

SQuaD: The Software Quality Dataset

A Multi-Dimensional Time-Aware Collection of Metrics for Large-Scale Empirical Research

Mikel Robredo
University of Oulu
Oulu, Finland
mikel.robredomanero@oulu.fi

Matteo Esposito
University of Oulu
Oulu, Finland
matteo.esposito@oulu.fi

Davide Taibi*
University of Southern Denmark
Vejle, Denmark
taibi@imada.sdu.dk

Rafael Peñaloza
University of Milano-Bicocca
Milan, Italy
rafael.penalozyanysen@unimib.it

Valentina Lenarduzzi*
University of Southern Denmark
Vejle, Denmark
lenarduzzi@imada.sdu.dk

Abstract

Software quality research increasingly relies on large-scale datasets that measure both the product and process aspects of software systems. However, existing resources often focus on limited dimensions, such as code smells, technical debt, or refactoring activity, thereby restricting comprehensive analyses across isolated quality dimensions. To address this gap, we present the Software Quality Dataset (SQuaD), a multi-dimensional, time-aware collection of software quality metrics extracted from 450 mature open-source projects across diverse ecosystems, including Apache, Mozilla, FFmpeg, and the Linux kernel. By integrating nine state-of-the-art static analysis tools, i.e., SonarQube, CodeScene, PMD, Understand, CK, JaSoMe, RefactoringMiner, RefactoringMiner++, and PyRef, our dataset unifies over 700 unique metrics at method, class, file, and project levels. Covering a total of 63,586 analyzed project releases, SQuaD also provides version control and issue-tracking histories, software vulnerability data (CVE/CWE), and process metrics proven to enhance Just-In-Time (JIT) defect prediction. The SQuaD enables empirical research on maintainability, technical debt, software evolution, and quality assessment at unprecedented scale. We also outline emerging research directions, including automated dataset updates and cross-project quality modeling to support the continuous evolution of software analytics. The dataset is publicly available on ZENODO (DOI: 10.5281/zenodo.17566690).

CCS Concepts

• **Computer systems organization** → **Maintainability and maintenance**; • **Security and privacy** → **Vulnerability management**; • **Software and its engineering** → **Software libraries and repositories**; **Software maintenance tools**; • **Information systems** → **Data mining**; • **General and reference** → **Metrics**; **Empirical studies**;

*Also with University of Oulu, Finland.



This work is licensed under a Creative Commons Attribution 4.0 International License.
MSR '26, Rio de Janeiro, Brazil

© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2474-9/2026/04
<https://doi.org/10.1145/3793302.3793312>

Keywords

software metrics, product metrics, process metrics, refactorings, behavioral metrics, RefactoringMiner, PyRef, SonarQube, Understand, CK, JaSoMe, PMD, CodeScene, RefactoringMinerPP

ACM Reference Format:

Mikel Robredo, Matteo Esposito, Davide Taibi, Rafael Peñaloza, and Valentina Lenarduzzi*. 2026. SQuaD: The Software Quality Dataset: A Multi-Dimensional Time-Aware Collection of Metrics for Large-Scale Empirical Research. In *23rd International Conference on Mining Software Repositories (MSR '26)*, April 13–14, 2026, Rio de Janeiro, Brazil. ACM, New York, NY, USA, 6 pages.
<https://doi.org/10.1145/3793302.3793312>

1 Introduction

Software maintenance is a core facet of Software Quality (SQua) as it helps teams to extend and correct their software system more easily [14]. The Software Engineering (SE) community considers multiple factors and mechanisms that affect and help to improve the software quality of a system [14, 23]. These facets can range from software vulnerabilities [10] to code quality issues [11], as well as technical debt management [31] and code refactoring operations [28, 34], among others.

Empirical mining software repository research relies on the public availability of open source software repositories hosted in platforms such as GitHub, and on the correctness of the mining activity that is performed [19]. Based on this premise, researchers often make significant efforts to select subsets of projects based on code quality [10, 33], as well as on the self-implemented *codes of conduct* in software foundations such as the *Apache Software Foundation* (ASF) [1]. Similarly, a popular technique for ensuring the quality of a software system during its development and maintenance is the employment of Static Analysis Tools (SAT) [11]. Multiple studies have exploited the use of SATs to remediate common quality issues [10, 22, 25, 35]. Most of the times, researchers concentrate on a specific set of projects and SATs due to the resource-intensive and time-consuming task of employing a larger number of SATs on a large-scale set of projects.

Consequently, existing works already provide the research community with large-scale datasets to enable researchers to answer potential research questions by leveraging the shared data. For instance, Lenarduzzi et al. [21] contributed to the SE literature with a large-scale dataset on Technical Debt (TD) metrics derived from SATs like SonarQube, later expanded by Graf-Vlachy and Wagner

[15]. In addition, further research efforts have been made to contribute to the SE community with datasets on software quality aspects such as software vulnerabilities [6], code smells and quality metrics [33], time series-based software evolution metrics [9], as well as code refactoring activity [18], among others [17, 20].

However, to date, no existing works combine all these SQua aspects into a single large-scale dataset. To such end, we leveraged nine state-of-the-art SATs to mine mature SE projects from sources such as the ASF, the Mozilla¹, the FFMpeg² foundations and the Linux kernel³. We employed **SonarQube** (and its **Sonar Way** configuration) and **CodeScene** to evaluate TD and security issues [21, 36], offering insights into maintainability and code health. Covering the aspect of code refactoring, we adopted **RefactoringMiner** (RMiner) [37], **RefactoringMiner++** (RMiner++) [27] and **PyRef** [4] for Java, C++ and Python languages accordingly. We assessed the *coding rule compliance* using **PMD** (*default rule set*) [10, 11] and **Understand** [5] by SciTools. We mined product and process metrics at different granularity levels with **CK** [3] and **JaSoMe** [16]. We expanded the mined data with up-to-date issue reports from GitHub, Jira and BugZilla issue trackers (ITS), their reported Common Vulnerabilities and Exposures (CVE), and Common Weakness Enumeration (CWE) types with their official definitions as well as additional process metrics demonstrated to improve JIT prediction accuracy [12, 13, 24]. Nonetheless, we designed the dataset to be extensible with additional SATs in future releases.

Thus, in this paper, we present *the Software Quality dataset* (SQuaD), which provides the community with a multi-dimensional time-aware collection of metrics for large-scale empirical research. The main contributions of this paper are:

- *The SQuaD*. A large-scale set of 450 projects where, by leveraging nine state-of-the-art SATs, we analyzed 725 metrics describing common SQua aspects from all the versions of their officially reported releases, covering metrics at method, class, file and project level.
- *Two data formats*. CSV files and a noSQL database, thus enabling researchers to access our dataset efficiently.
- The replication package with the scripts to use the SATs that produced this dataset.

Paper Structure. Section 2 describes the construction method adopted for this dataset. Section 3 presents the dataset and its usage. Section 4 highlights the future research opportunities the dataset can provide. Section 5 acknowledges the limitations of the dataset. Section 6 draws conclusions and future works.

2 Dataset construction

This section describes the data sources used to create the dataset, and the methodology used to gather the data, which we graphically present in Figure 1. The construction of the dataset required four main data mining stages: *Mining version control data*, *Mining SQua metrics from the selected SATs*, *extracting software vulnerability enumerations*, and *collecting software process metrics*.

2.1 Mining version control data

To collect the initial set of software repositories to include in our dataset, we considered mining classically investigated projects from sources such as the ASF, the Mozilla Foundation, and the Linux kernel [21]. We applied an additional filtering process to include only active, mature projects [8, 19, 32]. For that, we excluded **archived** projects or **based on forks**, as well as projects with **no available SBOM**. Furthermore, we excluded projects with **no activity in the last six months**, projects that had **less than three contributors**, and those that had **less than 50 stars on GitHub**.

We leveraged GitHub's API⁴ to mine their commit history as well as the issue tracking history for those projects that used GitHub as their ITS. We also mined the issue tracking history from those ASF reporting to use Jira⁵ and BugZilla⁶ as their official ITS⁷. With a total of 501 software repositories detected in the selected sources of data, only 450 reported published releases or tags in GitHub, which we set as the observational points to mine the SQua metrics, and thus build the historical development progress from the mined repositories.

2.2 Mining SQua metrics

This section describes the systematic methodology used to mine software metrics from the adopted SATs. Since each tool captures distinct aspects of software quality, Table 1 reports the number of metrics extracted per SAT and their covered dimensions. We provide instructions on replicating our mining pipeline in the replication package [30].

Table 1: Overview of adopted SATs, mined metrics, and aspects analyzed by each SAT.

Tool	#Metrics	Aspect covered (Metrics reference)
CK	88	Calculates class-level and method-level code metrics in Java projects. [3]
JaSoMe	70	Mines file, package, class & method quality metrics in Java projects. [16]
RMiner	103	Detects refactorings applied in the history of a Java project. [37]
RMiner++	16	Detects refactorings applied in the history of a C++ project. [27]
Understand	111	Mines file, class & entity quality metrics for multiple languages. [26]
SQ	192	Calculates several quality metrics & verifies the code's compliance against a specific set of "coding rules". [21]
PMD	114	Runs coding rules against source files to find violations. [11]
PyRef	9	Detects refactorings applied in the history of a Python project. [4]
CodeScene	22	Computes per-file comprehensive code health checks. [36]

Since each SAT required a different mining setting, we followed a systematic mining approach for each SAT in parallel. 1) We

¹<https://www.mozillafoundation.org/en/>

²<https://ffmpeg.org>

³<https://www.kernel.org/doc/html/latest/>

⁴<https://docs.github.com/en/rest?apiVersion=2022-11-28>

⁵<https://developer.atlassian.com/cloud/jira/platform/rest/v3/intro/#about>

⁶<https://bugzilla.readthedocs.io/en/5.2/api/>

⁷<https://issues.apache.org>

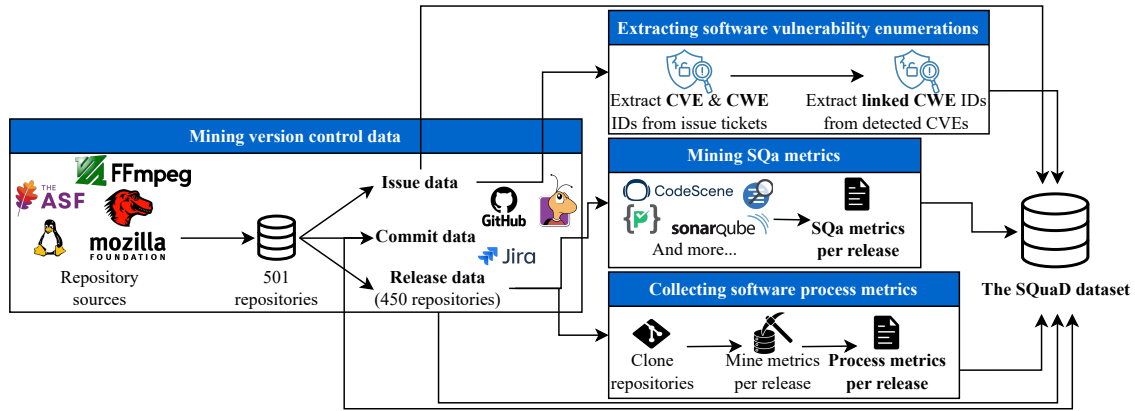


Figure 1: Overview of the dataset construction methodology.

cloned the software repository, and subsequently, 2) we looped through the project’s release commit hashes and checked out the cloned repository to the release version accordingly. 3) For each release iteration, we launched the SAT and mined the entire codebase of the repository.

With all the repositories mined for a specific SAT, we merged the outcome from all projects into a single CSV table. Since each of the adopted SATs mined the repositories at different granularity levels, we specify the granularity level, i.e., the analyzed object type per row, within the shared replication package. In case of overlapping metrics across SATs, and since each of them might use different methods to compute the metrics, the SQuaD includes them all.

In addition, and since SonarQube also reports coding issues based on the codebase’s compliance against SonarQube’s “coding rules” [22], we leveraged SonarQube’s API to retrieve all the raised issues for each of the release versions accordingly.⁸

2.3 Extracting software vulnerability enumerations

From the mined issue tracking data, we searched for all the regular expressions matching the pattern for official Software Vulnerability Enumerations (CVE) and that of software weaknesses, i.e., CVE- $\backslash d\{4\}-\backslash d\{4,7\}$ and CWE- $\backslash d\{3,4\}$. We acknowledge that this is the best effort, given that the correctness of reporting vulnerability enumerations depends on the author of the issue ticket. Subsequently, we collected all the publicly available information from the official Common Weakness Enumeration (CWE) index⁹ and the National Institute of Standards and Technology (NIST)¹⁰ for each of the matched enumerations. For that, on the one hand, we leveraged the available datasets with the official information for the currently existing CWEs and fetched the information about the matched CVEs. Similarly, we used the API access provided by the NIST and retrieved the information regarding each of the matched CVEs.

2.4 Collecting software process metrics

Recent research efforts have demonstrated that specific process metrics are more helpful than the structure of the source code itself when training JIT defect prediction models [12, 13, 24]. Since we already collected all the characteristics representing the structure of the code base throughout the entire release history of the mined repositories, we are now interested in collecting software process metrics at each release version of the projects. For that, we used Python’s `GitPython`¹¹ library to safely traverse through the entire version control history of the cloned repositories. Thus, we collected the process metrics highlighted in the literature at each release version (see shared entity relationship diagram [30]).

3 The Software Quality dataset

The SQuaD comprises measures for a total number of 725 SQua metrics distributed across the employed 9 state-of-the-art SATs. These results represent metric observations from a total of 63,586 analyzed project releases and tags, based on a total number of 450 software repositories. The dataset contains a total of 628,178 defect tickets, 2,622,413 GitHub commits, and official information on 1479 CVE and 175 CWE enumerations detected within the mined issue tickets. Furthermore, the dataset provides the computed value of 14 process metrics covering the entire version control history of the mined projects. On average, the projects included are over 9 years old, with a mean number of 125,500 total lines of code, 2465 stars in GitHub, and over 104 contributors per project.

The dataset is stored in two different formats. We utilize a noSQL database, specifically MongoDB.¹² We facilitate access to this format of the database through the *Binary JSON* (BSON) format¹³, the standard sharing format in MongoDB, and compressed via *Z-standard* [7]. Similarly, we provide the dataset in a series of CSV files following the same entity relationship designed for the database format. We include an entity diagram with the table hierarchy in the shared replication package [30] to facilitate its use.

⁸<https://docs.sonarsource.com/sonarqube-server/extension-guide/web-api>

⁹<https://cwe.mitre.org/index.html>

¹⁰<https://nvd.nist.gov/vuln>

¹¹<https://gitpython.readthedocs.io/en/stable/>

¹²<https://www.mongodb.com>

¹³<https://www.mongodb.com/resources/languages/bson>

- Table `projects_data` contains the links to the GitHub repository.
- Table `COMMITTS` reports the commit information retrieved from GitHub, including the commit hash, the commit message, the commit date and the alias of the commit author, among other attributes.
- Table `ISSUES` contains the issue tickets from the mined projects. Based on the column `its`, the table provides details about issues registered in GitHub, Jira and Bugzilla.
- Table `release_data` contains the identifier of the project releases and tags retrieved from GitHub as well as their related commit hash.
- Table `summary_statistics` contains summary statistics retrieved from GitHub, e.g., number of stars, the number of contributors, or the number of watchers.
- Table `PRJ_ITS_VLN_LINKAGE` contains the linkage between project identifiers, issue trackers, and detected vulnerability references.
- Table `cwe_data` contains the official information retrieved from the CWE official index on the enumerations detected in the issue tickets of the mined projects.
- Table `cve_data` contains the official information retrieved from the NIST on the enumerations detected in the issue tickets of the mined projects. Moreover, it also contains information on the enumerations related to the CWE weaknesses detected.
- Table `process_metrics` provides the computed values of the collected process metrics for all the releases mined from the projects included in the dataset.
- The `TOOL` tables consist a table per each SAT used during the mining process. Each of the tables contains observations at different granularity levels, in some cases uniform during the entire table (e.g. refactoring observations for RMiner), and in other cases at different granularity levels (e.g. file, class and method level for JaSoMe), always specifying the metric type through the `metric` column.

We made the dataset as well as the raw mined data accessible [29] in ZENODO. The compressed BSON database can be imported into MongoDB and thus be explored through the MongoDB shell or any other graphical interface supporting MongoDB. We also provide the dataset in CSV format, thus facilitating one CSV file per table listed above.

4 Impact and potential research directions

Software quality metrics stand as one of the most important source of information that can describe the development process of a project [21]. Consequently, this dataset stands as the largest dataset release till the date, combining SQua metrics from some of the state-of-the-art SATs employed for measuring SQua.

The SQuaD opens a wide availability of data for multiple potential use cases. Researchers can investigate time-dependent trends and variables over different granularity levels [28], for instance, in order to perform software evolution and change analysis. Similarly, multiple studies could leverage our dataset to benchmark different technical debt indicators such as code smells across different ecosystems (e.g. ASF, Linux Kernel).

Since the SQuaD integrates CVE/CWE and issue-tracking data, researchers investigating defect prediction can use our dataset to test further novel prediction models [10, 24], as well as forecasting models that might require the data to be already chronologically ordered. Building upon this, with the surge of models enabled by the *Transformer* architecture [38, 39], prediction models require a larger dimension of data for training. The SQuaD stands as a potential candidate to provide this capability to SE researchers.

5 Limitations

The creation of the SQuaD involved using some of the some of the most commonly used SATs. We are aware that these tools might analyze the code incorrectly under some conditions, especially when the programming language is structurally different. Hence, we aimed at only adopting state-of-the-art SATs to reduce this limitation. Similarly, the tools PyRef and RMiner++ reported null results when analyzing the same source code, hence the smaller size of their mined results. We relate this limitation to their novelty of their release, aiming to export the model of RefactoringMiner to other programming language.

Another important, yet controversial limitation of the dataset is its size. We aimed at mining some of the open-source repositories closely related to industry projects, and therefore, the dimension of the mined output resulted in a dimension that will require practitioners to have powerful machines to enable the use of the SQuaD.

6 Conclusion

In this work, we presented the SQuaD dataset. It stands as the largest source code dataset analyzing software projects based on different programming languages, and mined with SATs widely used in industry and research.

We described the dataset construction process to mine the data. We provided the SQuaD in CSV and BSON compressed formats to facilitate the compact use of the data. The SQuaD includes mined results of 725 SQua metrics from 9 different SATs, collected from the project versions across 450 software projects. The creation of the SQuaD required 7 months of mining process due to the license limitations of some other tools, as well as due to the size of some of the mined projects. The provided data allows researchers to perform large-scale studies without dealing with the data collection process, but directly fetch the data they need from the SQuaD and conduct the study.

Our plans involve expanding and updating the SQuaD by including new repositories, releasing the database in MySQL format, and expanding the tool selection.

Acknowledgments

The authors wish to acknowledge **CSC—IT Center for Science**, Finland, for generous computational resources. This work has been partially funded by projects Business Finland **6GSoft** [2], the Research Council of Finland **MuFano** (grants n. 349487 and 349488), and the Ministry of Education and Culture, Finland **FAST** (Finnish Software Engineering Doctoral Research Network). Lastly, the authors wish to acknowledge **SciTools** for their constant availability and assistance for using their software **Understand**.

References

- [1] 2025. About the ASF | Apache Software Foundation. <https://www.apache.org/foundation/>. Accessed: 2025-10-15.
- [2] Muhammad Azeem Akbar, Matteo Esposito, Sami Hyrynsalmi, Karthikeyan Dinesh Kumar, Valentina Lenarduzzi, Xiaozhou Li, Ali Mehraj, Tommi Mikkonen, Sergio Moreschini, Niko Makitalo, Markku Oivo, Anna-Sofia Paavonen, Risha Parveen, Kari Smolander, Ruoyu Su, Kari Systa, Davide Taibi, Nan Yang, Zheyang Zhang, and Muhammad Zohaib. 2024. 6GSoft: Software for Edge-to-Cloud Continuum. In *2024 50th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE Computer Society, Los Alamitos, CA, USA, 499–506. doi:10.1109/SEAA64295.2024.00082
- [3] Mauricio Aniche. 2015. *Java code metrics calculator (CK)*. Available in <https://github.com/mauricioaniche/ck/>.
- [4] Hassan Atwi, Bin Lin, Nikolaos Tsantalis, Yutaro Kashiwa, Yasutaka Kamei, Naoyasu Ubayashi, Gabriele Bavota, and Michele Lanza. 2021. PYREF: Refactoring Detection in Python Projects. In *21st IEEE International Working Conference on Source Code Analysis and Manipulation, SCAM 2021, Luxembourg, September 27-28, 2021*. IEEE, 136–141. doi:10.1109/SCAM52516.2021.00025
- [5] Alexander Bakhtin, Matteo Esposito, Valentina Lenarduzzi, and Davide Taibi. 2025. Network Centrality as a New Perspective on Microservice Architecture. In *22nd IEEE International Conference on Software Architecture, ICSA 2025, Odense, Denmark, March 31 - April 4, 2025*. IEEE, 72–83. doi:10.1109/ICSA65012.2025.00017
- [6] Quang-Cuong Bui, Riccardo Scandariato, and Nicolás E. Díaz Ferreyra. 2022. Vul4J: A Dataset of Reproducible Java Vulnerabilities Geared Towards the Study of Program Repair Techniques. In *19th IEEE/ACM International Conference on Mining Software Repositories, MSR 2022, Pittsburgh, PA, USA, May 23-24, 2022*. ACM, 464–468. doi:10.1145/3524842.3528482
- [7] Yann Collet and Murray S. Kuchera. 2021. Zstandard Compression and the 'application/zstd' Media Type. RFC 8878 (2021), 1–45. doi:10.17487/RFC8878
- [8] Dario Amoroso d'Aragona, Alexander Bakhtin, Xiaozhou Li, Ruoyu Su, Lauren Adams, Ernesto Aponte, Francis Boyle, Patrick Boyle, Rachel Koerner, Joseph Lee, Fangchao Tian, Yuqing Wang, Jesse Nyssölä, Ernesto Quevedo, Md Shahidur Rahaman, Amr S. Abdelfattah, Mika Mäntylä, Tomás Cerný, and Davide Taibi. 2024. A Dataset of Microservices-based Open-Source Projects. In *21st IEEE/ACM International Conference on Mining Software Repositories, MSR 2024, Lisbon, Portugal, April 15-16, 2024*. Diomidis Spinellis, Alberto Bacchelli, and Eleni Constantinou (Eds.). ACM, 504–509. doi:10.1145/3643991.3644890
- [9] Bruno Luan de Sousa, Mariza A. S. Bigonha, Kécia A. M. Ferreira, and Glaucia C. Franco. 2022. A Time Series-Based Dataset of Open-Source Software Evolution. In *19th IEEE/ACM International Conference on Mining Software Repositories, MSR 2022, Pittsburgh, PA, USA, May 23-24, 2022*. ACM, 702–706. doi:10.1145/3524842.3528492
- [10] Matteo Esposito, Valentina Falaschi, and Davide Falessi. 2024. An Extensive Comparison of Static Application Security Testing Tools. In *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering, EASE 2024, Salerno, Italy, June 18-21, 2024*. ACM, 69–78. doi:10.1145/3661167.3661199
- [11] Matteo Esposito, Mikel Robredo, Francesca Arcelli Fontana, and Valentina Lenarduzzi. 2025. On the correlation between architectural smells and static analysis warnings. *Softw. Qual. J.* 33, 4 (2025), 33. doi:10.1007/S11219-025-09730-7
- [12] Davide Falessi, Aalok Ahluwalia, and Massimiliano Di Penta. 2022. The Impact of Dormant Defects on Defect Prediction: A Study of 19 Apache Projects. *ACM Trans. Softw. Eng. Methodol.* 31, 1 (2022), 4:1–4:26. doi:10.1145/3467895
- [13] Davide Falessi, Simone Mesiano Laureani, Jonida Çarka, Matteo Esposito, and Daniel Alencar da Costa. 2023. Enhancing the defectiveness prediction of methods and classes via JIT. *Empir. Softw. Eng.* 28, 2 (2023), 37. doi:10.1007/S10664-022-10261-Z
- [14] Martin Fowler. 2002. Refactoring: Improving the Design of Existing Code. In *Extreme Programming and Agile Methods - XP/Agile Universe 2002, Second XP Universe and First Agile Universe Conference Chicago, IL, USA, August 4-7, 2002, Proceedings (Lecture Notes in Computer Science, Vol. 2418)*. Don Wells and Laurie A. Williams (Eds.). Springer, 256. doi:10.1007/3-540-45672-4_31
- [15] Lorenz Graf-Vlachy and Stefan Wagner. 2024. Different Debt: An Addition to the Technical Debt Dataset and a Demonstration Using Developer Personality. In *Proceedings of the 7th ACM/IEEE International Conference on Technical Debt, TechDebt 2024, Lisbon, Portugal, April 14-15, 2024*. Matthias Galster, Zadia Codabux, and Rodrigo O. Spinola (Eds.). ACM, 31–35. doi:10.1145/3644384.3644475
- [16] Rod Hilton and Dominik Ufer. 2021. *JaSoMe (Java Source Metrics) - Object Oriented Metrics analyzer for Java code*. Available in <https://github.com/rodhilton/jasome/>.
- [17] Youness Hourri, Alexandre Decan, and Tom Mens. 2025. A Dataset of Contributor Activities in the NumFrac Open-Source Community. In *22nd IEEE/ACM International Conference on Mining Software Repositories, MSR@ICSE 2025, Ottawa, ON, Canada, April 28-29, 2025*. IEEE, 159–163. doi:10.1109/MSR66628.2025.00035
- [18] István Kádár, Péter Hegedüs, Rudolf Ferenc, and Tibor Gyimóthy. 2016. A Manually Validated Code Refactoring Dataset and Its Assessment Regarding Software Maintainability. In *Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering, PROMISE 2016, Ciudad Real, Spain, September 9, 2016*. ACM, 10:1–10:4. doi:10.1145/2972958.2972962
- [19] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M. Germán, and Daniela E. Damian. 2016. An in-depth study of the promises and perils of mining GitHub. *Empir. Softw. Eng.* 21, 5 (2016), 2035–2071. doi:10.1007/S10664-015-9393-5
- [20] Rio Kishimoto, Tetsuya Kanda, Yuki Manabe, Katsuro Inoue, Shi Qiu, and Yoshiki Higo. 2025. A Dataset of Software Bill of Materials for Evaluating SBOM Consumption Tools. In *22nd IEEE/ACM International Conference on Mining Software Repositories, MSR@ICSE 2025, Ottawa, ON, Canada, April 28-29, 2025*. IEEE, 576–580. doi:10.1109/MSR66628.2025.00090
- [21] Valentina Lenarduzzi, Nyyti Saarimäki, and Davide Taibi. 2019. The Technical Debt Dataset. In *Proceedings of the Fifteenth International Conference on Predictive Models and Data Analytics in Software Engineering, PROMISE 2019, Recife, Brazil, September 18, 2019*. Leandro L. Minku, Foutse Khomh, and Jean Petric (Eds.). ACM, 2–11. doi:10.1145/3345629.3345630
- [22] Valentina Lenarduzzi, Nyyti Saarimäki, and Davide Taibi. 2020. Some Sonar-QuBE issues have a significant but small effect on faults and changes. A large-scale empirical study. *J. Syst. Softw.* 170 (2020), 110750. doi:10.1016/J.JSS.2020.110750
- [23] Valentina Lenarduzzi, Alberto Sillitti, and Davide Taibi. 2018. A Survey on Code Analysis Tools for Software Maintenance Prediction. In *Proceedings of 6th International Conference in Software Engineering for Defence Applications, SEDA 2018, Rome, Italy, June 7-8, 2018 (Advances in Intelligent Systems and Computing, Vol. 925)*. Paolo Ciancarini, Manuel Mazzara, Angelo Messina, Alberto Sillitti, and Giancarlo Succi (Eds.). Springer, 165–175. doi:10.1007/978-3-030-14687-0_15
- [24] Lech Madeyski and Marian Jureczko. 2015. Which process metrics can significantly improve defect prediction models? An empirical study. *Softw. Qual. J.* 23, 3 (2015), 393–422. doi:10.1007/S11219-014-9241-7
- [25] Fabio Palomba, Marco Zanoni, Francesca Arcelli Fontana, Andrea De Lucia, and Rocco Oliveto. 2016. 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, Raleigh, NC, USA, October 2-7, 2016*. IEEE Computer Society, 244–255. doi:10.1109/ICSME.2016.27
- [26] Foyzur Rahman and Premkumar T. Devanbu. 2013. How, and why, process metrics are better. In *35th International Conference on Software Engineering, ICSE '13, San Francisco, CA, USA, May 18-26, 2013*. David Notkin, Betty H. C. Cheng, and Klaus Pohl (Eds.). IEEE Computer Society, 432–441. doi:10.1109/ICSE.2013.6606589
- [27] Benjamin Ritz, Aleksandar Karakas, and Denis Helic. 2025. Refactoring Detection in C++ Programs with RefactoringMiner++. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering, FSE Companion 2025, Clarion Hotel Trondheim, Trondheim, Norway, June 23-28, 2025*. Leonardo Montecchi, Jingyue Li, Denys Poshyvanyk, and Dongmei Zhang (Eds.). ACM, 1163–1167. doi:10.1145/3696630.3728602
- [28] Mikel Robredo, Matteo Esposito, Fabio Palomba, Rafael Peñaño, and Valentina Lenarduzzi. 2024. Analyzing the Ripple Effects of Refactoring. A Registered Report. *This Registered Report has been accepted at ICSME (2024)*. doi:10.2139/ssrn.4938000
- [29] Mikel Robredo, Matteo Esposito, Davide Taibi, Rafael Peñaño, and Valentina Lenarduzzi. 2025. *SQuaD: The Software Quality Dataset - Dataset*. doi:10.5281/zenodo.17566690
- [30] Mikel Robredo, Matteo Esposito, Davide Taibi, Rafael Peñaño, and Valentina Lenarduzzi. 2025. *SQuaD: The Software Quality Dataset - Replication Package*. doi:10.5281/zenodo.17541471
- [31] Nyyti Saarimäki, Valentina Lenarduzzi, and Davide Taibi. 2019. On the diffuseness of code technical debt in Java projects of the apache ecosystem. In *Proceedings of the Second International Conference on Technical Debt, TechDebt@ICSE 2019, Montreal, QC, Canada, May 26-27, 2019*. Paris Argeriou and Klaus Schmid (Eds.). IEEE / ACM, 98–107. doi:10.1109/TECHDEBT.2019.00028
- [32] Nyyti Saarimäki, Mikel Robredo, Valentina Lenarduzzi, Sira Vegas, Natalia Juristo, and Davide Taibi. 2025. Does microservice adoption impact the velocity? A cohort study. *Empir. Softw. Eng.* 30, 5 (2025), 130. doi:10.1007/S10664-025-10673-7
- [33] Tushar Sharma and Marouane Kessentini. 2021. QScore: A Large Dataset of Code Smells and Quality Metrics. In *18th IEEE/ACM International Conference on Mining Software Repositories, MSR 2021, Madrid, Spain, May 17-19, 2021*. IEEE, 590–594. doi:10.1109/MSR52588.2021.00080
- [34] Danilo Silva, Nikolaos Tsantalis, and Marco Túlio Valente. 2016. Why we refactor? confessions of GitHub contributors. In *Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE 2016, Seattle, WA, USA, November 13-18, 2016*. Thomas Zimmermann, Jane Cleland-Huang, and Zhendong Su (Eds.). ACM, 858–870. doi:10.1145/2950290.2950305
- [35] Davide Taibi, Andrea Janes, and Valentina Lenarduzzi. 2017. How developers perceive smells in source code: A replicated study. *Inf. Softw. Technol.* 92 (2017), 223–235. doi:10.1016/J.INFSOF.2017.08.008
- [36] Adam Tornhill. 2018. Assessing Technical Debt in Automated Tests with CodeScene. In *2018 IEEE International Conference on Software Testing, Verification and*

- Validation Workshops, ICST Workshops, Västerås, Sweden, April 9-13, 2018.* IEEE Computer Society, 122–125. doi:10.1109/ICSTW.2018.00039
- [37] Nikolaos Tsantalis, Ameya Ketkar, and Danny Dig. 2022. RefactoringMiner 2.0. *IEEE Trans. Software Eng.* 48, 3 (2022), 930–950. doi:10.1109/TSE.2020.3007722
- [38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 5998–6008. <https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html>
- [39] Wei Zheng, Lijuan Tan, and Chengbin Liu. 2021. Software Defect Prediction Method Based on Transformer Model. In *2021 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*. 670–674. doi:10.1109/ICAICA52286.2021.9498179