Exploring Software Library Metrics with Repository Badges

by

Monica Bui
University of Alberta
bui1@ualberta.ca

Contents

List of Figures

1 Introduction

Libraries, Frameworks, and Application Programming Interfaces (APIs) provide developers a way to reuse existing functionalities built by someone else without having to re-implement already built features. Given a large collection of libraries out there, it is often difficult and not clear how to select the best one to use for your own project.

Developers may resort to first doing a general search of their desired library features with search results indicating various resources such as a Q&A website like StackOverflow [1] or a website that hosts open source libraries such as Github [2].

Previous research has examined different, inner aspects of libraries that may have mentioned in the above online resources [3, 4, 5, 6, 7]. These aspects or defined

2 Related Work

2.1 Library Selection Goals

2.2 Metrics

2.2.1 Quality Assurance

2.2.2 Community Support

2.2.3 Repository General Information

2.3 Summary

3 Background

3.1 Badges

3.2 Online Badge Services

4 Badge Implementation

4.1 Overarching Structure of Scripts

4.2 Security

4.2.1 Definition

The security badge is inspired by Mora's et. al [5] security metric implementation. However, there are limitations related to classifying some security vulnerabilities due to inaccurate issue descriptions. These inaccuracies would suggest that there was a security problem but in reality was another issue altogether. To avoid this conflict brought by issue descriptions, we propose to use another existing tool called SpotBugs [8]. From the SpotBugs [8] website description itself, the program *uses static analysis to look for bugs in Java code.* In conjunction with the FindSecBugs [9] plugin to provide a larger data set of security bug patterns to look for, both these tools will allow for greater accuracy of targeting security bugs.

The security badge represents the number of security bugs reported by SpotBugs [8] with the FindSecBugs [9] plugin. We filter for only security bug patterns and configure SpotBugs to only classify bugs on the highest confidence setting with maximum effort toggled on to increase precision.

4.2.2 Implementation

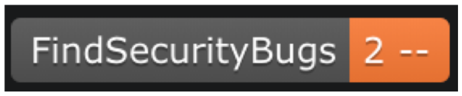

Figure 1: Example of Security Badge

First, we have a text file that holds open source Java library links hosted on GitHub [2]. A shell script clones and compiles them respectively under Gradle [10] or Maven [11]. After compilation, a script runs SpotBugs and FindSecBugs per library under our defined configured settings and stores the respective result into the database. The client can hit the security script endpoint to retrieve the saved results which then can be placed into Shields.io to output the security badge with an example shown in figure 1.

4.3 Last Discussed on Stack Overflow

4.3.1 Definition

4.3.2 Implementation

4.4 Issue Response Time

4.4.1 Definition

4.4.2 Implementation

4.5 Contributor Pull Request Merge Rate

4.5.1 Definition

4.5.2 Implementation

4.6 Release Frequency

4.6.1 Definition

4.6.2 Implementation

5 Evaluation

5.1 Open Source Evaluation

5.2 Future Work Evaluation

6 Threats to Validity and Limitations

6.1 Validity

6.2 Limitations

7 Conclusion and Future Work

References

[1] Stack-Overflow, “Stack overflow,” http://www.stackoverflow.com.

[2] GitHub, “Github,” ”http://github.com/”.

[3] A. Trockman, S. Zhou, C. Kästner, and B. Vasilescu, “Adding sparkle to social coding: An empirical study of repository badges in the npm ecosystem,” in *International Conference on Software Engineering*, ser. ICSE. ACM, 2018, pp. 511–522.

[4] A. C. Hora and M. T. Valente, “Apiwave: Keeping track of api popularity and migration.” in *ICSME.* IEEE Computer Society, 2015, pp. 321–323.

[5] F. L. de la Mora and S. Nadi, “Which library should I use?: a metric-based comparison of software libraries,” in *ICSE (NIER).* ACM, 2018, pp. 37–40.

[6] G. Uddin and F. Khomh, “Opiner: An opinion search and summarization engine for apis,” in *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Oct 2017, pp. 978–983.

[7] C. Chen, S. Gao, and Z. Xing, “Mining analogical libraries in q amp;a discussions – incorporating relational and categorical knowledge into word embedding,” in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, vol. 1, March 2016, pp. 338–348.

[8] SpotBugs, “Spotbugs,” ”https://spotbugs.github.io/”.

[9] FindSecBugs, “Findsecbugs,” ”https://find-sec-bugs.github.io/”.

[10] Gradle, “Gradle,” ”https://gradle.org/”.

[11] Maven, “Maven,” ”https://maven.apache.org/”.