2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR) | 978-1-7281-8710-5/20/$31.00 ©2021 IEEE | DOI: 10.1109/MSR52588.2021.00080

QScored: A Large Dataset of Code Smells and Quality Metrics

Tushar Sharma
Siemens Technology
Charlotte, USA
tusharsharma@ieee.org

Marouane Kessentini
University of Michigan
MI, USA
marouane@umich.edu

***Abstract*—Code quality aspects such as code smells and code quality metrics are widely used in exploratory and empirical software engineering research. In such studies, researchers spend a substantial amount of time and effort to not only select the appropriate subject systems but also to analyze them to collect the required code quality information. In this paper, we present QScored dataset; the dataset contains code quality information of more than 86 thousand C# and Java GitHub repositories containing more than 1.1 billion lines of code. The code quality information contains seven kinds of detected architecture smells, 20 kinds of design smells, eleven kinds of implementation smells, and 27 commonly used code quality metrics computed at project, package, class, and method levels. Availability of the dataset will facilitate empirical studies involving code quality aspects by making the information readily available for a large number of active GitHub repositories.**

***Index Terms*—Code quality, code smells, quality metrics, maintainability, technical debt.**

I. Introduction

Maintainability is an important aspect of software quality. A well-maintained software system is relatively easy to extend and correct [1]. The software engineering community considers code smells [1], [2] and code quality metrics [3] as the common mechanisms to identify issues that impact maintainability of a software system. The community has carried out extensive explorations proposing various ways to detect code smells as well as to investigate their causes and impacts [4].

Repository mining studies nowadays heavily depend on the source code repositories that are available on code repository hosting platform such as GitHub. The abundance of open-source repositories, on the one hand, offers opportunities to researchers to carry out mining studies on a relatively large scale; on the other hand, poses a challenge to select a subset of them, especially based on code quality. Software engineering researchers often require a set of high-quality repositories to carry out their empirical experiments. Most of the times, they select repositories based on their size, popularity (*e.g.,* number of stars), or activity (*i.e.,* the number of commits) as the options to choose repositories based on their quality is not available. Though there have been some attempts, for instance RepoReapers [5] and PHANTOM [6], to address the gap, these attempts do not consider detailed code quality measures for their repository evaluation. In addition, many empirical studies investigate relationship of code quality aspects with other software artifact or process aspects (for example, bugs prediction [7]–[10] and maintainability prediction [11]–[13]). Furthermore, once they select the repositories, the code quality information such as code smells and metrics are not readily available; they have to choose an appropriate tool, and run the tool on the selected subject systems to collect the required code quality information. For a large number of repositories, the activity demands computing resources as well as researchers' time and effort.

In this paper, we present *QScored* dataset to address the issues outlined above. The dataset offers the following information for 86,652 repositories written mainly in Java and C# having more than 1.1 billion lines of code from GitHub.

- The dataset contains a comprehensive set of detected code smells (seven kinds of architecture, 20 design, and eleven implementation smells) for each repository.
- It consists of ten project-level (such as *smell density* and *code duplication*), two package-level (*lines of code* and *smell density*), 12 class-level (such as *lack of cohesion of methods* and *lines of code*), and three method-level (such as *cyclomatic complexity*) commonly used code quality metrics.
- Historical trend of relative ranking of the repositories based on their code quality.

II. Dataset Construction

Figure 1 presents the architectural overview of construction method adopted for this dataset. QScored agent (or client) implements *repository selector*, *repository downloader*, *repository analyzer*, and *report uploader*. QScored server implements a set of REST APIs [14] to enable clients to interact with the server in addition to components that realize code analysis report validation and quality score computation. We elaborate the individual steps below.

A. Repository selector

We adopt a selection criterion to choose a subset of GitHub repositories. We select repositories written either in Java or C# programming languages. Also, we only select active repositories, *i.e.,* repositories that are modified at least once in the last one year. We implement a GraphQL script to specify

978-1-7281-8710-5/21/$31.00 ©2021 IEEE
DOI 10.1109/MSR52588.2021.00080

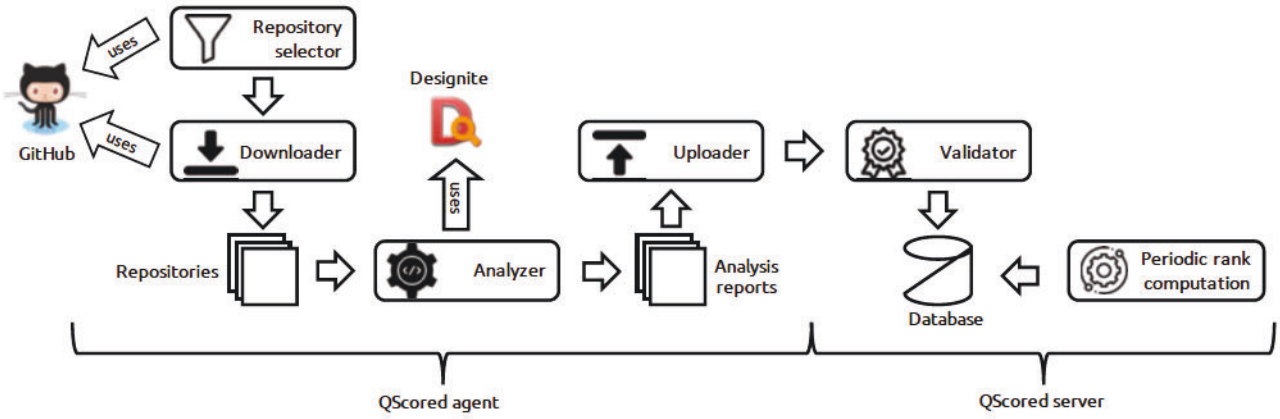

Fig. 1. Overview of the dataset construction method

the criterion and document the identified repositories in a CSV file.

B. Repository downloader

This module gets the list of repositories to download from the repository selector module and downloads the repositories.

C. Repository analyzer

The analyzer module takes one downloaded repository at a time and analyzes it using Designite [15] (for C# repositories) or DesigniteJava [16] (for Java repositories). These tools provide a detailed code analysis report. Each such report contains detected architecture, design, and implementation smells as well as commonly used code quality metrics. Designite detects seven kinds of architecture, 20 kinds of design, and eleven types of implementation smells. Sharma et al. [17] provide a detailed description of smells as well as the tool validation for Designite.

D. Report uploader

The report uploader module checks the size of the analyzed repository in the generated code analysis report. If the size of the repository is less than 1,000 lines of code, the module discard the repository to avoid populating the dataset with toy programs; otherwise, the module uploads the analyzed report to the QScored server along with metadata of the analysis (such as project name and URL of the repository). QScored provides a secure *upload* API by using API key that can be obtained for free from the QScored platform.

E. Report validator

QScored defines a portable XML schema to include detailed code quality information so that the visualization can be made tool and language independent. The schema can be accessed online.[1] Each code analysis report must confirm to this XML schema. Each incoming code analysis report is checked by the report validator module to ensure that the report adheres to the schema. Once the code analysis report passes the validation, the module parses the report and store the information in a PostgreSQL database.

[1]https://qscored.com/docs/extend_scope/

F. Database

Figure 2 presents the schema of the QScored database. Each analysis report creates a record in `solution` table. Each solution, in turn, may have multiple projects (especially for C# repositories). Code smell and metric records for each source code entity are stored in corresponding tables. Quality score and rank information is stored in `rank_score` table. Table I presents key size metrics of the database.

Source code entity/smell	Java	C#
#Repositories	55,284	31,368
LOC	758,940,537	385,127,027
#Components	1,389,672	914,830
#Classes	10,039,117	4,208,243
#Methods	73,972,917	16,072,342
#Architecture smells	855,793	341,697
#Design smells	8,405,248	4,178,885
#Implementation smells	45,777,230	28,597,401

TABLE I
KEY SIZE METRICS FOR THE DATASET

G. Quality score and rank computation

For each repository in its corpus, QScored computes a quality score based on the detected architecture, design, and implementation smells. Furthermore, a *quality rank* is computed periodically (weekly) based on the quality score. A lower value of quality score is desired and hence project with the smallest quality score is assigned the highest quality rank. We encourage the reader to refer to our previous work [14] that provides more details about computing quality score and ranking.

III. Availability and Usage

This section outlines different ways by in which the database can be explored and used. We also provide examples to help the reader better understand the database.

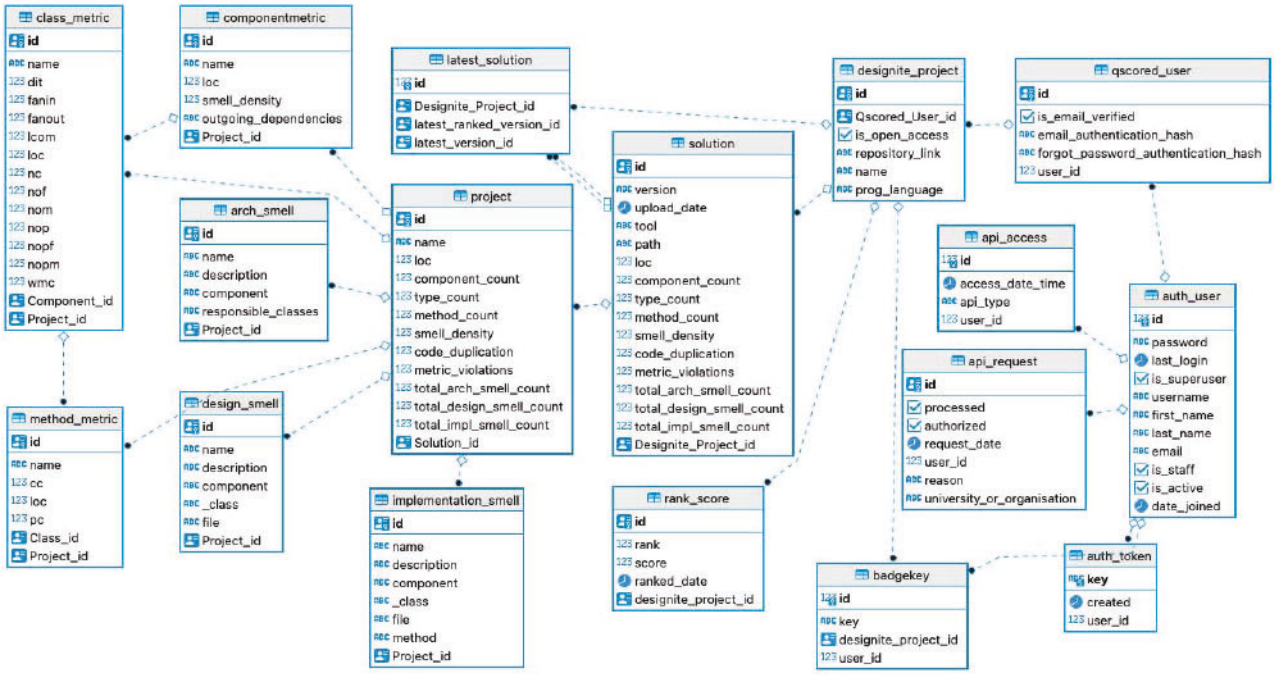

Fig. 2. Schema of the QScored database

A. Availability of dataset

1) Database dump: The database dump is made available publicly to download [18]. Size of the archived (in *tar* format) database is approximately 38 GB.

2) Dataset exploration using search APIs: Apart from the entire database dump, we offer an alternative way to explore the dataset in the form of QScored *search* APIs. The first search API `search_project_by_quality` allows users to search repositories based on their quality ranking in the dataset. The API allows users to specify filters *language* and LOC *range* to obtain a list of repositories that satisfy the criteria in the sorted order of their quality ranking. Another API `search_project` returns the metadata about the searched projects. The API takes *project name* and *repository link* as inputs. Both of the search APIs returns a list of project metadata containing project name and unique id, repository link, LOC, programming language, as well as quality rank and score. We provide examples of the above-mentioned APIs online.[2]

3) Dataset search using QScored platform: The dataset can also be explored via a web interface provided by the QScored platform.[3] The interface provides options to search the dataset based on repository name, programming language, size of the repository (in LOC), and quality score as well as quality rank.

[2]https://www.qscored.com/docs/api
[3]https://qscored.com/ranking/

B. Analysis scripts

The scripts to select repositories using GraphQL, to download them from GitHub, to analyze them using Designite and DesigniteJava, and to upload the code analysis reports to QScored server has been made available publicly [19].

C. Examples

The following SQL script provides a list of Java repository ranked in top 10 repositories in the dataset based on the latest ranking.

```
SELECT distinct designite_project.name
FROM designite_project, rank_score
WHERE designite_project.id =
rank_score.designite_project_id AND
rank_score.rank<=10 AND
prog_language ='java' AND
rank_score.ranked_date = (
SELECT MAX(rank_core.ranked_date)
FROM rank_score)
```

Similarly, the following example explores the difference in quality score of the repositories over time. The script emits five records; it shows that there are five repositories for which the minimum quality score is less than 2 and the maximum score is greater than 10 since the creation of the dataset. It must be noted that the dataset also contains total 13,613 versions of the dataset repositories; these versions are added periodically only for those repositories that have been changed since the

last analysis. The result highlights the evolution of the quality profile of these five repositories.

```
SELECT count(id), designite_project_id
FROM rank_score
GROUP BY designite_project_id
HAVING MAX(score)>10 AND MIN(score)<2
```

D. Privacy concerns

As we elaborate in our previous work [14], anybody can use the platform to upload their code analysis reports. The platform allows the users to mark a project private. To avoid privacy concerns, we remove projects uploaded by all the users apart from the authors of this paper and provide a cleaned dataset for public use. Similarly, we removed all usernames, email addresses, and API keys to honor the users' privacy.

IV. Impact and Potential Research Directions

Code smells are used extensively to carry out a wide range of exploratory and empirical studies. It includes the effect of code smells on bug prediction [7], on maintainability prediction [11], on maintenance effort [12], and on merge conflicts [20]. Similarly, code quality metrics are used widely for bug prediction [8]–[10], and maintainability prediction [4], [13]. To carry out such experiments, researchers spent a substantial amount of time and effort to not only select the appropriate subject systems but also analyze them to collect the required information. This dataset offers help to the software engineering researchers by reducing their effort in collecting code quality information. Specifically, we envision that the presented dataset could be relevant and useful in following applications.

A. Bug prediction

Software engineering researchers may utilize the offered dataset in software quality prediction studies including bug prediction and change prediction.

B. Machine learning for software engineering applications

With the emerging interest in machine learning techniques for software engineering applications, the information present in the dataset can help researchers to utilize the dataset for training machine learning models; for instance, quality metrics can be used as the features for code smell detection.

C. Correlating code quality with effort and productivity

The dataset can be used to establish a correlation between code quality and productivity of a software development team. Along the similar lines, the data can also be used to assess the effect of quality on other aspects of software development such as team churn. In this context, trend analysis of quality rank evolution of the analyzed repositories can also be investigated with other aspects or artifacts of software development.

V. Related work

Though code repository hosting platforms such as GitHub host a huge number of repositories, a large subset of them are either too small, inactive, or of poor quality [21]. The software engineering community has proposed a few mechanisms to separate them from the high-quality active repositories. GHTorrent [22] provides a method to query GitHub projects, obtain metadata, and select projects based on custom criteria. RepoRepears [5] analyzed a large number of GitHub repositories and evaluated them based on eight dimensions (architecture, community, continuous integration, documentation, history, issues, license, and unit testing). PHANTOM [6] applies k-means algorithm on five factors extracted from git history to classify a large number repositories in a resource-efficient manner. Markovtsev et al. [23] create a public git archive containing more than 182 thousand repositories and provide mechanism to obtain metadata of the repositories. However, none of these studies collect code quality information and treat code quality as one of the primary criterion to filter out or rank the source code repositories.

There has been some efforts to create code quality datasets. COMETS [24] consists of time-series data of 17 object-oriented metrics of ten open-source projects. Palomba et al. [25] offers a dataset of 243 manually validated instances of five code smells extracted from 20 open-source repositories. Lenarduzzi et al. [26] presents a technical debt dataset created from 33 Java open-source projects where they collect technical debt of each commit from SonarQube and detect code smells using Ptidej. Similarly, Diamantopoulos et al. [27] propose a dataset of 3,000 most popular open-source Java repositories containing DevOps metrics and coding violations detected by using PMD. The proposed dataset QScored is not only significantly large compared to the above mentioned datasets but also contains a rich set of code quality information.

VI. Limitations, Conclusions, and Future Work

The presented dataset covers repositories written only in Java and C#. This limitation can be addressed by utilizing code quality analysis tools for other programming languages; however, an extensive tool support to detect architecture and design smells for other languages is lacking presently.

Code quality information has been extensively used by the software engineering research community. We present a large dataset containing comprehensive code quality information for more than 86 thousand active repositories from GitHub. The dataset will further foster the software repository mining research involving code quality aspects by readily providing the required code quality information.

In the future, we would like to extend the dataset with all active Java or C# repositories on GitHub. Towards this, we have plan to create an automated pipeline subscribed to GitHub to fetch a project, analyze it, and upload it automatically. We also plan to extend the quality analysis to other aspects of code quality such as test quality by analyzing test smells and making it a part of the dataset.

References

[1] M. Fowler, *Refactoring: Improving the Design of Existing Programs*, 1st ed. Addison-Wesley Professional, 1999.

[2] G. Suryanarayana, G. Samarthyam, and T. Sharma, *Refactoring for Software Design Smells: Managing Technical Debt*, 1st ed. Morgan Kaufmann, 2014.

[3] S. H. Kan, *Metrics and Models in Software Quality Engineering*, 2nd ed. Addison-Wesley Longman Publishing Co., Inc., 2002.

[4] T. Sharma and D. Spinellis, "A survey on software smells," *Journal of Systems and Software*, vol. 138, pp. 158 – 173, 2018.

[5] N. Munaiah, S. Kroh, C. Cabrey, and M. Nagappan, "Curating github for engineered software projects," *Empirical Software Engineering*, vol. 22, no. 6, pp. 3219–3253, Dec 2017.

[6] P. Pickerill, H. Joshua Jungen, M. Ochodek, M. Maćkowiak, and M. Staron, "PHANTOM: Curating GitHub for engineered software projects using time-series clustering," *Empirical Software Engineering*, May 2020.

[7] F. Palomba, M. Zanoni, F. A. Fontana, A. De Lucia, and R. Oliveto, "Smells like teen spirit: Improving bug prediction performance using the intensity of code smells," in *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2016, pp. 244–255.

[8] J. Ferzund, S. N. Ahsan, and F. Wotawa, "Analysing bug prediction capabilities of static code metrics in open source software," in *Software Process and Product Measurement*, R. R. Dumke, R. Braungarten, G. Büren, A. Abran, and J. J. Cuadrado-Gallego, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 331–343.

[9] M. D'Ambros, M. Lanza, and R. Robbes, "An extensive comparison of bug prediction approaches," in *2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)*, 2010, pp. 31–41.

[10] E. Giger, M. D'Ambros, M. Pinzger, and H. C. Gall, "Method-level bug prediction," in *Proceedings of the 2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, 2012, pp. 171–180.

[11] A. Yamashita and L. Moonen, "Do code smells reflect important maintainability aspects?" in *2012 28th IEEE International Conference on Software Maintenance (ICSM)*, 2012, pp. 306–315.

[12] D. I. K. Sjøberg, A. Yamashita, B. C. D. Anda, A. Mockus, and T. Dybå, "Quantifying the effect of code smells on maintenance effort," *IEEE Transactions on Software Engineering*, vol. 39, no. 8, pp. 1144–1156, 2013.

[13] H. Alsolai and M. Roper, "A systematic literature review of machine learning techniques for software maintainability prediction," *Information and Software Technology*, vol. 119, p. 106214, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950584919302228

[14] V. Thakur, M. Kessentini, and T. Sharma, "QScored: An Open Platform for Code Quality Ranking and Visualization," in *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2020, pp. 818–821.

[15] T. Sharma, " Designite - A Software Design Quality Assessment Tool," May 2016, available online at http://www.designite-tools.com. [Online]. Available: https://doi.org/10.5281/zenodo.2566832

[16] ——, "Designitejava (enterprise)," Sep. 2019, available online at http://www.designite-tools.com/designitejava. [Online]. Available: https://doi.org/10.5281/zenodo.3401802

[17] T. Sharma, P. Singh, and D. Spinellis, "An empirical investigation on the relationship between design and architecture smells," *Empirical Software Engineering (EMSE)*, vol. 25, pp. 4020–4068, 2020.

[18] T. Sharma, "QScored: A Large Dataset of Code Smells and Quality Metrics," Jan. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.4468361

[19] ——, "Qscored agent," Feb. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.4568414

[20] I. Ahmed, C. Brindescu, U. A. Mannan, C. Jensen, and A. Sarma, "An empirical examination of the relationship between code smells and merge conflicts," in *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2017, pp. 58–67.

[21] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, "An in-depth study of the promises and perils of mining github," *Empirical Software Engineering*, vol. 21, no. 5, pp. 2035–2071, Oct 2016.

[22] G. Gousios, "The ghtorrent dataset and tool suite," in *Proceedings of the 10th Working Conference on Mining Software Repositories*, ser. MSR '13. IEEE Press, 2013, pp. 233–236.

[23] V. Markovtsev and W. Long, "Public git archive: A big code dataset for all," in *Proceedings of the 15th International Conference on Mining Software Repositories*, ser. MSR '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 34–37. [Online]. Available: https://doi.org/10.1145/3196398.3196464

[24] C. Couto, C. Maffort, R. Garcia, and M. T. Valente, "Comets: A dataset for empirical research on software evolution using source code metrics and time series analysis," *SIGSOFT Softw. Eng. Notes*, vol. 38, no. 1, p. 1–3, Jan. 2013. [Online]. Available: https://doi.org/10.1145/2413038.2413047

[25] F. Palomba, D. Di Nucci, M. Tufano, G. Bavota, R. Oliveto, D. Poshyvanyk, and A. De Lucia, "Landfill: An open dataset of code smells with public evaluation," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, 2015, pp. 482–485.

[26] V. Lenarduzzi, N. Saarimäki, and D. Taibi, "The technical debt dataset," in *Proceedings of the Fifteenth International Conference on Predictive Models and Data Analytics in Software Engineering*, ser. PROMISE'19. New York, NY, USA: Association for Computing Machinery, 2019, p. 2–11. [Online]. Available: https://doi.org/10.1145/3345629.3345630

[27] T. Diamantopoulos, M. D. Papamichail, T. Karanikiotis, K. C. Chatzidimitriou, and A. L. Symeonidis, "Employing contribution and quality metrics for quantifying the software development process," in *Proceedings of the 17th International Conference on Mining Software Repositories*, ser. MSR '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 558–562. [Online]. Available: https://doi.org/10.1145/3379597.3387490