Designing an Effective User Interface for Analyzing Software Repositories

Adam Tutko
Department of Electrical Engineering and Computer Science
University of Tennessee
Knoxville, Tennessee, US
atutko@vols.utk.edu

I. Introduction

The introduction of Git[1], a version control system, has radically changed the software development process. Currently, Git is considered an essential skill for software engineering developers and is necessary for collaboration between developers [1]. This reliance on Git means that software development creates a vast pool of publicly available data. This data provides an opportunity for software engineering researchers interested in analyzing the software development process. Such analysis could facilitate numerous tools for developers that substantially increase developer productivity.

This opportunity has become apparent to researchers and the field is growing rapidly. However, analysis is riddled with difficulties because starting data extraction is a non-trivial matter. Mining Software Repository researchers are limited to crawling the Git ecosystem manually or leveraging APIs for data extraction. Unfortunately, these API are often difficult to use [3], [4] and might not be robust enough to retrieve the desired information. Alongside this, sampling is frequently done by manually selecting projects (e.g., [5], [6], [8], [9]), which may be biased, and this practice has the potential to introduce data integrity problems [7].

To overcome these issues, my goal is to design and implement a user interface that enables software engineering researchers to effectively analyze code repositories. This interface will aim to address the complications other API currently face. The interface will allow rapid retrieval of a vast amount of Git data and will specifically target ease of use issues that have been noted for other API. It will be built to retrieve the desired data based on minimal and easily provided input. Thus, removing the need for users to expend effort understanding the system before use.

II. Progress to Date

My initial work has been in engineering the collaborative project, World of Code (WoC) [2], a system intended to overcome the issues with retrieving data from code repositories. WoC allows for quick data retrieval of relevant data by archiving public Git repositories and providing an API for interacting with the data. To date, WoC contains over 100 million projects and nearly 40 million authors.

[1]https://git-scm.com/

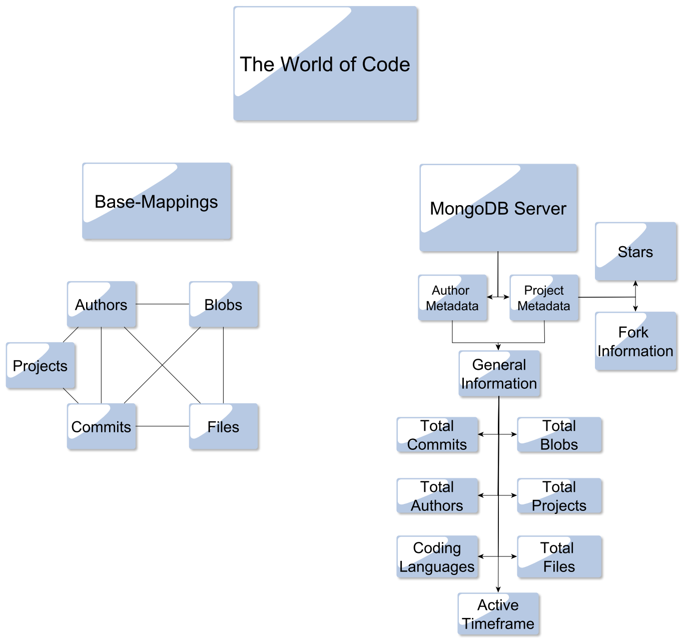

Fig. 1. System Layout for resources within the World of Code

The data is randomly stored in base-mappings by using their SHA1 hash. The World of Code provides the potential to perform various samplings from a massive set of Git repositories. Furthermore, users can filter this data extensively by including basic boolean restrictions on the query.

However, while the World of Code is a potentially exceptional resource for analyzing Git, the system has a number of roadblocks for users unfamiliar with it. The World of Code currently lacks an easily used user interface. To leverage the system, users must work with the base-mappings between Git data. Unfortunately, this often requires users to write scripts meant to retrieve the targeted data.

Towards overcoming this usability issue of WoC, I created a set of metadata that is commonly used by software engineering researchers and designed an API to interact with the data. This data was stored on a MongoDB server for easy retrieval and analysis. These metadata collections include basic information on Git projects and authors.

978-1-7281-6901-9/20/$31.00 © 2020 IEEE

For projects, the metadata includes how long the project was active, the coding languages used, if the project is a fork of another project, the star rating on GitHub (if applicable), and the number of authors, commits, and files in the project. The star rating was calculated by cross-referencing information from WoC and GitHub.

Similarly, the author metadata includes information on how long the author was active, the coding languages used, and the number of commits, files, and projects related to that author.

The metadata system provides a more easily used, but less powerful, option for researchers when analyzing projects and authors. Primarily, it offers researchers potentially useful data without requiring a working knowledge of how to communicate with the World of Code. This system, and the compiled datasets within, could help mitigate the difficulty with starting analysis.

III. System Usage

The World of Code system boasts many advantages as a resource for Git data retrieval, but the system still has potential for improvements regarding accessibility and usability.

Currently, there are a number of options for communication with the World of Code, but they require researchers to be proficient in programming and have extensive knowledge of how data is stored in Git. The languages currently integrated with WoC are Python, Perl, and the Unix Command Line. Thus, to be able perform personal queries on the system, having an understanding of one or more of these languages is necessary. On top of this, a basic understanding of how the World of Code is formatted is necessary.

The metadata collections were compiled to mitigate these issues. However, these datasets are currently only available within the World of Code. Researchers interested in the metadata collections must request access to the World of Code in order to gain access to the MongoDB server hosting the metadata.

Our team is considering how to address the accessibility issues regarding the metadata datasets. I am currently designing an online resource that will allow direct access to the metadata collections. Interested users will be required to sign up to a website and then be allowed to query the MongoDB collections containing the information. This interface will be provided on top of a website related to the World of Code already in development.

IV. Future Work

My primary goal moving forward will be to address issues with gaining access to and using these resources. I plan to continue work on the World of Code interface to further simplify it's use and greatly increase the productivity of it's users. Providing a simpler user interface for communicating with the World of Code will be a big step in making the resource accessible. Currently, I can see a few options for improving the World of Code to consider. One course of action is to provide users a graphical user interface for WoC that can perform basic queries.

The interface would be built as an optional extension for the World of Code. The GUI would be primarily designed for users not confident in their programming abilities by allowing simple queries of the system. In tandem with the GUI, integrating other popular languages as further communication methods for WoC could greatly increase the pool of potential users. By providing these two options to users, the difficulties with using the system would be addressed.

Another potential course of action would be to survey researchers who had previously analyzed Git repositories. This survey would try to determine what data they believe would be useful in further metadata datasets. Questions would be focused on determining what data they targeted in their previous research and if there was information they desired but were unable to gather. This information would allow us to build upon the currently available metadata collections by providing more relevant and interesting data in future versions.

Discovering the necessary improvements to the World of Code will require no small amount of research and consideration. However, these future improvements have the potential to make this resource a virtual goldmine when analyzing Git repositories. This interface will introduce researchers to a system that eclipses other Git data retrieval API. This will exponentially increase the number of useful tools available to software developers. Such an exponential increase in tools could propel the modern software design process to an entirely new level.

References

[1] Frans F Blauw. The use of git as version control in the south african software engineering classroom. In *2018 IST-Africa Week Conference (IST-Africa)*, pages Page–1. IEEE, 2018.

[2] Georgios Gousios and Diomidis Spinellis. Mining software engineering data from github. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, pages 501–502. IEEE, 2017.

[3] Dimitris Kolovos, Patrick Neubauer, Konstantinos Barmpis, Nicholas Matragkas, and Richard Paige. Crossflow: a framework for distributed mining of software repositories. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, pages 155–159. IEEE, 2019.

[4] Yasutaka Sakamoto, Shinsuke Matsumoto, and Masahide Nakamura. Integrating service oriented msr framework and google chart tools for visualizing software evolution. In *2012 Fourth International Workshop on Empirical Software Engineering in Practice*, pages 35–39. IEEE, 2012.

[5] A. C. Short and A. Z. Henley. Towards an empirically-based ide: An analysis of code size and screen space. In *2019 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 199–203, 2019.

[6] Danilo Silva and Marco Tulio Valente. Refdiff: detecting refactorings in version histories. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, pages 269–279. IEEE, 2017.

[7] Fabian Trautsch, Steffen Herbold, Philip Makedonski, and Jens Grabowski. Adressing problems with external validity of repository mining studies through a smart data platform. In *Proceedings of the 13th International Conference on Mining Software Repositories*, pages 97–108, 2016.

[8] Xin Yang, Raula Gaikovina Kula, Norihiro Yoshida, and Hajimu Iida. Mining the modern code review repositories: A dataset of people, process and product. In *Proceedings of the 13th International Conference on Mining Software Repositories*, pages 460–463, 2016.

[9] Yue Yu, Huaimin Wang, Vladimir Filkov, Premkumar Devanbu, and Bogdan Vasilescu. Wait for it: Determinants of pull request evaluation latency on github. In *2015 IEEE/ACM 12th working conference on mining software repositories*, pages 367–371. IEEE, 2015.