

© 2022 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Access to this work was provided by the University of Maryland, Baltimore County (UMBC) ScholarWorks@UMBC digital repository on the Maryland Shared Open Access (MD-SOAR) platform.

**Please provide feedback**

Please support the ScholarWorks@UMBC repository by emailing [scholarworks-group@umbc.edu](mailto:scholarworks-group@umbc.edu) and telling us what having access to this work means to you and why it's important to you. Thank you.

# Sonarlizer Xplorer: a tool to mine Github projects and identify technical debt items using SonarQube

## Abstract

The advancement of artificial intelligence and the implementation of machine learning capabilities in programming languages such as Python, along with cloud services, allow researchers to apply methods to cluster and predict behaviors and patterns in software engineering data. On the other hand, these methods need a large amount of data in order to work with high accuracy in different contexts. This paper introduces Sonarlizer Xplorer: a tool that captures a large number of technical debt items and code metrics from public Github projects. Sonarlizer Xplorer is composed of two sub-tools. The first is Github Xplorer, responsible for mining public Github repositories from an initial project. The second is Sonarlizer, responsible for taking projects and analyzing them using SonarQube. We used the tool over four months, collecting technical debt items and code metrics on almost 46,000 public Java projects. In addition, we mined over 57 million repositories and 4 million users.

**CCS Concepts:** • Information systems → Data mining; • Software and its engineering → Software libraries and repositories.

**Keywords:** metric and issue analyzer, public projects mining, Github, SonarQube

## ACM Reference Format:

. 2022. Sonarlizer Xplorer: a tool to mine Github projects and identify technical debt items using SonarQube. In *Proceedings of TechDebt '22: International Conference on Technical Debt 2022 (TechDebt '22)*. ACM, New York, NY, USA, 5 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

## 1 Introduction

The implementation of machine learning methods in programming languages has helped researchers and practitioners to apply them to create and improve software tools [5, 9]. In addition, research areas such as software engineering can also use these methods to improve the precision of results

and discover insights that would be missed by traditional statistical analyses.

Machine learning is one of the most common approaches used from artificial intelligence [7, 10]. It can be used to create clusters and predict behavior/patterns in data sets. The process to validate the application of a machine learning algorithm in a specific context consists of steps such as the following:

1. Split the data in around 80% to train the algorithm and 20% to validate it;
2. Input training data so that the algorithm can learn how to categorize or predict based on that data;
3. Apply data validation to assess the algorithm's precision and variance.
4. Repeat steps 1, 2, and 3 several times, splitting the data randomly and calculating the mean and variance of the accuracy to avoid bias in the training data.

Applying the previous steps depends on a large amount of data to decrease training bias and increase accuracy for different contexts.

Many experiments and case studies use only one or a few software projects as the object of research on technical debt studies. Just a few of them provide large amounts of data about technical debt.

The main goal of this paper is to introduce Sonarlizer Xplorer<sup>1</sup>: an open-source tool to identify a massive number of technical debt items and code metrics from public Github<sup>2</sup> repositories.

Our application is composed of two other tools: Github Xplorer and Sonarlizer. The first is responsible for mining public projects on Github using the relationship between repositories, users, and organizations. The second goes through the repository list and analyzes each repository using SonarQube to identify technical debt items and extract code metrics.

The main feature of Sonarlizer Xplorer is the ability to provide a list of technical debt items and code metrics for public Github projects. Another feature that contributes to the process is the generation of lists of public Github repositories, users, and organizations.

The primary users of our open-source tool are researchers who need to collect a large number of technical debt items from real projects. In addition, a researcher with programming skills can expand the tool to collect specific data to develop their own research.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

*TechDebt '22*, May 22–23, 2022, Pittsburgh, United States

© 2022 Association for Computing Machinery.

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...\$15.00

<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

<sup>1</sup><https://github.com/diogojpina/sonarlizer-xplorer>

<sup>2</sup><http://github.com>

This paper is organized as follows: Section 2 describes Sonarlizer Xplorer features, main users, and use cases. Section 3 discusses the tool architecture, technologies, and implementation. Section 4 reports the results of using the tool. Section 5 describes the tools related to Sonarlizer Xplorer. Section 6 presents a few possible enhancements. Section 7 shows the conclusions and final remarks. Section 8 shows the tool license.

## 2 Sonarlizer Xplorer

Sonarlizer Xplorer is a tool to mine and analyze public Github projects resulting in a dataset with many technical debt items and code metrics. It also results in a list of Github repositories, users, and organizations. The mining process starts with a repository. The tool finds users and organizations through the Github API. It then finds new repositories using the repositories already found. Then the tool analyzes each project using SonarQube in order to identify technical debt items and extract code metrics.

Sonarlizer Xplorer is developed in Node.JS and uses MongoDB data to handle large volumes of non-relational data. In addition, all SonarQube data, including technical debt items and code metrics, are stored in PostgreSQL. We chose these technologies to allow distributed computing, large amounts of data, and a simple tool installation.

### 2.1 Features

Sonarlizer Xplorer's main feature is the generation of lists of technical debt items and code metrics. Also, the tool provides a list of users, repositories, and organizations found during the mining process. In addition, it also provides all SonarQube results for each project analyzed.

The application can be customized to get more data provided by the Github API, such as commit files, modifications, and comments; Github issues; pull requests; events; and topics. It is also possible to add more SonarQube extensions to analyze more aspects such as test coverage, security vulnerability, code metrics, and more technical debt types.

### 2.2 Main Users

The primary users of Sonarlizer Xplorer tools are researchers who need large amounts of data from software projects to use in their investigations, in particular technical debt items and code metrics. Such a need for large data sets is often associated with the use of machine learning as part of the research design.

### 2.3 Use Cases

Sonarlizer Xplorer can be used to collect data and profile public Github projects, that is, to calculate statistics for technical debt items and code metrics. For example, we can use it to calculate the mean, median, and standard deviation of project's lines of code. Alternatively, it can also be used to

compute the average of a specific technical debt type on projects with certain characteristics, such as size (e.g. "on average, how many God Classes are found in projects with less than 5KLOC?").

Lastly, our tool can be very useful when finding public software projects hosted on Github and their respective developers. After analyzing the SonarQube project, we sent an interactive survey to the developers, asking when a technical debt item should be paid off. We used the code metrics collected by the tool and the survey responses to train and evaluate machine learning methods to try to prioritize the payment of technical debt items.

## 3 Architecture, Technologies, and Implementation

First, Sonarlizer Xplorer calls Github Xplorer to walk through the Github API mining public repositories and storing them in a MongoDB database. Then, the tool calls Sonarlizer to analyze each repository using SonarQube to identify technical debt items and extract code metrics. Figure 1 shows the Sonarlizer Xplorer architecture where the flow starts on Github and finishes with technical debt items and code metrics stored in a PostgreSQL database.

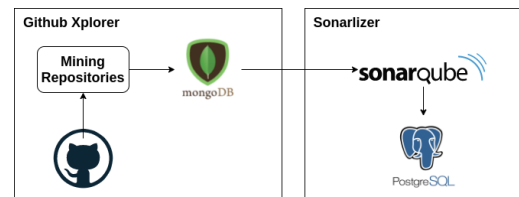


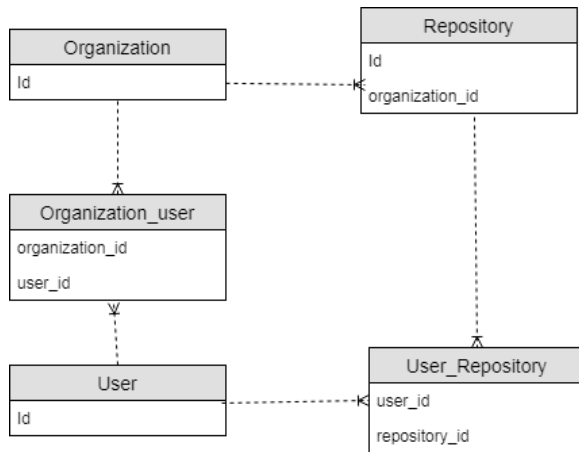
Figure 1. Sonarlizer Xplorer architecture.

### 3.1 Github Xplorer

GitHub Xplorer is a tool for mining public Github repositories to find real software projects and related developers. Each project's source code is stored in a repository. There is currently no public dataset available containing Github repositories and developers' information, so we used Github Xplorer to collect the data to construct this dataset.

**3.1.1 Finding Repositories and Developers.** A repository is an entity that represents a project codebase, and it is related to many users (developers). It can also be owned by an organization (defined as a group of developers). Examples of repositories include Apache Maven, Google Kubernetes, Microsoft Visual Studio Code, and Tensor Flow. Users can contribute to many repositories. They can also participate in many organizations. Examples of organizations include Apache Foundation, Microsoft, and Kubernetes. Note that both users and organizations can own repositories, but a repository has only one owner (user or organization). Figure

2 shows Github data entities and the relationships among them.



**Figure 2.** Github data entities and relationships.

We used the GitHub REST API to access public Github data. This API has a limit of 5,000 requests per hour per access token, resulting in 120 thousand requests per day. Although this seems like a lot of requests, it would take more than 80 days to mine 10 million repositories, for instance.

We implemented a feature that uses a list of access tokens to overcome this issue. Then, each time the tool runs, it takes a random token to access the API, which helps reduce the mining time. For example, by adding 10 access tokens to the tool, the 10 million requests to find repositories would be made in about 8 days instead of 80.

Each Github API call returns only one repository, organization, or user (developer). For this reason, we had to make use of entity relationships to find all the data. For example, when we request a repository, we can access the list of developers who contribute to it and the owner organization, if it exists. When we request an user, we can access the list of repositories they contribute to and the list of organizations to which they belong.

Github Xplorer uses three sets (lists) to store entities to be processed: repositories, organizations, and users. For each repository, the tool gets the entity data, list of users, and organizations (if an organization is the owner); thus, each repository request can find new users and one organization. For each user, the tool collects user data, its repositories and organizations; thus, each user request can find new repositories and organizations. For each organization, the tool gets organization data, its repositories and users; thus, each organization request can find new users and repositories.

The following snippet shows the pseudo-code for processing the repository set. The Git Xplorer similarly treats user and organization sets.

```

1 SET repositories_to_process , users_to_process ,
  organizations_to_process ;

```

```

2 SET repositories_set , users_set ,
  organizations_set ;
3
4 begin parallel :
5 while repositories_to_process is not empty
6   repository = pop repositories_to_process
7   getGithubInfo(repository)
8   addUpdate(repository , repositories_set)
9
10  users = getGithubUsers(repository)
11  push(users , users_to_process)
12
13  organization = getGithubOrganizations(
    repository)
14  push(organizations ,
    organizations_to_process)
15 end parallel

```

Note that a user or organization is stored in the set only if they are not on the processed or to be processed list.

The data mining starts by adding a Github repository to the repository set. The tool consumes the user and organization lists iteratively to get the complete entity information and related entities that have not yet been processed to feed the queues.

On Sonarlizer Xplorer, each repository is stored in a document in the repository in a MongoDB collection. The status property identifies when the project was (status = 1) or not (status = 0) analyzed. Figure 3 shows an example of repository document in MongoDB.

### 3.2 Sonarlizer

Sonarlizer automatically performs a SonarQube analysis in order to identify technical debt items and code metrics in the public software projects hosted on GitHub and discovered by Github Xplorer. For Java projects, Sonarlizer needs to compile the code before analyzing it. For that, the tool identifies the builder, which can be Apache Maven<sup>3</sup>, Apache Ant<sup>4</sup>, or Gradle<sup>5</sup>. Then, it uses the build specific commands for these tools to compile and send the compiled files to SonarQube to analyze. If no builder is identified, the standard SonarQube command is performed through the SonarScanner tool.

As input, Sonarlizer queries the MongoDB repository collection to retrieve the Github repository data. Sonarlizer clones the repository from GitHub to a local machine. Then it performs a SonarQube analysis to identify technical debt items such as naming convention violations, high code complexity and large code files; and code metrics, such as number of lines, number of files, complexity average by file, and cognitive complexity. SonarQube has a list of rules that identify technical debt items, and when one of these rules is broken, an issue is created. Some examples of rules are *Member Name*, *Unused Imports*, *Nested If Depth*, and *Method Length*.

<sup>3</sup><https://maven.apache.org>

<sup>4</sup><https://ant.apache.org>

<sup>5</sup><https://gradle.org>

```

1 {
2   _id: ObjectId('6054100161804f0006f9583d'),
3   id: 206483,
4   full_name: 'apache/maven',
5   status: 1,
6   archived: false,
7   created_at: ISODate('2009-05-21T03:22:03.000Z'),
8   default_branch: 'master',
9   description: 'Apache Maven core',
10  disabled: false,
11  fork: false,
12  forks: 1853,
13  forks_count: 1853,
14  has_downloads: true,
15  has_issues: false,
16  has_pages: false,
17  has_projects: true,
18  has_wiki: false,
19  homepage: 'https://maven.apache.org/ref/current',
20  language: 'Java',
21  name: 'maven',
22  network_count: 1853,
23  node_id: 'MDEwOlJlcG9zaXRvcnkyMDY0ODM=',
24  open_issues: 57,
25  open_issues_count: 57,
26  'private': false,
27  pushed_at: ISODate('2021-03-17T10:43:08.000Z'),
28  size: 47319,
29  stargazers_count: 2473,
30  subscribers_count: 212,
31  updated_at: ISODate('2021-03-19T01:59:59.000Z'),
32  watchers: 2473,
33  watchers_count: 2473
34 }

```

Figure 3. Repository document example.

Finally, Sonarlizer goes through the commit list to relate each source file to the users who have contributed to it.

## 4 Results

As a result, Sonarlizer Xplorer provides a list of technical debt items and code metrics for a large number of public Github repositories. Figure 4 shows an issue list example on SonarQube and a code metrics list example on SonarQube.

We used the tool over about four months to mine Github. We managed to find 57,382,956 repositories and 4,430,010 users<sup>6</sup>. Table 1 shows the numbers of projects found by programming language.

In parallel with mining, we also extracted 609,884 Java projects where 45,994 (about 7.5%) were successfully analyzed by SonarQube. The low rate is due to the fact that, to analyze Java code on SonarQube, it is first necessary to compile projects. Even using a builder, most of the projects require specific Java versions, configuration files, or additional parameters to compile successfully. Table 2 shows examples of the number of technical debt items by severity and projects by *ncloc*.

<sup>6</sup>Data sample in <https://raptor210.startdedicated.com/sonarlizer-xplorer/data.tgz> - temporary URL

**Table 1.** Example of project quantities mined by programming language.

Language	# of projects
JavaScript	9,867,584
Python	5,372,460
Java	4,777,964
Ruby	2,031,174
C++	1,993,982
C	1,468,370
C#	1,466,983

**Table 2.** Number of technical debt items by severity and projects divided by non-comment line of code (*ncloc*).

Severity	# TD items	ncloc	# Projects
Blocker	467k	<1k	27k
Critical	2.4M	1k - 10k	14k
Major	5.2M	10k - 100k	4.6k
Minor	6.6M	100k - 500k	390
Info	400k	>500k	17

## 5 Related Tools

Most of the Github mining tools can mine data from a given repository, such as RepoDriller [2], which extracts commits, developers, modifications, diffs, and source code. GH Torrent [8] allows querying of Github events from public repositories. GH Crawler [1] is a Microsoft project that finds new repositories, although it was created to retrieve all GitHub entities related to an organization, repository, user, and team.

All these tools analyze just one project in order to identify technical debt items and code metrics. SonarQube [6] is one of the tools that can do that. Findbugs [4] finds bugs and technical debt items and classifies them according to the severity. PMD [3] runs a cross-language static code analysis to find technical debt items.

Although there are many tools to mine a given Github repository and some can be used to identify technical debt items in that repository, none of them finds the repositories and analyzes them to identify technical debt items and extract code metrics.

## 6 Future Enhancements

A first improvement could be adding new repository integration to Github Xplorer, for example, to allow for mining public projects on Gitlab<sup>7</sup>, BitBucket<sup>8</sup> and SourceForge<sup>9</sup>.

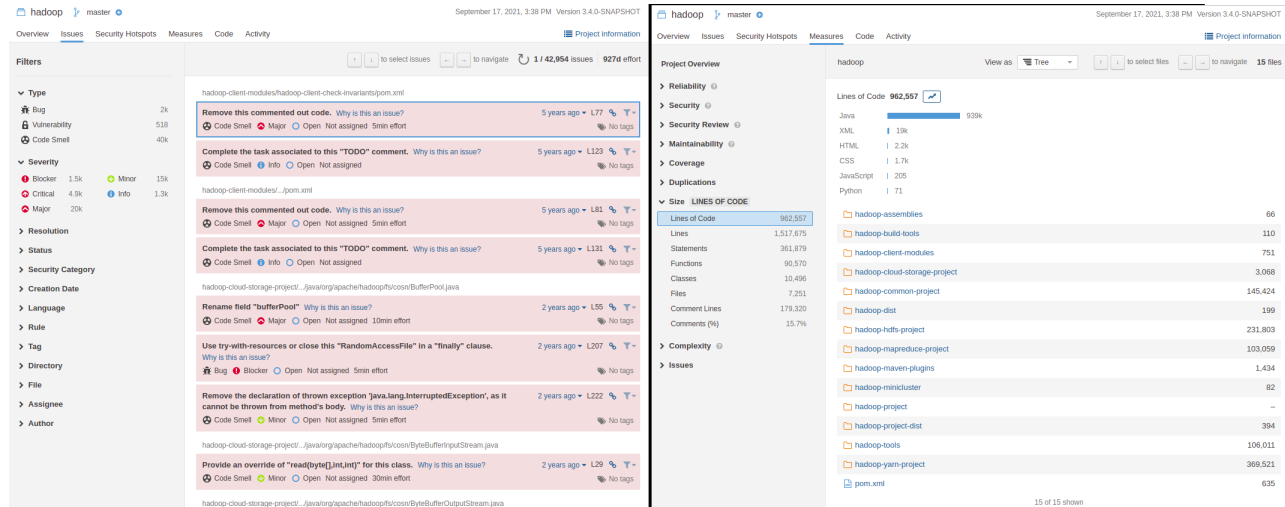
On the other hand, to increase the data types provided by the tool, the Github Analyzer tool could collect other metrics and data related to the hosted project, such as commits and

<sup>7</sup><https://www.gitlab.com>

<sup>8</sup><https://www.atlassian.com/software/bitbucket>

<sup>9</sup><https://sourceforge.net>





**Figure 4.** Left: Example of technical debt items on SonarQube. Right: Example of code metrics on SonarQube.

releases. Besides that, Sonarlizer Xplorer could use other tools to analyze the code and provide more code metrics and find new technical debt items, such as FindBugs<sup>10</sup> and PMD<sup>11</sup>.

## 7 Final Remarks

This paper introduces Sonarlizer Xplorer, a tool to mine public Github repositories and analyze them using SonarQube to identify technical debt items and code metrics. We describe how the tool explores Github through its API to find a list of public repositories related to an initial repository. Then it runs a SonarQube analysis to extract a list of technical debt items and a list of code metrics.

We also show possible applications of the tool in research using conventional statistics and more complex methods such as machine learning to analyze and predict patterns and behaviors for technical debt prioritization.

The tool is in its first version. Thus, several features can be implemented, such as mining other code repository host platforms, applying other code analysis tools, and collecting metrics from code repositories.

Sonarlizer Xplorer is a tool that researchers can use to increase the number of technical debt items for analysis, in order to derive more robust conclusions and find new approaches and behaviors that cannot be found by analyzing a few projects.

## 8 License

The Sonarlizer Xplorer is licensed under the MIT License, a short and simple permissive free software license to any

person obtaining a copy of this software and associated documentation files<sup>12</sup>. More details can be found in the license file<sup>13</sup>.

## Acknowledgments

This study was financed in part by the Coordenacao de Aperfeicoamento de Pessoal de Nivel Superior – Brasil (CAPES) – Finance Code 001.

## References

- [1] [n.d.]. Microsoft GHCrawler. <https://github.com/Microsoft/ghcrawler>
- [2] M Aniche. 2012. Repodriller. <https://github.com/mauricioaniche/repodriller>
- [3] Joao Eduardo M Araujo, Silvio Souza, and Marco Tulio Valente. 2011. Study on the relevance of the warnings reported by Java bug-finding tools. *IET software* 5, 4 (2011), 366–374.
- [4] Nathaniel Ayewah, William Pugh, David Hovemeyer, J David Morgenthaler, and John Penix. 2008. Using static analysis to find bugs. *IEEE software* 25, 5 (2008), 22–29.
- [5] David Barstow. 1988. Artificial intelligence and software engineering. In *Exploring artificial intelligence*. Elsevier, 641–670.
- [6] G Ann Campbell and Patroklos P Papapetrou. 2013. *SonarQube in action*. Manning Publications Co.
- [7] Issam El Naqa and Martin J Murphy. 2015. What is machine learning? In *machine learning in radiation oncology*. Springer, 3–11.
- [8] Georgios Gousios. 2013. The GHTorrent dataset and tool suite. In *Proceedings of the 10th Working Conference on Mining Software Repositories (San Francisco, CA) (MSR)*. 233–236. <http://www.gousios.gr/bibliography/G13.html>
- [9] Mark Harman. 2012. The role of artificial intelligence in software engineering. In *2012 First International Workshop on Realizing AI Synergies in Software Engineering (RAISE)*. IEEE, 1–6.
- [10] Tom M Mitchell et al. 1997. Machine learning. (1997).

<sup>10</sup><http://findbugs.sourceforge.net>

<sup>11</sup>[https://pmd.github.io/latest/pmd\\_java\\_metrics\\_index.html](https://pmd.github.io/latest/pmd_java_metrics_index.html)

<sup>12</sup><https://github.com/git/git-scm.com/blob/main/MIT-LICENSE.txt>

<sup>13</sup><https://github.com/diogojpina/sonarlizer-xplorer/blob/master/LICENSE>