ENSE 375 GitHub Report:

GitHub Code Review and Code Quality Plugins

Carter Brezinski, 200391111

22 February 2021

Introduction:

In recent years, GitHub has become a central hub for many software engineers, small projects, and growing businesses. Its ease of use enables users to increase their workflow and quality of code. GitHub has added to its initial benefit of providing shared work environments through repositories by adding a marketplace full of different extensions and plugins, whose functionalities range from continuous integration to dependency management. These plugins can greatly improve a developer's productivity, as well as the quality of the code that they produce and push to their repositories. The two plugins that will be covered in this section of our report are Code Review and Code Quality plugins. A brief introduction to these plugins, how they work, and most importantly how they would benefit our group and our code will be explained in greater detail below.

Code Review Plugin: DeepSource

The first plugin that will be explored is the DeepSource Code Review plugin. This plugin is one of the more commonly used plugins for development with languages such as Java, JavaScript, and Go. DeepSource works primarily in the field of Code Review, but it can also provide benefits in the field of Code Quality.

Much of DeepSource's code review prowess can be attributed to its code analyzers. DeepSource analyzers work primarily to define errors or potential issues in a developer's repository and report them back for the developer to see. These analyzers cover a large list of languages, and they primarily focus on finding counterproductive or highly ineffective parts of a user's code. This can include code that can be considered high risk for bugs, and even code that can reduce overall performance or total runtime of a project. Additionally, DeepSource uses transformers, which convert and automatically format code through the use of various structural tools such that all code is processed through the same lens.

It is through the use of DeepSource's analyzers, transformers, AutoFix, and automatic review functionalities that developers can improve their workflow. These functionalities allow for users to directly view and change issues present in their files, thus reducing the overall workload for a reviewer. As well, these functionalities can also automatically suggest certain fixes for these detected issues, which can save on production time and improve code quality. In addition, the automatic fixes will instinctively create pull requests with suggested changes, which reduces the amount of refactoring that a developer must undertake; in turn, this minimizes the risk of new errors being produced..

Overall, this plugin would benefit us as a group due to its comprehensive analysis and revision control for projects. The ability to automatically find issues, communicate such issues to the developer, and suggest resolutions is incredibly beneficial for time, production, and overall quality of code. As was mentioned above, the fact that this plugin can provide reviews for the developer strictly through the use of analyzing the code is highly advantageous, as it can occur independent of other reviewers and developers.

Code Quality Plugin: Code Climate

Code Climate is the second plugin; it prioritizes its focus and benefits with respect to code quality, but does so through its continuous use of code reviews and progress reports. Code Climate provides users with the ability to receive reports based on the line-by-line tests done to the developer's code. As well, Code Climate integrates well with numerous additional programs and services, such as Flowdock, Slack, and Lighthouse. These three services may not be considered useful for projects with limited scope (such as ours), but in a larger work environment, compatibility is a crucial and invaluable aspect for an external plugin. Code Climate allows for users to directly configure the frequency at which code reviews are generated, and the level of detail at which code is reviewed; the plugin will subsequently inform the user of any areas in the code that it deems "problematic." Additionally, as it is mentioned on their GitHub page, Code Climate also supports the ability for teams to run the exact same analysis locally, which reduces the requirement for team feedback during production.

Code Climate contains various methods of analysis for producing results that are filtered back to the developer. The first is the Churn method, in which the service includes the number of times that the specific file has been changed or updated in the previous 90 days. This method is

intended to focus on the maintainability of the file in question, as it records the frequency at which the file is changed or "churned." The next method is the Cognitive Complexity method, which executes a deeper analysis of one's code and deduces its readability and understandability. Code Climate can perform this analysis in real-time, and provide suggestions for highly readable alternatives if it detects overly complex code. Another method is Cyclomatic Complexity; in contrast to the Cognitive Complexity method, this method focuses on every possible outcome that a developer's code can experience throughout the entire testing and execution phases of the project. Additionally, Code Climate includes a duplication method to detect repeated or redundant sections of one's code. Finally, Code Climate uses a grading system for its maintainability method to grade the overall state of each file; the score is determined through evaluation of 10 different criteria, including number of methods, method complexity, and number of return statements.

We envision that the various features of Code Climate demonstrate high applicability and potential for our project. First, the Churn method will enable us to definitively observe each file's magnitude of maintenance, as it can record and display the amount of changes made throughout a semester. Secondly, the Cognitive Complexity method could enable us to develop a cohesive style of coding, and improve our code's readability. Thirdly, through the Cyclomatic Complexity method, Code Climate will allow us to observe the amount of paths and outcomes that our code could follow. Fourth, if there are portions of our code which could be simplified due to redundancy or unnecessary duplication, Code Climate's duplication module will be extremely advantageous. Finally, through Code Climate's maintainability method, we can receive frequent feedback on the overall status of our code, which will improve cleanliness and limit refactoring.

Conclusion:

It should be mentioned that both code review and code quality plugins tend to overlap due to their similar benefits in a project or workplace environment. Each of the plugins mentioned above could be of great use to Group C in the development of the final project, as they can ensure that everyone's code meets a certain standard. In addition, these plugins will also provide better management and enable consistent monitoring of the code that is being produced. In a workplace environment, cohesive structural design and quality of code are important for

both maintainability of the final product, and understandability of the product by all parties involved. This is the primary reason why integrating DeepSource and Code Climate into our group's project environment are likely to be beneficial.