developers [29]. We are currently expanding our investigation to other sources in the Community archetype. Our main goal is to have a prototype to start validating the documentation landscape metaphor with developers. A possible scenario to analyze is the onboarding of new team members on projects in an already advanced state of development. We have four points in our agenda.

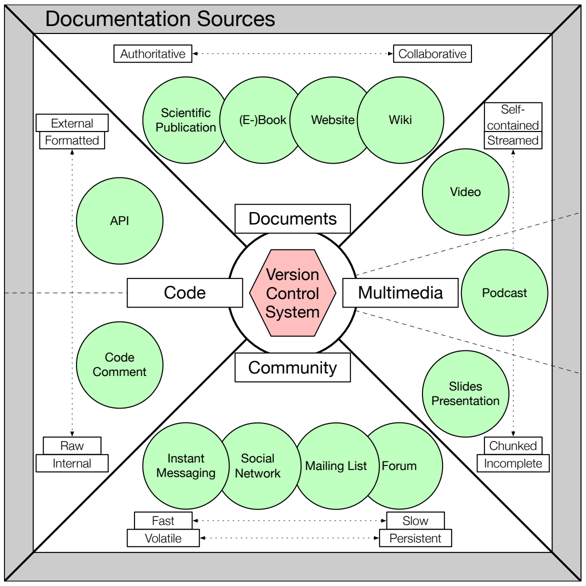

Figure 1: Documentation landscape of a software system

Visualizing the documentation landscape – Given a software system (*i.e.,* its GitHub repository), we will automatically generate an explorable visual representation of its documentation landscape. A topological map of the elements composing the landscape will enable us to evaluate the impact of this metaphor on software comprehension, maintenance, and re-engineering tasks.

From instant messaging to community – We need to refine our domain model of a Discord server and generalize it to other instant messaging instances. We will then consider covering other sources in the Community archetype as well.

Integrating multiple sources – We will include other archetypes to complete the picture of the landscape. For each source we plan to give a characterization in terms of metrics that might be relevant for comprehension tasks. We will also extend the domain model with the specific content of each source (*e.g.,* developers' Twitter accounts, video tutorial playlists).

Unstructured forms of documentation – We have tools to identify the source code that developers share via instant messaging [29]. We plan to aggregate the context of the surrounding discussions to understand why developers share this code [10]. Reconstructing these conversation streams poses both a theoretical and a technical challenge that we would like to address.

5 CONCLUSION

We introduced the context of our research by referring to software documentation, its importance, and its evolution in the recent years. Then, we motivated our work and specified its scope, focus, and intended goals. We presented our initial classification of archetypes and sources in the documentation landscape. Finally, we outlined possible directions and challenges for future research on this topic.

[1]See **https://discord.com** [accessed February 9, 2022]
[2]See **https://pharo.org** [accessed February 9, 2022]

REFERENCES

[1] Nahla Abid, Natalia Dragan, Michael L. Collard, and Jonathan I. Maletic. 2017. The Evaluation of an Approach for Automatic Generated Documentation. In *Proceedings of ICSME 2017 (International Conference on Software Maintenance and Evolution)*. IEEE, 307–317.

[2] Emad Aghajani, Csaba Nagy, Mario Linares-Vásquez, Laura Moreno, Gabriele Bavota, Michele Lanza, and David C. Shepherd. 2020. Software Documentation: The Practitioners' Perspective. In *Proceedings of ICSE 2020 (International Conference on Software Engineering)*. ACM, 590–601.

[3] Emad Aghajani, Csaba Nagy, Olga Lucero Vega-Márquez, Mario Linares-Vásquez, Laura Moreno, Gabriele Bavota, and Michele Lanza. 2019. Software Documentation Issues Unveiled. In *Proceedings of ICSE 2019 (International Conference on Software Engineering)*. IEEE/ACM, 1199–1210.

[4] Maurício Aniche, Christoph Treude, Igor Steinmacher, Igor Wiese, Gustavo Pinto, Margaret-Anne Storey, and Marco Aurélio Gerosa. 2018. How Modern News Aggregators Help Development Communities Shape and Share Knowledge. In *Proceedings of ICSE 2018 (International Conference on Software Engineering)*. ACM, 499–510.

[5] Tegawendé F. Bissyandé, David Lo, Lingxiao Jiang, Laurent Réveillère, Jacques Klein, and Yves Le Traon. 2013. Got issues? Who cares about it? A large scale investigation of issue trackers from GitHub. In *Proceedings of ISSRE 2013 (International Symposium on Software Reliability Engineering)*. IEEE, 188–197.

[6] Joshua Charles Campbell, Chenlei Zhang, Zhen Xu, Abram Hindle, and James Miller. 2013. Deficient Documentation Detection a Methodology to Locate Deficient Project Documentation Using Topic Analysis. In *Proceedings of MSR 2013 (Working Conference on Mining Software Repositories)*. IEEE, 57–60.

[7] Jie-Cherng Chen and Sun-Jen Huang. 2009. An Empirical Analysis of the Impact of Software Development Problem Factors on Software Maintainability. *Journal of Systems and Software* 82, 6 (2009), 981–992.

[8] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. 2012. Social Coding in GitHub: Transparency and Collaboration in an Open Software Repository. In *Proceedings of CSCW 2012 (Conference on Computer Supported Cooperative Work)*. ACM, 1277–1286.

[9] Barthélémy Dagenais and Martin P Robillard. 2010. Creating and Evolving Developer Documentation: Understanding the Decisions of Open Source Contributors. In *Proceedings of FSE 2010 (International Symposium on Foundations of Software Engineering)*. ACM, 127–136.

[10] Andrea Di Sorbo, Sebastiano Panichella, Corrado A. Visaggio, Massimiliano Di Penta, Gerardo Canfora, and Harald C. Gall. 2021. Exploiting Natural Language Structures in Software Informal Documentation. *IEEE Transactions on Software Engineering* 47, 8 (2021), 1587–1604.

[11] Osama Ehsan, Safwat Hassan, Mariam El Mezouar, and Ying Zou. 2020. An Empirical Study of Developer Discussions in the Gitter Platform. *Transactions on Software Engineering and Methodology* 30, 1 (2020), 1–39.

[12] Aron Fiechter, Roberto Minelli, Csaba Nagy, and Michele Lanza. 2021. Visualizing GitHub Issues. In *Proceedings of VISSOFT 2021 (Working Conference on Software Visualization)*. IEEE, 155–159.

[13] Andrew Forward and Timothy C Lethbridge. 2002. The Relevance of Software Documentation, Tools and Technologies: A Survey. In *Proceedings of DocEng 2002 (Symposium on Document Engineering)*. ACM, 26–33.

[14] Golara Garousi, Vahid Garousi-Yusifoğlu, Guenther Ruhe, Junji Zhi, Mahmoud Moussavi, and Brian Smith. 2015. Usage and Usefulness of Technical Software Documentation: An Industrial Case Study. *Information and Software Technology* 57 (2015), 664–682.

[15] Mehdi Golzadeh, Alexandre Decan, Damien Legay, and Tom Mens. 2021. A Ground-Truth Dataset and Classification Model for Detecting Bots in GitHub Issue and PR Comments. *Journal of Systems and Software* 175 (2021), 110911.

[16] Hideaki Hata, Nicole Novielli, Sebastian Baltes, Raula Gaikovina Kula, and Christoph Treude. 2021. GitHub Discussions: An Exploratory Study of Early Adoption. arXiv:2102.05230

[17] Rafael Kallis, Andrea Di Sorbo, Gerardo Canfora, and Sebastiano Panichella. 2021. Predicting Issue types on GitHub. *Science of Computer Programming* 205 (2021), 102598.

[18] Riivo Kikas, Marlon Dumas, and Dietmar Pfahl. 2016. Using Dynamic and Contextual Features to Predict Issue Lifetime in GitHub Projects. In *Proceedings of MSR 2016 (Working Conference on Mining Software Repositories)*. IEEE/ACM, 291–302.

[19] Alexander LeClair, Sakib Haque, Lingfei Wu, and Collin McMillan. 2020. Improved Code Summarization via a Graph Neural Network. In *Proceedings of ICPC 2020 (International Conference on Program Comprehension)*. ACM, 184–195.

[20] Bo Lin, Shangwen Wang, Kui Liu, Xiaoguang Mao, and Tegawendé F. Bissyandé. 2021. Automated Comment Update: How Far are We?. In *Proceedings of ICPC 2021 (International Conference on Program Comprehension)*. IEEE/ACM, 36–46.

[21] Bin Lin, Alexey Zagalsky, Margaret-Anne Storey, and Alexander Serebrenik. 2016. Why Developers Are Slacking Off: Understanding How Software Teams Use Slack. In *Proceedings of CSCW/SCC 2016 (Conference on Computer Supported Cooperative Work and Social Computing Companion)*. ACM, 333–336.

[22] Christian D. Newman, Natalia Dragan, Michael L. Collard, Jonathan I. Maletic, Michael J. Decker, Drew T. Guarnera, and Nahla Abid. 2018. Automatically Generating Natural Language Documentation for Methods. In *Proceedings of DysDoc 2018 (International Workshop on Dynamic Software Documentation)*. IEEE, 1–2.

[23] Jalves Nicacio and Fabio Petrillo. 2021. Towards Improving Architectural Diagram Consistency Using System Descriptors. In *Proceedings of ICPC 2021 (International Conference on Program Comprehension)*. IEEE/ACM, 401–405.

[24] Dennis Pagano and Walid Maalej. 2011. How Do Developers Blog? An Exploratory Study. In *Proceedings of MSR 2011 (Working Conference on Mining Software Repositories)*. ACM, 123–132.

[25] Chris Parnin, Christoph Treude, Lars Grammel, and Margaret-Anne Storey. 2012. *Crowd Documentation: Exploring the Coverage and the Dynamics of API Discussions on Stack Overflow*. Technical Report. Georgia Institute of Technology.

[26] Esteban Parra, Ashley Ellis, and Sonia Haiduc. 2020. GitterCom: A Dataset of Open Source Developer Communications in Gitter. In *Proceedings of MSR 2020 (International Conference on Mining Software Repositories)*. ACM, 563–567.

[27] Jirat Pasuksmit, Patanamon Thongtanunam, and Shanika Karunasekera. 2021. Towards Just-Enough Documentation for Agile Effort Estimation: What Information Should Be Documented?. In *Proceedings of ICSME 2021 (International Conference on Software Maintenance and Evolution)*. IEEE.

[28] Luca Ponzanelli, Gabriele Bavota, Massimiliano Di Penta, Rocco Oliveto, and Michele Lanza. 2014. Mining StackOverflow to Turn the IDE into a Self-Confident Programming Prompter. In *Proceedings of MSR 2014 (Working Conference on Mining Software Repositories)*. IEEE/ACM, 102–111.

[29] Marco Raglianti, Roberto Minelli, Csaba Nagy, and Michele Lanza. 2021. Visualizing Discord Servers. In *Proceedings of VISSOFT 2021 (Working Conference on Software Visualization)*. IEEE, 150–154.

[30] Martin P Robillard. 2009. What Makes APIs Hard to Learn? Answers from Developers. *IEEE Software* 26, 6 (2009), 27–34.

[31] Martin P Robillard and Robert DeLine. 2011. A Field Study of API Learning Obstacles. *Empirical Software Engineering* 16, 6 (2011), 703–732.

[32] Martin P. Robillard, Andrian Marcus, Christoph Treude, Gabriele Bavota, Oscar Chaparro, Neil Ernst, Marco Aurélio Gerosa, Michael Godfrey, Michele Lanza, Mario Linares-Vásquez, Gail C. Murphy, Laura Moreno, David Shepherd, and Edmund Wong. 2017. On-demand Developer Documentation. In *Proceedings of ICSME 2017 (International Conference on Software Maintenance and Evolution)*. IEEE, 479–483.

[33] Lin Shi, Xiao Chen, Ye Yang, Hanzhi Jiang, Ziyou Jiang, Nan Niu, and Qing Wang. 2021. A First Look at Developers' Live Chat on Gitter. In *Proceedings of ESEC/FSE 2021 (European Software Engineering Conference and Symposium on the Foundations of Software Engineering)*. ACM, 391–403.

[34] Ian Sommerville. 2015. *Software Engineering* (10th ed.). Pearson.

[35] Margaret-Anne Storey, David F. Fracchia, and Hausi A. Müller. 1999. Cognitive Design Elements to Support the Construction of a Mental Model During Software Exploration. *Journal of Systems and Software* 44, 3 (1999), 171–185.

[36] Jirateep Tantisuwankul, Yusuf Sulistyo Nugroho, Raula Gaikovina Kula, Hideaki Hata, Arnon Rungsawang, Pattara Leelaprute, and Kenichi Matsumoto. 2019. A topological analysis of communication channels for knowledge sharing in contemporary GitHub projects. *Journal of Systems and Software* 158 (2019), 110416.

[37] Yuan Tian, Palakorn Achananuparp, Ibrahim Nelman Lubis, David Lo, and Ee-Peng Lim. 2012. What does software engineering community microblog about?. In *Proceedings of MSR 2012 (Working Conference on Mining Software Repositories)*. IEEE, 247–250.

[38] Christoph Treude, Martin P. Robillard, and Barthélémy Dagenais. 2015. Extracting Development Tasks to Navigate Software Documentation. *IEEE Transactions on Software Engineering* 41, 6 (2015), 565–581.

[39] Gias Uddin and Martin P Robillard. 2015. How API Documentation Fails. *IEEE Software* 32, 4 (2015), 68–75.

[40] Fengcai Wen, Csaba Nagy, Gabriele Bavota, and Michele Lanza. 2019. A Large-Scale Empirical Study on Code-Comment Inconsistencies. In *Proceedings of ICPC 2019 (International Conference on Program Comprehension)*. IEEE/ACM, 53–64.

[41] Juan Zhai, Xiangzhe Xu, Yu Shi, Guanhong Tao, Minxue Pan, Shiqing Ma, Lei Xu, Weifeng Zhang, Lin Tan, and Xiangyu Zhang. 2020. CPC: Automatically Classifying and Propagating Natural Language Comments via Program Analysis. In *Proceedings of ICSE 2020 (International Conference on Software Engineering)*. ACM, 1359–1371.

[42] Jian Zhang, Xu Wang, Hongyu Zhang, Hailong Sun, and Xudong Liu. 2020. Retrieval-Based Neural Source Code Summarization. In *Proceedings of ICSE 2020 (International Conference on Software Engineering)*. ACM, 1385–1397.

[43] Junji Zhi, Vahid Garousi-Yusifoğlu, Bo Sun, Golara Garousi, Shawn Shahnewaz, and Guenther Ruhe. 2015. Cost, Benefits and Quality of Software Development Documentation: A Systematic Mapping. *Journal of Systems and Software* 99 (2015), 175–198.