CAM: A Collection of Snapshots of GitHub Java Repositories Together with Metrics

Yegor Bugayenko
yegor256@huawei.com
Huawei, Russia, Moscow

Abstract

Even though numerous researchers require stable datasets along with source code and basic metrics calculated on them, neither GitHub nor any other code hosting platform provides such a resource. Consequently, each researcher must download their own data, compute the necessary metrics, and then publish the dataset somewhere to ensure it remains accessible indefinitely. Our **CAM** (stands for "Classes and Metrics") project addresses this need. It is an open-source software capable of cloning Java repositories from GitHub, filtering out unnecessary files, parsing Java classes, and computing metrics such as Cyclomatic Complexity, Halstead Effort and Volume, C&K metrics, Maintainability Metrics, LCOM5 and HND, as well as some Git-based Metrics. At least once a year, we execute the entire script, a process which requires a minimum of ten days on a very powerful server, to generate a new dataset. Subsequently, we publish it on Amazon S3, thereby ensuring its availability as a reference for researchers. The latest archive of 2.2Gb that we published on the 2nd of March, 2024 includes 532K Java classes with 48 metrics for each class.

1 Motivation

First, research projects that analyze Java code usually extract it from repositories where open-source projects store their files, such as GitHub. It is common practice in papers explaining results to fully disclose the coordinates of the open-source code being extracted. However, source code is inherently volatile: repositories change their locations and files are modified, as demonstrated by Robles [10]. To ensure the replicability of their research results, paper authors must somehow guarantee that the source code used at the time of research remains available and intact throughout the paper's lifetime. One obvious solution would be to make copies of the repositories being extracted and then host them somewhere they are "forever" available.

Second, research methods typically involve filtering out certain types of files found in repositories, such as plain text documents or graphic images, which are not source code. Additionally, some source code files may need to be excluded because they are auto-generated or contain unparseable Java code, making them unsuitable for most methods of code analysis.

Third, most source code analysis research involves collecting metrics from the files found in extracted repositories, such as lines of code, complexity, cohesion, and so on. Most of these metrics are already known, and their retrieval mechanisms are trivial, as summarized by Nuñez-Varela et al. [9].

Thus, there is an obvious duplication of work among different research projects: (a) they have to "host" extracted data to ensure desirable replicability, as noted by Cosentino et al. [3], (b) they must implement filtering of source code fetched from GitHub, and (c) they have to collect popular metrics. Having a ready-to-use archive of downloaded, filtered, and measured source code files would help many research projects reduce the amount of work required.

2 Methodology

In order to help research projects in all three tasks mentioned above, we created **CAM**[1] archive: an open-source collection of scripts regularly (at least once a year) being executed in Docker containers in our proprietary computing environment with results published in form of an "immutable" ZIP archive as either a GitHub "asset" attached to the next release of our GitHub repository or an object in

[1] https://github.com/yegor256/cam

Amazon S3 (depending on the size of the archive). Here, immutability is not technically guaranteed but promised: even though we, being the owners of the repository, are able to replace any previously created assets, we are not going to do so in order to not jeopardize the idea. Instead, new releases will be published retaining previously generated assets unmodified.

At the time of writing, our GitHub repository consists of scripts written in Makefile, Python, Ruby, and Bash, which do exactly the following:

- Fetch 1000 repositories from GitHub, which have `java` language tag, have more than 1K and less than 10K stars, and are at least as big as 200Kb;
- Remove files without `.java` extension, Java files with syntax errors, `package-info.java` files, `module-info.java` files, files with lines longer than 1024 characters, and unit tests;
- Calculate KLoC, NCSS, Cyclomatic Complexity [8], Cognitive Complexity [2], LCOM5 [7], NHD [4], TCC [1], number of attributes, number of constructors, number of methods, number of static methods, and some other metrics.

The size of the latest archive generated on the 2nd of March, 2024 is 2.2Gb. It includes 532K Java classes with 48 metrics for each class. It took us 10 days on a server with eight vCPU and 32Gb of RAM to generate the data.

3 Limitations

As of January 2023, Dohmke [6] reported that GitHub hosts more than 420 million repositories, including at least 28 million public repositories, which is the world's largest source code host as of June 2023. According to [5], Java is the 4th most popular language on GitHub. Thus, it is reasonable to assume that there are millions of Java repositories on GitHub. It is technically impossible to download and parse even a few percent of this huge data source. In the **CAM** project, we download and scan only a thousand repositories (planning to download a few thousand in the future). Such a tiny fraction of the entire possible scope of analysis is obviously not representative enough. Researchers must understand this limitation and only use **CAM** when representability of the entire Java domain is not the goal of the research.

Even though most of the metrics that we collect have formal definitions given in the papers where the metrics were originally introduced, for example NHD [4] and TCC [1], there are certain modifications that we had to make to their original algorithms. This happened mostly because modern Java classes have certain features that were not present when said metrics were introduced. Researchers must understand that the metrics generated by the scripts in **CAM** are not exactly the same metrics that were described by their authors.

Even though our scripts download only reasonably popular Java repositories, some of them contain Java files with broken syntax. Also, some files use new Java syntax introduced only in recent versions of Java (such as, for example, "records" introduced in Java 21). The parser[2] that we use in **CAM** is only capable of parsing Java 8. We simply exclude all files that are not parseable by this parser. Researchers who are looking for the most current syntax of Java must remember this limitation and try to find another source of data.

4 Conclusion

We expect **CAM** archives to be used by research teams analyzing Java source, which want (a) to guarantee replicability of their results and (b) to reduce data pre-processing efforts. We also expect open-source community to contribute to **CAM** scripts, making filtering more powerful and adding more code metrics to the collection.

References

[1] James M. Bieman and Byung-Kyoo Kang. 1995. Cohesion and Reuse in an Object-Oriented System. *SIGSOFT Software Engineering Notes* 20, SI (1995), 259–262. https://doi.org/10.1145/223427.211856

[2] G. Ann Campbell. 2018. Cognitive Complexity: An Overview and Evaluation. In *Proceedings of the International Conference on Technical Debt*. 57–58. https://doi.org/10.1145/3194164.3194186

[3] Valerio Cosentino, Javier L. Cánovas Izquierdo, and Jordi Cabot. 2017. A Systematic Mapping Study of Software Development With GitHub. *IEEE Access* 5 (2017), 7173–7192. https://doi.org/10.1109/ACCESS.2017.2682323

[4] Steve Counsell, Stephen Swift, and Jason Crampton. 2006. The Interpretation and Utility of Three Cohesion Metrics for Object-Oriented Design. *ACM Transactions on Software Engineering and Methodology (TOSEM)*

[2] https://github.com/c2nes/javalang

15, 2 (2006), 123–149. https://doi.org/10.1145/1131421.1131422

[5] Kyle Daigle. 2023. Octoverse: The state of open source and rise of AI in 2023. https://github.blog/2023-11-08-the-state-of-open-source-and-ai/. [Online; accessed 13-03-2024].

[6] Thomas Dohmke. 2023. 100 million developers and counting. https://github.blog/2023-01-25-100-million-developers-and-counting/. [Online; accessed 13-03-2024].

[7] Brian Henderson-Sellers, Larry L. Constantine, and Ian M. Graham. 1996. Coupling and Cohesion (Towards a Valid Metrics Suite for Object-Oriented Analysis and Design). *Object Oriented Systems* 3, 3 (1996), 143–158.

[8] Thomas J. McCabe. 1976. A Complexity Measure. *IEEE Transactions on Software Engineering* 4 (1976), 308–320. https://doi.org/10.1109/TSE.1976.233837

[9] Alberto S. Nuñez-Varela, Héctor G. Pérez-Gonzalez, Francisco E. Martínez-Perez, and Carlos Soubervielle-Montalvo. 2017. Source Code Metrics: A Systematic Mapping Study. *Journal of Systems and Software* 128 (2017), 164–197. https://doi.org/10.1016/j.jss.2017.03.044

[10] Gregorio Robles. 2010. Replicating MSR: A Study of the Potential Replicability of Papers Published in the Mining Software Repositories Proceedings. In *IEEE Working Conference on Mining Software Repositories*. 171–180. https://doi.org/10.1109/MSR.2010.5463348