Journal of Data Acquisition and Processing
ISSN: 1004-9037
https://sjcjycl.cn/
DOI: 10.5281/zenodo.7660401

PROGRAMMING TOOL FOR ASSEMBLING SOFTWARE BUGS THROUGH OPEN SOURCE REPOSITORY

Shallu Juneja#1*,Gurjit Singh Bhathal#2, Brahmaleen K. Sidhu#3
[1]Research Scholar, Department of CSE, Punjabi University, Patiala
shallujuneja9@gmail.com
ORCID - 0000-0001-6451-7541
[2]Assistant Professor, Department of CSE, Punjabi University, Patiala
gurjit.bhathal@gmail.com
ORCID - 0000-0002-4762-4617
[3]Assistant Professor, Department of CSE, Punjabi University, Patiala
brahmaleen.sidhu@gmail.com
ORCID - 0000-0001-6519-7957
*Corresponding Author – Shallu Juneja

Abstract
Several open source repositories are available through which assembling of relevant data is of utmost importance for software developers to save time and cost. These open source software repositories such as JIRA, BUGZILLA, PROMISE, NASA, Trac, Mantis and so on consist of various software projects along with their attributes like Bug ID, Bug Priority, One-line description of software Bug etc. In this paper we have described the programming tool for assembling software faults or bugs through open source repository. This programming tool is implemented in javascript which uses puppeteer API to open chromium browser(open source version) and makes HTTP request to the project repository to assemble relevant dimensions. We conducted a successful gathering of bug information from over hundreds of Apache projects including Aardvark, Accumulo, Hadoop, Lucene and others.All relevant dimensions are assembled which includes attributes of software bugs such as software bug ID, its description, type , Priority etc. Resultant reports are useful for resolving various issues related to software fault prediction.
Keywords: Software Repository, Puppeteer API, Software Bugs, Bugs Attributes

1. INTRODUCTION

Bug reports are used to document the bugs. A bug report is a written representation of a flaw or issue that occurred while using the product. Each bug report includes details like the date, versions, bug identifier, bug description, severity, priority, and other characteristics. Additionally, it includes a thorough description of a bug in plain language, which aids researchers in their analysis of bug reports. Prior to the start of the actual testing process, software fault prediction seeks to detect fault-prone software modules utilizing some fundamental characteristics of the software project. It assists in achieving target software quality with less expense and work. Software professionals and the IT sector have long regarded software fault prediction as a crucial problem. For the traditional methods to work, a malfunctioning module or prior expertise with faults are required in order to discover software errors within an application. Using machine learning techniques, an automated software fault recovery model enables the program to significantly predict and repair software errors. The program operates more efficiently and has fewer errors, which saves time and money. There are numerous bug tracking tools available, including Jira, Bugzilla, Trac, and Mantis.

There are various types of software repositories as private, semiprivate and public repositories (Radjenovic et al. 2013).

Private/commercial Repository:-In such type of repositories neither the source code nor the fault dataset are accessible. Companies within organizational use maintain and use this kind of dataset. It might not be possible to repeat the study based on these datasets.

Public/freeware Repository:- Only the project's source code and defect information are accessible in this kind of repository. Typically, the metric values are not accessible. As a result, it necessitates that the user computes the metric values from the source code and convert them to the accessible fault data. Since determining metric values and mapping their defect information is a crucial activity, this method calls for extra caution. Any mistake can result in biased learning.

Public Repository:- The value of the metric and the defect information are both made available to the public in this sort of repository (Ex. NASA and PROMISE data repositories). Repeatable research can be conducted utilizing the datasets.

JIRA: Jira is a collection of agile work management tools that enables collaboration amongst all teams from concept to customer, giving you the freedom to work with others to produce your best work. Jira provides a variety of tools and deployment choices that are designed specifically for software, IT, business, operations teams, and other groups.

BUGZILLA : Bugzilla is an open source bug tracking tool that is updated and maintained by the Mozilla Foundation. It enables development and testing teams to track bug fixes and code changes in tasks including the creation and deployment of software and apps.

PROMISE: The software engineering community at large, as well as researchers creating predictive software models (PSMs), will find a collection of publicly accessible datasets and tools here. The repository was established to promote repeatable, verifiable, debatable, and/or improved software engineering predictive models.

MANTIS Bug Tracker: MantisBT is an open source issue tracker that provides a delicate balance between simplicity and power. Users are able to get started in minutes and start managing their projects while collaborating with their teammates and clients effectively.

TRAC: A web-based, open-source system for tracking bugs and managing projects is called Trac. It has been adopted by a number of organisations for use as a bug tracking system for both proprietary projects and products as well as free and open-source software.

NASA Open Source Development: NASA runs its popular web-based social code and revision control tool as a public repository.

APACHE PROJECTS: The Apache projects stand out for their collaborative, consensus-based development approach and open, practical software licence, which permits developers who acquire the software for free to redistribute it under non-free conditions.

The programming tool for compiling software errors or bugs using an open-source repository has been discussed in this work. This java script-based programming tool opens the open-source Chrome browser using the puppeteer API and sends an HTTP request to the project repository to retrieve the necessary dimensions.

We collected bug data successfully from more than a thousand Apache projects.

2. LITERATURE REVIEW

- By foreseeing defects before the testing process, software fault prediction reduces the need for fault discovery activities. Additionally, it facilitates the more effective use of testing resources and streamlines software quality assurance (SQA) initiatives to be used in the later stages of software development. The practical importance of software defect prediction has drawn a great deal of attention to this field during the past two decades. The researchers have been inspired to conduct study and come to broad findings by the accessibility of open-source data sources like NASA and PROMISE. (Rathore et al 2019).
- A bug reporting collection system has been created that may be used to gather bugs from the free Jira bug tracking system (for Apache projects) and generate reports on all the properties of the bugs. The information acquired can be applied to many different things, such as classifying issues based on their one-line and lengthy descriptions and then categorising them as concurrency, security, or semantic bugs, or predicting their severity using machine learning methods. (Kaur et al 2017)
- The process of creating software is both technical but also quite collaborative. A sociological examination of the players, artefacts, and activities may therefore be helpful for gaining a complete knowledge of this process. This served as the foundation for a series of studies we conducted on the Free/Open Source Software (F/OSS) development processes. (Ekbia et al)
- Software Fault Prediction has a wide range of potential applications and difficulties, which will benefit academics in the future. Some of the difficulties include the employment of several machine learning technologies to anticipate problems with the best outcomes, the trending study topic of cross-company predictions, the improvement of datasets through feature extractions, and others. Class Imbalance issues with datasets need to be tackled utilising Agile based approaches so that ensemble models can be developed in the future to improve performance.(Juneja et al 2022)
- It can be difficult to estimate development effort accurately. Predicting development effort has been done using data mining and other methods. These methods do, however, encounter technical difficulties, particularly as software repositories grow in size on a daily basis.(Tariq et al 2020)
- For the purpose of extracting the bug data via web interfaces, we established a schema and put an automated method into place in this article. We have also performed trials with several XML and HTML parsers.(Yuk, Y. et al 2013)
- With the help of bug information and code features, this study suggests a novel technique for locating related defects in source code. It locates and extracts code features of bug method from source code after first extracting bug features from bug information in bug repositories.(Wang et al 2010)
- By selecting 2,060 real-world defects from three significant, representative open-source projects—the Linux kernel, Mozilla, and Apache—we can examine the characteristics of software bugs. We manually investigate the root causes, impacts, and components of these

problems. In the future, we'd like to research software bugs in other languages, including Java, to understand how language influences bug characteristics.(Tan L et al 2014)

- The work in this study does not create any such automated system for online classification; instead, it merely offers to apply the proposed approach to an offline database that has been downloaded from the Bugzilla repository. As a result, research may be done to make the system available for the classification of bug reports in real time. (Sharma G et al 2015)
- The ability to automate software security analysis and assessment is gaining traction, thanks in part to the changing nature of how software is delivered to users.(Sadeghi, A et al 2014)
- BUMPER (BUg Metarepository for dEvelopers and Researchers) is a shared infrastructure for developers and researchers interested in mining data from multiple (heterogeneous) repositories.(Nayrolles, M et al 2016)
- Text classification was used in this study to assess the severity levels of defect reports and predict the severity levels of unseen defect reports in real time. (Malhotra et al 2013)
- This paper demonstrates that the severity of a bug can be predicted using other information contained in a bug report, specifically the textual information describing the bug.(Lamkanfi et al 2010)
- Our method effectively automates the identification of SBRs, which would otherwise necessitate significant effort on the part of security engineers to manually assess each BR in a BTS to determine which BRs are SBRs.(Gegick et al 2010)

3. METHODOLOGY USED

Reports are assembled consisting of bug attributes. Using the Puppeteer API, we have launch the Chromium browser (the open-source version of Google Chrome) and made an HTTP request to a project repository from within an async function. Once there, we have used DOM element selectors to select all of the project elements on the page. We have made HTTP requests to all of these projects and assembled the relevant dimensions, including the bug ID, description, type, and priority, for all of the bugs. Once all of the data is collected, it is stored in a JSON object. Using the Node.js file system module, we have save this JSON object to a file on the computer at a specified file path.

Finally, we have converted the JSON object to an Excel file using the json2xls library. Figure 1 depicts the entire process of assembling Bug reports from thousands of projects from the open source repository JIRA.

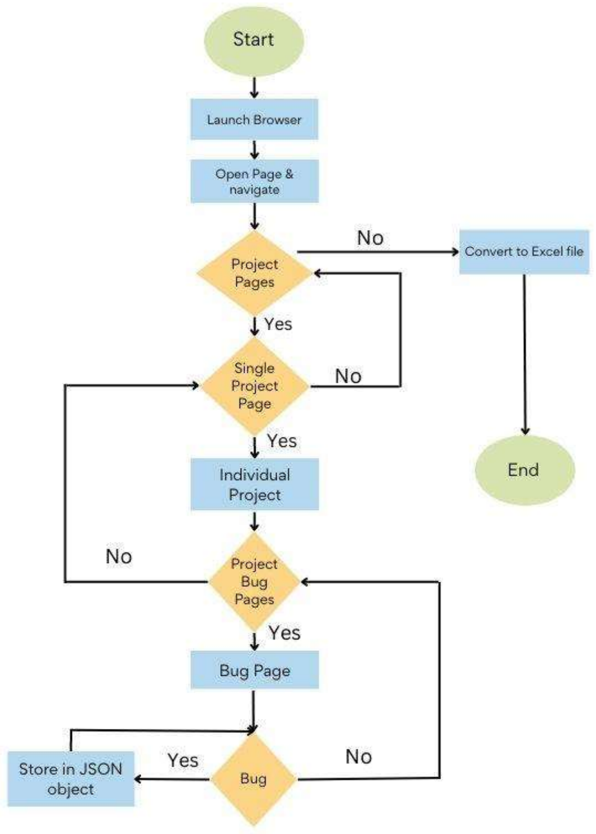

Figure 1

4. TECHNICAL EXPLANATION

Puppeteer : Puppeteer is a Node.js library that allows you to control Chrome or Chromium via the DevTools Protocol. It can be used to automate tasks, scrape websites, and carry out other tasks that would be difficult or time-consuming to carry out manually. Puppeteer is used in this code to launch a browser, open new tabs, and navigate to specific pages.

FS :The fs (File System) module is a Node.js built-in module that offers an API for interacting with the file system. It can read and write files, create and delete directories, and carry out other file-related tasks. The fs function is used in this code to write the contents of the jsondata array to a JSON file.

Express : Express is a Node.js web framework that provides a set of robust features for building web applications. It is intended to make it simple to create APIs and other types of web services, and it can be used in conjunction with a variety of templating engines to create full-featured web applications. Express is used in this code to start a server that will listen on port 5050.

Json2xls : json2xls is a library for converting JSON data to an Excel file. It is capable of converting JSON data into an Excel spreadsheet, which can then be downloaded or saved to the file system. Json2xls is used in this code to convert the contents of the JSON file created earlier in the code to an Excel file.

1. **Pseudocode**

```
PROCEDURE main()
has_next_button <- true
WHILE has_next_button == true
projectelementHandleArray <- tab.$$('.cell-type-name a')
FOR EACH project_element IN projectelementHandleArray
link_suffix <- project_element.getAttribute("href")
link_per_project(newTab, "https://issues.apache.org" + link_suffix)
END FOR
END WHILE
today <- new Date()
dd <- String(today.getDate()).padStart(2, '0')
mm <- String(today.getMonth() + 1).padStart(2, '0')
yyyy <- today.getFullYear()
today <- dd + mm + yyyy
jsonfilepath <- `./${today}.json`
fs.writeFileSync(jsonfilepath, JSON.stringify(jsondata))
convert <- function () {
xls <- json2xls(jsonfile)
fs.writeFileSync(filename, xls, 'binary', (err) => {
IF err THEN
console.log("writeFileSync :", err)
ELSE
console.log( filename+" file is saved!")
END IF
})
}
END PROCEDURE

PROCEDURE link_per_project(newTab, project_Link)
newTab.goto(project_Link)
next_available <- "true"
WHILE next_available == true
bugs <- newTab.$$('.issue-content-container')
FOR EACH bug IN bugs
TRY
bug.click()
bug_id_element <- newTab.$("#key-val")
bug_description_element <- newTab.$("#summary-val")
bug_label_element <- newTab.$(".labels-wrap.value .labels")
Catch
console.log(Error Description)
```

```
        continue;
    let json <- {project_name, bug_id, bug_description}
    END FOR
END PROCEDURE
```

5. RESULTS

The Programming Tool collects bug reports from various Apache projects using the Jira Repository and generates data in the form of various reports. Jira REST APIs are used to interact with Jira remotely. It offers a consistent interface for interacting with Jira and its other applications. JSON is the input and output format for Jira REST APIs.

Table 1: Attributes of Software Bug

Bug Attribute	**Description**
Bug Id	Uniquely identifies the Bug.
Summary	A succinct but informative summary of the problem
Priority	The issue's priority is appropriate. It can be a blocker, a critical, a major, a minor, or a trivial issue.
Components	Rrepresents all relevant parts of the bug.
Affect version(s)	Version where the problem was noticed
Fix version	the problem is expected to be resolved
Assignee	the problem that the project owner was assigned
Resolution	represents whether the issue is resolved or not
Environment	Contains essential information about the setting where the problem occurs.
Description	describes every aspect of the problem, including how to duplicate it and a potential fix.

.

Table 2: Generated Bug Attribute Sample of Lucene Project

Attribute Name	**Attribute Value**
Project Name	Lucene – Core
Bug ID	LUCENE-10665
Bug Description	Deadlock in Analysis SPI Loader
Bug Type	Bug
Bug Priority	Critical

Bug Label	None
Bug Status	Patch Available
Bug Resolution	Unresolved
Bug assignee	Unassigned
Bug Reporter	Jasir KT
Bug Created	28/Jul/22 10:37
Bug Updated	05/Nov/22 10:54

The table 1 shows various attributes of software bugs along with their description. These attributes are general attributes of software bugs or faults which are needed during prediction of software faults in turn saving cost and time needed for development of software projects by software developers. The table 2 shows generated sample attributes of software bug of Lucene project through open source repository and can be utilized for predicting software faults and its other issues such as severity of software bugs etc.

6. CONCLUSION AND FUTURE SCOPE

The goal of this research is to use this programming tool in the future to report various requirements and maintenance requests. We collected bug data successfully from more than a thousand Apache projects. We are able to compile all pertinent information, such as the Bug ID, description, kind, and Priority. The resulting reports are helpful for addressing a variety of software failure prediction-related problems. In order to confirm its efficacy and efficiency, we will conduct more experiments using the programming tool on significant open source projects in the future.

7. REFERENCES

1. Radjenović, D., Heričko, M., Torkar, R., & Živkovič, A. (2013). Software fault prediction metrics: A systematic literature review. *Information and software technology*, *55*(8), 1397-1418.
2. Rathore, S. S., & Kumar, S. (2019). A study on software fault prediction techniques. *Artificial Intelligence Review*, *51*, 255-327.
3. Kaur, A., & Jindal, S. G. (2017, January). Bug report collection system (BRCS). In *2017 7th International Conference on Cloud Computing, Data Science & Engineering-Confluence* (pp. 697-701). IEEE.
4. Ekbia, H., & Gasser, L. Common Ground: For a Sociology of Code. *Submitted to Information Technology and People, special issue on Social Theory [Електронний ресурс].–Режим доступу: http://130.203*, *133*.
5. Shallu Juneja, Gurjit Singh Bhathal and Brahmaleen K Sidhu(2022). Current Trends And Literature Review Of Machine Learning Models For Predicting Software Fault Based On Textual And Numeric Data. *Applied data Science and Smart Systems conference ,*Punjab, India 2022

6. Tariq, S., Usman, M., & Fong, A. C. (2020). Selecting best predictors from large software repositories for highly accurate software effort estimation. *Journal of Software: Evolution and Process*, *32*(10), e2271.
7. Yuk, Y., & Jung, W. (2013, June). Comparison of extraction methods for bug tracking system analysis. In *2013 International Conference on Information Science and Applications (ICISA)* (pp. 1-2). IEEE.
8. Wang, D., Lin, M., Zhang, H., & Hu, H. (2010, July). Detect related bugs from source code using bug information. In *2010 IEEE 34th Annual Computer Software and Applications Conference* (pp. 228-237). IEEE.
9. Tan, L., Liu, C., Li, Z., Wang, X., Zhou, Y., & Zhai, C. (2014). Bug characteristics in open source software. *Empirical software engineering*, *19*, 1665-1705.
10. Sharma, G., Sharma, S., & Gujral, S. (2015). A novel way of assessing software bug severity using dictionary of critical terms. *Procedia Computer Science*, *70*, 632-639.
11. Sadeghi, A., Esfahani, N., & Malek, S. (2014). Mining the categorized software repositories to improve the analysis of security vulnerabilities. In *Fundamental Approaches to Software Engineering: 17th International Conference, FASE 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014, Proceedings 17* (pp. 155-169). Springer Berlin Heidelberg.
12. Nayrolles, M., & Hamou-Lhadj, A. (2016, March). BUMPER: a tool for coping with natural language searches of millions of bugs and fixes. In *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)* (Vol. 1, pp. 649-652). IEEE.
13. Malhotra, R., Kapoor, N., Jain, R., & Biyani, S. (2013). Severity assessment of software defect reports using text classification. *International Journal of Computer Applications*, *83*(11), 13-16.
14. Lamkanfi, A., Demeyer, S., Giger, E., & Goethals, B. (2010, May). Predicting the severity of a reported bug. In *2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)* (pp. 1-10). IEEE.
15. Gegick, M., Rotella, P., & Xie, T. (2010, May). Identifying security bug reports via text mining: An industrial case study. In *2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)* (pp. 11-20). IEEE