Software Defect Prediction with Bug-Code Analyzer - a Data Collection Tool Demo

Goran Mauša #1, Tihana Galinac Grbac #2, Bojana Dalbelo Bašić *3

* Faculty of Electrical Engineering and Computing, University of Zagreb
Unska 3, 10000 Zagreb, Croatia
* bojana.dalbelo@fer.hr

**Faculty of Engineering, University of Rijeka
Vukovarska 58, 51000 Rijeka, Croatia
1goran.mausa@riteh.hr
2 tihana.galinac@riteh.hr

Abstract—Empirical software engineering research community aims to accumulate knowledge in software engineering community based on the empirical studies on datasets obtained from the real software projects. Limiting factor to building the theory over thus accumulated knowledge is often related to dataset bias. One solution to this problem is developing a systematic data collection procedure through standard guidelines that would be available to open community and thus enable reducing data collection bias. In this paper we present a tool demonstration that implements a systematic data collection procedure for software defect prediction datasets from the open source bug tracking and the source code management repositories. Main challenging issue that the tool addresses is linking the information related to the same entity (e.g. class file) from these two sources. The tool implements interfaces to bug and source code repositories and even other tools for calculating the software metrics. Finally, it offers the user to create software defect prediction datasets even if he is unaware of all the details behind this complex task.

Keywords—Automated Tool, Mining Software Repositories, Software Defect Prediction

I. INTRODUCTION

Software Defect Prediction (SDP) deals with resource management in software quality assurance. It predicts the location of defects in source code based on experience from the past. The collection of data from past experience is often time consuming and therefore expensive. It involves data collection from two separate processes that use separate repositories: the bug tracking and the source code committing. These repositories’ purpose does not include the SDP. Their structures are not clearly linked representing another great obstacle and causing the use of SDP in software development life cycle to be extremely rare [1]. It is important to stress that the techniques used for SDP purposes are very sensitive on dataset bias. There is an evident need for a systematic procedure that can be used to automate the data collection process. This paper present the Bug-Code (BuCo) Analyzer as a first step in that direction. It collects the bug data from bug tracking repository and the data from source code management repository, links them appropriately and analyzes the source code into a single dataset. The SDP dataset contains two crucial categories of information: the software metrics (like size and complexity measurements) and the number of bugs for each software module (most often for each file). The main contribution of this tool is standardized and automated SDP data collection for which the user does not have to be aware of the projects’ specificities nor the details of intermediate and complex analyses running in the background of data collection. The target of data collection are open source projects due to their availability of both repositories.

We need accurate SDP datasets to continue our research on influence of data imbalance on SDP models [2]. We were motivated to construct the data collection tool by the fact that publicly available SDP dataset repositories often suffer from quality issues [3], [4]. The lack of a clear and detailed description of data collection process make such datasets unable to replicate and validate their quality which is an important feature [5]. There are some researchers that briefly described the data collection procedure they used on open source projects [6], [7]. However, they constructed only a small number of datasets, containing lower number of software metrics and analyzing only certain releases. In order to identify all the issues in a SDP data collection procedure from open source projects we conducted an exploratory study. The result of that study was a data collection procedure that we implemented in the BuCo Analyzer. The BuCo Analyzer is able to collect SDP datasets from all releases of all projects written in java that used Bugzilla as the bug tracking repository and GIT as the source code management repository.

First analyses of BuCo tool are done in [8]. We compared our tool with the more complex technique used by the ReLink tool which only purpose is bug-code linking [9]. We notice that our regular expression technique performs even better upon a benchmark dataset.

II. THE BUG-CODE ANALYZER

Fig. 1 presents the tool architecture. External interfaces to Bugzilla and GIT allow the tool to collect the raw data, while the other two tools perform the calculation of software metrics. The process of
connecting the externally collected data is the essential internal part of the tool. The BuCo Analyzer offers a number of features in each step of the data collection process. Following subsections give an overview of these features and the tool benefit.

A. Bug Repository Collection

The tool offers the user to download bugs from the Bugzilla repository of three major open source communities: Eclipse, Mozilla, and Apache. Its features are following:

- The choice of (Classification, Product, Version) mirrored from the Bugzilla repository
- Bug collection of selected project and its releases in local database
- The collected bugs are resolved and closed as fixed and caused a loss in functionality

Although all the three major communities use the Bugzilla, there are some inconsistencies and each is approached differently. However, our user does not have to be aware of any such details.

B. Source Code Repository Collection

The tool offers the user to collect the source code of the project selected in previous step from the GIT repository. Its features are following:

- Automatic search for GIT sub-repositories, release tags and date
- Automatic download of each project to the local machine

These steps usually require manual search for multiple results, take long time when having large projects with many releases, often contain ambiguous data (e.g., release tag in GIT not equal to version in Bugzilla) and thus are prone to human error. The standardized data collection procedure implemented in our tool bypasses these errors. Furthermore, we found no other tool that performs such a task.

C. Bug - Commit Linking

There is a number of techniques that can be used to perform the linking between bugs and commits. That is why this step is most prone to bias. With BuCo Analyzer, the user can choose any combination of following techniques:

- Regular expression search for Bug ID and predetermined surrounding set of characters
- Authorship correspondence of the bug assignee and the commit developer
- Time correlation with custom tolerance in the number of days between closing the bug and committing the fix
- Key term included or excluded in the commit message

The regular expression was formed after an exhaustive manual investigation of simpler techniques' performance and the key term exclusion technique is a novel one.

D. Source Code Analysis

The tool calculates the software product metrics for all the source code files of the selected releases in a project using the external interfaces and stores the data in database using internal interfaces. Its main feature is:

- Automatic processing of each project release

There are various tools that can calculate the metrics. However, not all of them are suitable for automation that is obligatory for making the analysis of multiple projects and their releases in reasonable time. We implement two such tools, calculating up to 50 metrics for each file. The tool also analyzes whether the files exist in subsequent release and whether they are changed in any of the calculated metric.

E. SDP Dataset Generation

This final step is automated, but only after all the previous steps are done. The tool offers the user to generate reports on any of the used metrics and the result of these analyses:

- SDP datasets for all the analyzed projects and their releases
- General report calculating the quantity of analyzed data
- Custom report for specific metrics and their values
- Statistical report for metric values (min, max, median, interquartile range, normality test, correlation matrix)

III. CONCLUSION

BuCo Analyzer offers all the functionalities necessary to create the SDP datasets from the bug tracking and the source code management repositories. To the best of our knowledge, it is the only tool that provides such a complete process. Most of the steps are fully automated, making the otherwise time-consuming and difficult tasks less demanding. In future work, we plan to improve the search for untraceable bugs and include the SDP data mining modules.

ACKNOWLEDGMENT

The work presented in this paper is supported by the University of Rijeka research grant 13.09.2.2.16.

REFERENCES