LOG6307E – Release Eng. App. Mining Software Repositories

Fall 2023

Paper review of "Programmers' Build Errors: A Case Study (at Google)"

Group 01

1949477 – Ming Xiao Yuan

Submitted to: Zohreh Sharafi

September 19, 2023

Summary:

The programmers' usually follow a development process called "edit-compile-test". It usually involves the programmers write some code, compile it and testing the results. It is often overlooked how important the compilation time may take in big projects. In the scope of this paper, it takes a deep dive into the errors that occur during this process. It's important to address this problem because "slow compiles may cause the programmer to be distracted by other tasks or lose context and reduce the number of changes that can be attempted during the day" (Seo, 2014, p.1). The main contribution of this paper is it breaks down the characterization of the developer's build patterns within the workflow to provide more context and insights. It also quantifies and categorizes the build errors and their frequencies based on 26.6 million of builds withing Google. These contributions are important because it helps "establishing the importance of building in the overall development process, and in quantifying the cost of specific resolutions times" (Seo, 2014, p.9). It also helps highlighting the need for techniques that can analyze the code build file to reduce build errors. The evaluation results of this paper are that "10 percent of the error types account for 90% of the build failures. Furthermore, dependency errors are the most common" (Seo, 2014, p.9)

Positive points:

Points #1: Relevance of data

The purpose of the research paper is to analyze empirically the errors produced during compilation and find characteristics and patterns within. It's essential that the data genuinely represents most developers' behaviors. Consequently, the authors excluded instances of, for example, very high number of builds done by *Janitors*, *Infrastructure developers*, *Builders* and *Robots*, as they don't do "normal" development. Likewise, builds from *inactive developers*, such as project managers or release engineers, were excluded due to their impertinence. Within the relevant data, the authors chose to further analyze the compilations errors from projects done in Java and C++, as they are the most utilized languages within Google. Therefore, the authors managed to filter down until arriving at 2 very relevant and convincing samples for this study.

Points #2: Accuracy of results

The empirical analysis done in this research is based on counting build errors. For greater precision, the authors applied a consistent methodology to data from both C++ and Java languages. The methodology consists of first count the build errors, then classified them into five categories: *Dependency*, *Type mismatch*, *Syntax*, *Semantic*, and *Other*. The graphical representations revealed that most build errors stemmed from specific types, notably "*undeclared_var_use*" and "*no_member*". The former refers to utilizing a non-existent variable, while the latter implies referencing a non-existent attribute in a class. These findings indicate that a limited number of error types are responsible for most build errors and is consistent across two languages. Having a comparison further bolsters the accuracy of the result.

Points #3: Comparison with other studies

The authors compare their results with other studies, therefore confirming or challenges their results. One differing outcome from another study showed that programmers typically recompile immediately after a compilation failure, while this research suggests there are notable delays between compiles. They provide a likely explanation for this difference and is that the other study might only considers the resolution time for single-error compiles. Conversely, an example of a study confirming the author's result is that "48% of compiles failed; *cannot resolve identifier* was the most common error, followed by *Type mismatch*." (Seo, 2014, p.10). By doing comparisons with other research means the authors are confident in their results. Additionally, it paves the way for more in-depth analysis.

Negative points:

Points #1: Study conduct within a single company

Despite Google having engineers with extensive expertise, the data they produce might not be representative of other companies' developers. As a result, while the findings might be relevant for Google, they may not be generalizable. However, this study can set as context and guidelines for future studies.

Points #2: Data acquired only through centralized build system

When a developer makes new modifications of existing code, they often compile it locally using tools like IntelliJ or Eclipse. However, before merging it into the main codebase, it must pass through a centralized compiler. This study only captures errors detected by the centralized compiler, overlooking those from individual local compilers. It would be interesting to have the data from individual compilers too to further ascertain the results.

Points #3: Data may contain outlier values due to anormal workflow

The ideal dataset would focus solely on development builds, excluding those from tests, release setups, or experimenting with new languages. Even though efforts have been made to eliminate such outliers, like removing data from code janitors or compiler developers. However, some unwanted data might still be present. This could potentially distort the results if they are significant in quantity.

Recommendations:

My first recommendation would be to use some more familiar term throughout the paper. Sometimes I find it hard to understand certain formulation because there are very technical terms. It's also a good idea to provide more context before using technical terms.

My second recommendation would be using this methodology on data provided by other companies than Google to validate the results. Because using data from only 1 company does not completely ensure the universality of the results.

My final recommendation would be the authors to provide more practical recommendations on the result. The authors only clarify the importance of having quantitative tool but does not delve into specifics of such tools.

References

Seo, H., Sadowski, C., Elbaum, S., Aftandilian, E. and Bowdidge, R. 2014. Programmers' build errors: a case study (at Google). *In Proc. International Conference on Software Engineering*. ACM.