Code reviews are most effective on code that has already successfully compiled.

1. Introduction

The Code Review efficiency is depending on the different situation. To Peer Code Review, it is more effective on code that have already compiled. But to the Self-Code Reviews doing it before compiling is better. The Supervisor Code Reviews is always happened after the compiling, so in this paper it will just have a brief introduction.

Code Reviews is a useful technique using in Software Engineering Projects. The Code Reviews is one of the method for developers check source code for error systematically [1]. It helps developers, testers, project managers to find out defects, tracking defects and have a deeps understanding of the project. When people talking about Code Reviews they may means differently. There are three types of Code Reviews means by different people, Self-Code Reviews, Peer Code Reviews, and Supervisor Code Reviews. Self-Code Reviews is try to improve the quality of the project and improving the skills of the developers. Self-Code Reviews is one of the necessary phase in Personal Software Process (PSP). The Peer Code Reviews is to improving the project quality. Peer Code Reviews is a quality ensure method to software engineering project. The Supervisor Code Reviews is to get know about the source code in the software engineering project and it also helps the new coming developers to maintain the old software.

2. Code Reviews plays an important part in Personal Software Process (PSP)

Personal Software Process (PSP) is a process that raised by Humphrey to improve the software programming process to individual developer in software engineering project [2]. The purpose of Personal Software Process (PSP) is to help the software developer able to control their own software processes and know their ability to make the software project more productive and in good quality. Another aim of PSP is to make the software programmer have the ability to predict the process of their project including but not limited to time, effort, and quality.

PSP have several level, when a student start learning he will start with PSP0 and after he finished the course his PSP level will become PSP2.1 [3].

- PSP0 and PSP0.1 have 3 phases, in order they are Planning, Coding, Compiling, Test and Post Mortem.
- PSP1 and PSP1.1 have all the phases which PSP0 and PSP0.1 have and beyond that add a Design phase after Planning phases. Besides, developer need to estimating and record the time and size of their project.
- PSP2 and PSP2.1 based on PSP1 and PSP1.1 and add Design Review phase after Design Phase and Code Review phase after Coding phase. Software engineers need to learn how to evaluate and improving the process according to the recorded defect logs.

Plan and Tracking phase just like Design, Design Review and Code Reviews are not using in just one project, but through the projects that the developer have programed. By using the history data just like the time, size and defects log, the developer not only can reduce their mistakes make in the future but also can estimate their performance for future project more accurate.

As states before, one of the PSP aims is to improve and secure the quality of the software project, phase Design Review and Code Review is the two method to make sure the quality [3]. The quality of the project is mainly measured by the terms of the defects. The lesser defects that the project have means that the project in PSP is more successful. To find out where the defects are injected and record the type and solution for each kind of defects helps developer to reduce their defects amount and prevent the same kind of defects that happened in the same situation in the future. To meet this requirement, the PSP recommend using Design Reviews and Code Reviews. To make sure the Design Review and Code Review really works, PSP recommend using Design Review Checklist and Code Review Checklist to help developer to review their design and code in a systematically order.

In PSP2.1 the Code Reviews phase is added after Coding phase. But some of companies and software engineers put this phase after compiling. We will talk about the differences between these two methods.

In PSP the Code Reviews means Self-Code Reviews.

3. Code Reviews have different types and find different types of defects

Code Reviews is one of the methods to secure the quality of a software project. By function, Code Reviews can be divided into three types: Self-Code Reviews, Peer Code Reviews and Supervisor Code Reviews.

Self-Code Reviews is done by developer himself to find out defects, failures and errors in their source code. Compare to the Peer Code Review and Supervisor Code Reviews, Self-Code Reviews such as Code Reviews in PSP have an extra step to recording the defects, size, time and estimated error to evaluate self-performance in the current project and also use for make the estimate more accurate in the future.

Peer Code Reviews is done between developers to find out defects or errors in the code.
In “Automated Defect Prevention: Best Practices in Software Management.” written by Kolawa, Adam; Huizinga, Dorota, Peer Code review techniques have two types [4].

- Formal code review: formal code reviews is a code review that including very careful checking process with a lot of details phases. For example, Fagan

Inspection. This kind of code review will check the code text line by line and be proven to be effective [4].

- Lightweight code review: this is the kind of review that reviews by other developers that don't according to so many process but still have a good efficiency. Such as Over the Shoulder, Email pass-around, Pair programming and Tool Assisted are all Lightweight code reviews techniques.

Code Reviews can catch almost 60% of the product defect according to Boehm and Basili's research [5]. Besides the defects that catch by code review have several types and conclude in two big types: functional defects and evaluability defects.
Defects that cause failure: this kind of defects will cause the system stop running or crash.
Defects that cause error: this is a kind of defects that cause the software give out a wrong result by a not correct function or data definition.
Defects that cause deviation from quality: this is the defects that every time a developer change the program.
The first two types can also be combine to be Functional Defect and the third can be called Evaluability Effects. According to Siy and Votta's research, 85 percentage of the defects that are the Evaluability Effects [5]. This means that the most of the code review cannot find out the any execution problems including failure and error. And what make up this 85 percentage is mostly like meaningless variable definition. And the left 15 percentages mostly are wrong spelling mistakes.

The Supervisor Code Reviews mostly including the Peer Code Reviews process and mainly focus on evaluate the project and know about the source code for further development and maintain. Supervisor Code Reviews is one of the methods to evaluate project after it is successfully done before. When a new developer come to an existing project, a Supervisor Code Reviews can help the new comer to get into the project quickly and the space to improving the project quality in the future at the same time.

4. Process of Compiling and the purpose of using compiler

Compile is the process using a compiler to translate a programming language to machine language for computer to execute [6]. After coding, if developers want to test the program, they need to compile the program and execute it. If there is a failure that cause compile is not successful, means there is a grammar failure or other failure.

The failure of compile is a sign for developer to notice that there are functional defects in the program. If the programmer did code review before, he may found out these defects.

To compile a program mainly need to go through three parts in a compiler: [7]

- Frontend: Find syntax and semantics and label them, then using a temporary

source code for the next phase to do the next step. In this phase, frontend will check the perform types and collet perform type information. The frontend will also give errors and warnings by using a lot of analysis including but not limited to lexical, syntax and semantic analysis.

- The middle end: In this phase, middle end mainly is used to reduce the unnecessary code and improving the efficiency of the source code. In this phase the unused code or unreached source code will be removed, the loop will be reduced if possible. After doing this, middle end will generate a new temporary source code for the backend to do the next step.
- The back end: This phase is mainly transform the source code into the code that the computer can execute. It first makes the assembly code and the register location for computer to run in the process. It also deals with the multi-thread problem that run in the process to make sure the different thread won't crash into each other and work efficiently. Then it will transfer the source code into language that the computer can execute.

As the process showing above we can see that in the compiling phase the compiler is not just perform like a translator for human to the computer but also a failure and algorithm checker and fixer [6]. It helps developers to automatically find and remove a lot of lexical, syntax and semantic mistakes and unnecessary source code. This is another thing need to take into consideration when it comes to compare with Code Reviews.

What is more, as we using the compiler to compile our program, we can see that if the compiling is not going right, the compiler will give errors and warning message to tell you where the failure happened [7]. If it is a normal type of failure just like out of boundary, the errors and warning message will give a message that alert when does the failure happened and which line of source code that this failure happened. This will help the developers reducing a lot of time to find out where the errors are and how to fix it.

5. The different purpose of different types of Code Reviews?

The main purpose of code review as states before is to secure and improve the software quality [4]. The way code reviews can make sure quality is to find out defects and fix defects. But in different kind of code reviews have a slightly difference purpose.

Self-Code Reviews using checklist to review the code write by the developer himself to find defects, error and failure to improve the quality of the software and the skills to the developer [3]. By using the checklist, the developer can find out the defects including the types of the defects, the inject phase of the defects, the remove time of the defects. By recording these kinds of information and correct the defects by themselves, the program quality will be improved and their skills in programming will improved. What is more, by using this kind of method, the

developer can estimate their effort, time using in different project in the future to make the project in a good schedule and quality.

Peer-Code Reviews is another technique to improving the quality of software engineering project [4]. By reviewing each other's source code, they can find out the defects that the developers can't found by themselves. In the same time, some mistakes they haven't make before will comes to them to raise awareness for them. Peer-Code Reviews mainly purpose is to find out defects in the source code by others to cove the blind side of the developers themselves to improve the quality of the program.

Supervisor Code Reviews mainly to over check the source code again and have a better understanding for the source code such as introducing the project to a new coming college.

6. **Differences between not compile and compiled.**

Go through the process that the compiler compiling the source code we can find that the compiling process not only just doing as a translator between human and machine. It also acts as a checker for the failures that happened a lot and a professor to improve the efficiency of running the code.

Doing Code Reviews before the compiling phase will find out not only the failures that may stop compiling but also the errors that happened after the compiling is already done.

In the opposite, doing Code Reviews after the compiling can find the failure that the compiler cannot found such as incorrect format of the input. It also will find out the errors that happened after the compiling.
No matter before or after, the final result of Code Reviews is all find the failure and errors. But the differences are the effort, time and benefit that spend on the Code Reviews.

7. **Efficiency are different in different types of Code Reviews**

Which is better, before or after the compiling is mainly depending on the purpose of the Code Reviews. Because the different purpose of the Code Reviews, the answer will be different.

To Self-Code Review, doing Code Review before the compiling is much better that before the compiling. Beside to improving the quality of the project, Self-Code Review have another very important purpose which is to improving the skills of themselves. Comping by the compilers will make them more easily to find out the failure of the source code, but at the same time their experience to find out and avoid to make the same failure is reduced. This is the Self-Code Reviews doesn't want to see. Using Code Reviews after the compiling will not help the developer to find out

the reason why they make the mistake and will make them do the same mistake in the future.

But to the Peer Code Review, it will be much different. The main purpose of Peer Code Reviews is to find out the failures and errors and make the software run without problems. In Peer Code Reviews, developers is not expected some common problems that happens over and over again. The simple and common question will cost a lot of unnecessary effort and time. If the Code Review is done after the compiling, most of the common failure of the program will be figure out and fixed. This will increase the efficiency of the Code Reviews process and help the developer to focus more on the errors and not the failure. By doing this, the productivity of the project teams will be better than doing this before the compiling.

8. Conclusion

Due to the different purpose of Code Reviews, the answer to the question will be different.

When this comes to Self-Code Reviews, doing Code Reviews before compiling is more efficient because the main purpose of Self-Code Reviews is to improve both the quality of the program and the skills of the developers themselves. To do Code Review before compiling helps developers to find out the weakness in their programming process and help them to avoid to doing the same thing in the future. But after the compiling, the developer will just ignore these problems that solved by compiler, and it is no help to improving their own skills.

But to the Peer Code Reviews, doing after the compiling is much better. If the Code Reviews is doing before the compiling, and during the code review, they find out some failure that stop the program from compiling. Then the developer need to go back to the source code and fix the failure. After doing this, the source code is changed and another Peer Code Reviews is needed. This will cost unnecessary effort and time to doing the things which the compiler can help to do. The Peer Code Reviews is mainly to make sure the program is in good quality and make it work properly, but not improving the skills.

So doing Self-Code Reviews before the compiling is more efficiency and doing Peer Code Reviews after compiled is more efficiency.

Word Count: 2699