PANDORA: Continuous Mining Software Repository and Dataset Generation

Hung Nguyen
Aalto University
Helsinki, Finland
hung.5.nguyen@aalto.fi

Francesco Lomio
Tampere University
Tampere, Finland
francesco.lomio@tuni.fi

Fabiano Pecorelli
Tampere University
Tampere, Finland
fabiano.pecorelli@tuni.fi

Valentina Lenarduzzi
University of Oulu
Oulu, Finland
valentina.lenarduzzi@oulu.fi

***Abstract*—During the mining software repository activities, a huge amount of data gathered from different sources is analyzed. Different tools have been developed for collecting and aggregating data from repositories, but they do not easily allow researchers to develop new extractors, to integrate the data collected from other platforms, and in particular from platforms that delete the data periodically. Moreover, mining software repository studies are commonly performed on old versions of software projects and their results are not commonly periodically updated. As a result of the non-continuously updated studies, practitioners often do not trust results from empirical studies. In order to overcome the aforementioned issues, in this paper, we present PANDORA, a tool that automatically and continuously mines data from different existing tools and online platforms and enables to run and continuously update the results of mining software repository studies. To evaluate the applicability of our tool, we currently analyzed 365 projects (developed in different languages), continuously collecting data from December 2020 to May 2021 and running an example study, investigating the build-stability of SonarQube rules.**

Link to dashboard: http://sqa.rd.tuni.fi/superset/dashboard/1
Link to source code: https://github.com/clowee/PANDORA
Link to 5-minutes video: https://youtu.be/CuVO9YGJ59I

***Index Terms*—Mining Software Repository, Data Analysis, Machine Learning**

I. Introduction

Mining software repository (MSR) activity requires analyzing a huge amount of data gathering from different sources such as source code, version control systems, and issue tracking systems [1], [2], [3]. The adaption of these tools can be a valid and useful support to researchers during their development activities, such as code reviews, testing, or code quality evaluation.

Different online platforms are used to collect data from software projects. GitHub[1], Sourceforge[2] and Jira[3] are some of the most commonly used platforms. Other platforms such as SonarCloud[4], BlackDuck[5], CodeScene[6] and many others evaluate different information on the project repositories. However, in order to save disk space, they commonly delete the detailed analysis results from their database after a short period of time, making complex the analysis of the whole project history for MSR researchers.

Different tools have been developed for collecting and aggregating data from repositories. From databases aggregating and collecting history of code repositories (e.g. The SourceForge Research Data Archive [4], the GHArchive), Issue trackers [5] but also tools for collecting and storing multiple information on multiple platforms (e.g. Software Heritage[7] and OpenHub[8] (formerly Ohoh)s).

However, all these platforms allow researchers to download different types of data to be used in their studies, but do not allow researchers to develop new extractors to integrate the data collected from other platforms, and in particular to platforms that delete the data periodically. Moreover, these tools and do not allow researchers to continuously apply their MSR approaches to the current version state of the software. As a result, studies conducted a few years ago might be outdated because of the new versions of the software analyzed or because of the new version of the tools analyzing the projects.

As an example, in 2020 we published two papers analyzing the fault proneness of SonarQube[9] issues [6], [7]. However, both works were performed using a dataset collecting project data from 2001 to 2015 and using an older version of SonarQube (version 7.4). Re-running the same study today, with the most recent version of SonarQube (version 8.9LTS on June 2021) and on project data collected in the last year would probably yield totally different results. The same might apply to a large number of similar studies conducted in the last years.

As a result of the non-continuously updated studies, practitioners often do not trust results from empirical studies, especially because they are applied to old versions of the projects or they are not easily replicable in their context.

In order to overcome to the aforementioned issues, in this paper, we present PANDORA, a tool that automatically and continuously mines data from different existing tools and online platforms and enable to run and continuously update the results of MSR studies.

[1]GitHub http://github.com
[2]https://sourceforge.net
[3]https://www.atlassian.com
[4]https://sonarcloud.io
[5]https://www.blackducksoftware.com
[6]https://codescene.io
[7]https://www.softwareheritage.org
[8]OpenHub https://www.openhub.net
[9]SonarQube http://www.sonarqube.org

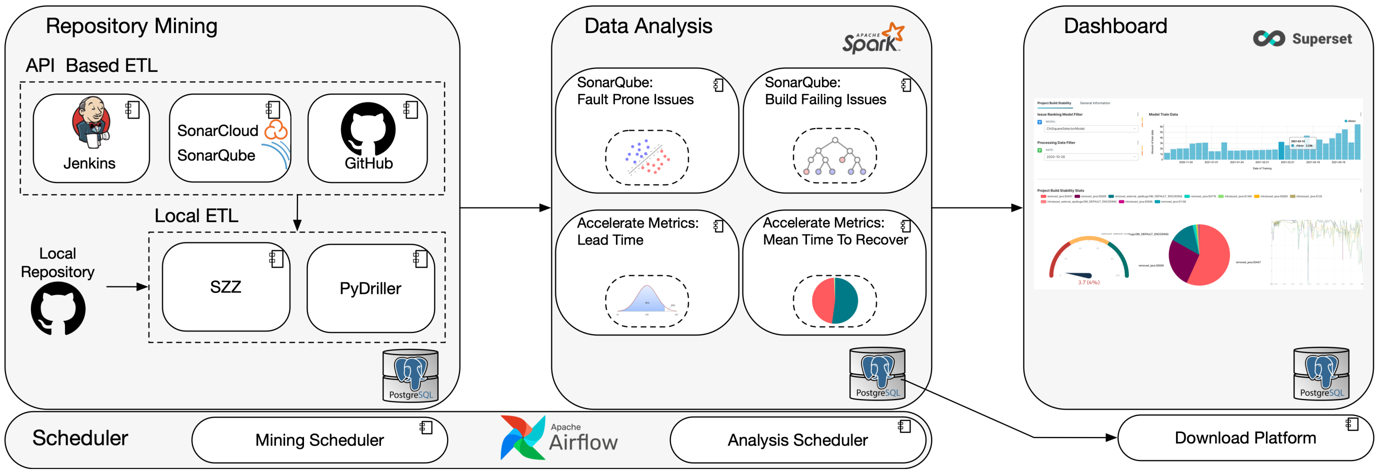

Fig. 1. PANDORA Overview

In details, PANDORA provides different benefits to:

- *Continuous Dataset Creation.* PANDORA enables to continuously mine data from repositories (e.g. GitHub), Issue trackers (e.g. Jira), and any online platform (e.g. SonarCloud). Developers can add new plug-ins to develop new connectors to collect data from any other platform or standalone tool (e.g. PyDriller [8] and Checkstyle [10]).
- *Continuous application of custom statistical and machine learning models*. Researchers can upload their python scripts to analyze the data and schedule a training frequency for their prediction models (e.g. once a month).
- *Simple and replicable data analysis approach*. Researchers do not need to know how to mine the data, but they can simply use them.
- *Data Visualization.* Dashboard for visualizing the results of the study
- Dataset export for offline usage. Data scientist and software engineers can easily download the last version of the dataset and use it for their empirical studies.

To evaluate the usefulness of our tool, we currently analyzed 365 projects (developed in different languages), continuously collecting data from December 2020 to May 2021 and running a toy-study, investigating the build-stability of SonarQube rules.

How to access to PANDORA. PANDORA is available at[11]. The complete dataset is downloadable at [12]. Please, consider that the dataset is automatically updated every three days. Moreover, the source code is freely available in the PANDORA project repository [9].

The remainder of this paper is structured as follows: Section II describes the tool, while Section III presents and the empirical validation of this tool. Section V depicts the roadmap including the next steps and Section VI draws the conclusion and highlights the future works.

[10]https://checkstyle.sourceforge.io
[11]http://sqa.rd.tuni.fi/superset/dashboard/1
[12]http://130.230.52.210/download

II. PANDORA

A. PANDORA *Architecture*

PANDORA is composed by five main components (Figure 1):

- **Repository Mining**: aimed at Extracting information from the repository, Transforming and Loading it into the database (ETL). The process is based on the ETL plugin that can be either API based, or executed on the locally cloned repositories.
- **Data Analysis**: enables to integrate data-analysis plugins that will be executed in Apache Spark [13], each using a specific methodology (Machine Learning/Statistical Analysis) to solve a specific task.
- **Dashboard**: visualization tool based on Apache Superset [14], used for inspecting and visualizing the data and the results of the analysis performed in the Data Analysis block.
- **Scheduler**: based on Apache AirFlow [15], aimed to interact with the other blocks in order to schedule the execution of (i) the repository mining, and (ii) the training/fitting of the models used in the Data Analysis block.
- **Download Platform** to enable the download of the dataset collected.

Repository Mining. This component is aimed to Extract, Transform, and Load (ETL) the information from multiple sources, including online repositories, but also local cloned repositories. In the current version, PANDORA we implemented three ETL plug-ins, to extract data from APIs: SonarCloud[16] using SonarQube[17] APIs, Jenkins[18], and GitHub[19].

We are planning to integrate locally executable tools such as Pydriller [8] and SZZ algorithm [10].

[13] https://spark.apache.org
[14] https://superset.apache.org
[15] https://airflow.apache.org
[16]https://sonarcloud.io
[17]https://www.sonarqube.org
[18]https://www.jenkins.io
[19]https://github.com

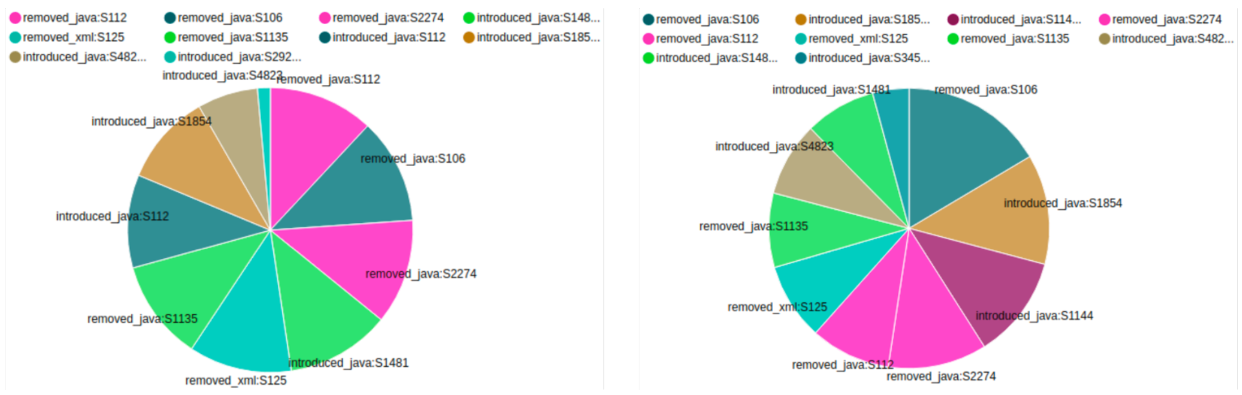

Fig. 2. Sonarqube rules build stability considering two models - Chi Square Selector model (left) and Random Forest model (right).

The ETL process is composed of three steps:

- Step 1: Extract data from Software Repositories and Platform (e.g. Jira and GitHub)
- Step 2: Transform the extracted data. In this step, we need to carefully match the data obtained from each source and merge them with the existing data in the database. As an example, some sources will provide information at the commit level. Therefore, we need to ensure to save the data to the database using the correct commit hash. Other types of data transformation are also needed, including transforming dates (timezones, formats, ...), but also aggregating data using compatible IDs (project id, project version, commit hash, ...)
- Step 3: Load the data into the back-end database for persistence (PostgreSQL), while making sure that the fields in the tabular data are in an appropriate form.

Data Analysis. The data analysis process can be executed as python script in Apache Spark[13]. Spark enables to execute analysis on a distributed cluster and it is used to prepare data, train, test and validate a range of Machine Learning and statistical models. In case the models require training, it is important to specify the training and inference interfaces, in order to enable the scheduler to trigger the training activity periodically.

Scheduler. The scheduler, based on Apache Airflow[15], is a fundamental component of PANDORA. It is responsible of periodically triggering the plugins in the repository mining and in the data analysis phase.

As an example, researchers might decide to analyze the data monthly, to update their quality models monthly, re-training them with the latest data.

Dashboard. The data visualization is performed with a highly customizable dashboard. We adopted Apache Superset [14] for this purpose. Superset gather data from the data analysis results stored in the database.

Superset enables to create new pages, tabs and to add different visualizations such as lists, histograms, pie-charts and many other type of charts.

Paltform Download. The download platform is a simple web platform that allows to download the dataset generated up to the download date for offline usage. The results are downloaded in csv format.

B. *Customizability*

One of the main caracteristics of PANDORA is its customizability. It is built in a way that it is possible to integrate additional plug-ins with relatively little effort. At the moment it is possible to easily add additional dataset, and to modify the data analysis script. A step by step guide on how to import additional datasets is available in the project repository II More functionality and the possibility to add new ETL will be added in the future.

III. PANDORA AT WORK: A CASE STUDY

In this section, we report the empirical validation of PANDORA describing the study design and the obtained results.

A. *Study Design*

In this Section, we describe the study design including goal, context, data collection, and data analysis.

Goal: The *goal* of this empirical study is to evaluate PANDORA with the *purpose* of assessing its applicability investigating a toy study. The *focus* of the toy study is to evaluate the build-stability of SonarQube rules.

Context: We analyzed all the projects of the Apache Software Foundation[20] officially using SonarCloud[16] and Jenkins[18] including 365 projects.

Data Collection: PANDORA automatically-collected data from SonarCloud, Jenkins, and GitHub for all the Apache Software Foundation projects (365 projects, developed in different languages). For this example study, we considered data collected from September 2020 to May 2021. The Apache Software Foundation page of SonarCloud lists all the projects that officially adopt it[21].

Data Analysis: We adopted a design similar to [6] comparing three common machine learning classifiers to find the

[20]https://www.apache.org

[21]Apache Software Foundation projects @Sonarcloud https://sonarcloud.io/organizations/apache/projects

TABLE I
COMPARISON BETWEEN THE STATE-OF-THE-ART TOOLS SUPPORTING MINING SOFTWARE REPOSITORIES. OUR WORK IS REPORTED IN BOLD.

Authors	Tool Name	Scope	Target Stage
Gousios [11]	GHTorent	Repositories characteristics	Data Mining
Sokol et al. [12]	MetricMiner	Repositories characteristics + Code Metrics	Data Mining + Dashboard
Spadini et al. [8]	PyDriller	Repositories characteristics + Historical analysis	Data Mining
Dueñas et al. [13]	GrimoireLab	Process metrics	Data Mining + Dashboard
Nguyen et al.	**PANDORA**	**Customizable**	**Data Mining + Data Analysis + Dashboard**

SonarQube issues (independent variable) that affect the build stability (dependent variable), by classifying build stability as successful or not.

The models chosen are Logistic Regression [14], Decision Tree [15] and the Random Forest model [16].

We trained the three classifiers using 80% of the dataset and we validated them using the remaining 20%. We evaluate the accuracy of the models using considering the Receiver Operating Characteristic and its Area Under the Curve (AUC-ROC) and the F-measure, and we finally selected the classifier that provides the highest overall accuracy. For the purpose of the evaluation, the system is configured to train the model once a day considering the data of the last 12 months. However, the system can be configured to train the models less often (e.g. once a month).

B. Validation Results

The best performing model, considering all the metrics, is the Random Forest, with an AUC-ROC of 93.53% and F-measure of 67.53%. The random forest was therefore used to find which issues affect the build stability. The classifier automatically calculates the importance of the features it is using for the prediction, therefore we took the feature selected by the random forest itself. In order to validate the results, we also calculated the issues affecting build stability through a Chi Square Selector model. The results of the two models can be found in Figure 2.

The results obtained with the two models are similar to each other. The issues considered responsible for the build quality were therefore selected from the issues found responsible from both the models. In total 8 rules were found to affect the build stability according to both the random forest model and the Chi Square Selector model.

It is interesting to note that the SonarQube rules list that affects the build stability change significantly if considering older data. This confirms that the results of quality models might change over time, and therefore need a continuous approach to train the data periodically and to learn from developers.

The training was executed every three days from September 2020 up to now, considering the data of the previous 12 months. It is interesting to see that the training of such a large amount of data on a distributed Apache Spark cluster usually takes no more than 4 hours. However, we are planning to increase the time between training sessions as soon as we will add more plug-ins. The results of this study show that it is possible to continuously collect and update MSR studies, enabling researchers and practitioners to easily keep track of the changes.

IV. RELATED WORK

Over the last years, researchers have devoted significant effort to developing publicly accessible tools to support mining software repositories studies. This section reports and discusses some of the most used and best-known tools supporting data mining and data analysis, along with highlighting the main differences between such tools and PANDORA. A quick summary of the main differences can be found in Table I.

Gousios [11] presented GHTorent, a mirror of GitHub providing queriable data about repositories hosted on GitHub, e.g., pull requests, issues, etc. Differently from PANDORA, GHTorent only provides support for data mining with a specific focus on repository-related information.

Sokol et al. [12] presented MetricMiner, a web application providing support to mining software repository studies with features for the automatic clone and metric extraction from Git and SVN repositories. MetricMiner also provides the opportunity of visualizing data in a dashboard and performing some statistical calculations by means of R scripts that are automatically written and executed. The main differences between this tool and PANDORA lie in the fact that (i) PANDORA is able to extract a wider set of metrics, and PANDORA also integrates a customizable Data Analysis component useful to run Machine Learning and other sophisticated statistical models. Moreover, according to the links provided in the original publication, MetricMiner seems to be no longer available and maintained.

Later on, Spadini et al. [8] released PyDriller, a Python Framework that supports mining historical and repository-related information on Git repositories. PyDriller only focuses on data mining without providing support for data analysis and data visualization.

Finally, GrimoireLab, by Dueñas et al. [13], is a toolset integrating modules to retrieve process-related information from multiple types of software repositories. It also provides a dashboard for the interactive visualization of the data retrieved. This tool only focuses on data mining and data visualization, without providing support for data analysis.

With respect to the aforementioned tools, PANDORA differs for its customizable nature, which allows users to tailor their own experiments as they prefer, as well as for providing support at all the mining software repository process stages.

in particular to platforms that delete the data periodically

V. Roadmap

We are planning to continue the development of PANDORA, to make it accessible to as many researchers as possible.

The next planned activities are:

- Development of new Repository Mining plugins. We are actively working on the integration of SZZ, PyDriller and Spotbugs[22]. Moreover, we are also planning the integration of issue tracking systems, including Jira, BugZilla and GitHub Issues.
- Development of data analysis snippets to easily develop new plug-ins using the most common data analysis techniques
- Development of a simpler installation process, based on Infrastructure as a code, and including a cloud-based installation script to enable researchers to install the platform on amazon AWS.
- Validation of the tool with an in-vivo assessment involving real researchers working in the field of Mining Software Repositories. We plan to perform a twofold validation: on the one hand, we aim at evaluating PANDORA's features and their ease of use by means of a task simulating the entire MSR process on the tool; on the other hand, we plan to ask participants to plug-in their own components into the different pipeline steps in order to assess PANDORA's customizability.

We are also planning to publish all the results of our empirical studies on PANDORA, so as to show how the results might change over time.

VI. Conclusion and Future Work

This paper introduces PANDORA an extensible mining software repository platform to continuously extract data from software repositories and continuously update the result of empirical studies.

PANDORA enables researchers to concentrate on the data analysis phase, relying on common dataset of data continuously collected by PANDORA. Moreover, PANDORA enables practitioners to see the most updated results of empirical studies, and also to apply them on their code.

PANDORA has currently a limited amount of plug-ins. However, new plug-ins will be developed by our team in the near future, and other researchers might contribute too to the development of other plugins, enriching the platform and enabling more stakeholders to access to their results thanks to a graphical and continuously updated dashboard.

References

[1] K. Chaturvedi, V. Sing, and P. Singh, "Tools in mining software repositories," in *Int. Conf. on Computational Science and Applications*, 2013.

[2] F. Z. Sokol, M. F. Aniche, and M. A. Gerosa, "Metricminer: Supporting researchers in mining software repositories," in *Int. Working Conference on Source Code Analysis and Manipulation*, 2013, pp. 142–146.

[3] A. Zaidman, B. Van Rompaey, S. Demeyer, and A. van Deursen, "Mining software repositories to study co-evolution of production test code," in *International Conference on Software Testing, Verification, and Validation*, 2008, pp. 220–229.

[4] M. Van Antwerp and G. Madey, "Advances in the sourceforge research data archive," in *Workshop on Public Data about Software Development (WoPDaSD)*, 2008, pp. 1–6.

[5] M. Ortu, G. Destefanis, B. Adams, A. Murgia, M. Marchesi, and R. Tonelli, "The jira repository dataset: Understanding social aspects of software development," in *International conference on predictive models and data analytics in software engineering*, 2015, pp. 1–4.

[6] V. Lenarduzzi, F. Lomio, and H. Huttunen, "Are sonarqube rules inducing bugs?" in *International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2020.

[7] V. Lenarduzzi, N. Saarimäki, and D. Taibi, "Some sonarqube issues have a significant but small effect on faults and changes. a large-scale empirical study," *Journal of Systems and Software*, vol. 170, 2020.

[8] D. Spadini, M. Aniche, and A. Bacchelli, "Pydriller: Python framework for mining software repositories," in *European Software Engineering Conference on the Foundations of Software Engineering*, 2018.

[9] H. Nguyen, F. Lomio, and V. Lenarduzzi, "Pandora github repository," https://github.com/clowee/PANDORA, 2021.

[10] J. Śliwerski, T. Zimmermann, and A. Zeller, "When do changes induce fixes?" *SIGSOFT Softw. Eng. Notes*, vol. 30, no. 4, pp. 1–5, May 2005.

[11] G. Gousios, "The ghtorent dataset and tool suite," in *Working Conference on Mining Software Repositories (MSR)*, 2013, pp. 233–236.

[12] F. Z. Sokol, M. F. Aniche, and M. A. Gerosa, "Metricminer: Supporting researchers in mining software repositories," in *International Working Conference on Source Code Analysis and Manipulation (SCAM)*, 2013, pp. 142–146.

[13] S. Dueñas, V. Cosentino, J. M. Gonzalez-Barahona, A. del Castillo San Felix, D. Izquierdo-Cortazar, L. Cañas-Díaz, and A. Pérez García-Plaza, "Grimoirelab: A toolset for software development analytics," vol. 7, no. e601. [Online]. Available: https://doi.org/10.7717/peerj-cs.601

[14] D. R. Cox, "The regression analysis of binary sequences," *Journal of the Royal Statistical Society. Series B*, vol. 20, no. 2, 1958.

[15] L. Breiman, J. Friedman, C. J. Stone, and R. Olshen, *Classification and regression trees Regression trees*. Chapman and Hall, 1984.

[16] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[22]SpotBugs https://spotbugs.github.io