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 to come and utilize their system to increase their workflow and quality of code. GitHub has added to its initial benefit of its users being able to share work amongst various repositories by adding a marketplace full of different applications and plugins which can greatly improve a developers productivity and quality of the code they produce and push to their repositories. These plugins can range from anything like Continuous Integration to assisting in Managing Dependencies. 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.

How DeepSource does a large majority of its reviewing is through the use of its code analyzers. DeepSource analyzers work primarily to define errors or potential issues in a developer's repository and report it 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 what is called a transformer. These transformers convert and auto format the code through the use of various code formatters so that all code gets looked at in the same way.

It is through the use of DeepSource's analyzers and transformers that developers can improve their workflow due to the benefits of DeepSource's AutoFix and automatic review functionalities. These functionalities allow for users to directly view and change issues present in their files, thus reducing the overall workload for a reviewer. Additionally these functionalities can also automatically suggest certain fixes for these issues detected, which can save on production time and improve code quality. In addition to this, these automatic fixes will instinctively create pull requests with these suggested changes, which also reduces the work a developer has gone through changing everything and risking another error.

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, point out these issues to the developer, and recommend a quick fix for the code is incredibly beneficial for time, production, and overall quality of code. As it 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 accomplish the same goal without needing to rely on another reviewer or developer.

Code Quality Plugin: Code Climate

Code Climate is the second plugin, it prioritizes its focus and benefits to that of code quality, but it does so through its continuous use of code reviews and progress reports. It does this by providing users or teams the ability to receive reports based on the line-by-line tests done to the developers code. Code climate integrates well with other programs or services as well. Services like Flowdock, Slack, and Lighthouse may not be considered useful for this current project, but in a larger work environment having a plugin that can mesh well with other services and programs is very helpful. Code Climate allows for users to directly configure just how often a code report can be sent that can indicate the quality of your code, how much it was tested, and it will inform you of what is considered ‘poorly tested’ code. Additionally, as it is mentioned on their GitHub page, this plugin also supports the ability for teams to run the exact same analysis locally, which reduces the requirement for team feedback during production.

This plugin chooses to use various different analysis methods to produce their results that get filtered back to the developer. The first method is the Churn method, where the service includes the number of times the specific file has changed or updated in the last 90 days. This method is meant to look at the maintainability of the file being made versus how often the file is being changed or ‘churned’. The next method is the Cognitive Complexity method. This method can be considered more unique than the churn method, because it takes a deeper look at your code and deciphers just how difficult the code in the file is to understand. A benefit to Code

Climate is that as these changes are being made to the file, if it catches something that could be considered hard to read it will advise that you use a method that is considered easier to read. Another method is the Cyclomatic Complexity, different from the Cognitive Complexity method, this method focuses on all the different executions a developer's code can experience throughout the entire testing and execution phases of the project. Code Climate also uses a duplication method to discover which portions of your code are duplicated or highly similar to one another and could be simplified or are considered redundant. And finally Code climate uses a grading system for their Maintainability method to grade files based on their technical debt that is determined through 10 different checks ranging from the number of methods, to method complexity, to the number of return statements present.

Now based on all of these methods, the question is how could Code Climate benefit our group's project? Based on the churn method, when the development for our project begins we will actually be able to see what files are taking the most time to produce, receiving the most changes, and which files are considered too heavy maintenance. Based on the above method of Cognitive Complexity, this method could help all of us have a more cohesive coding style and ensure that our code is less sloppy and easier to read overall. Based on the Cyclomatic Complexity method, Code Climate will allow us as developers to see just how many paths our code that we're developing can potentially go through during our testing and execution phases. Based on the Duplication method, if there are portions of our code which could be simplified due to redundancy or unnecessary duplication, having a plugin that can detect and inform us of such will be very beneficial. And finally based on the Maintainability method, having a certain grading standard for our code that we as a group should aim for would hopefully produce a more cohesive final product.

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 their development of our final project to ensure that everyone's code meets a certain defined standard. In addition to this it will also allow for us as a group to have better management and for all of us to be able to monitor the code being produced. In a workplace environment, cohesive structural design and quality of code are important for maintainability of the products and for the product to remain understandable by all parties working on the project. This is why integrating these plugins into our group’s project environment could be a big benefit for our group project.