A Study on Code Peer Review Process Monitoring using Statistical Process Control

Alhassan Muhammad Abubakar  
Faculty of Computing,  
Universiti Teknologi Malaysia,  
81310, Skudai, Johor  
Email: mohanajeeb66@yahoo.com

Dayang N. A. Jawawi  
Department of Software Engineering,  
Faculty of Computing,  
Universiti Teknologi Malaysia,  
81310, Skudai, Johor  
Email: dayang@utm.my

Abstract—Software development process (SDP) and Software products are like two sides of a coin. We cannot achieve one without another. Today, in our software industries, monitoring software process is very challenging. Many problems of software process monitoring are hampering the quality of our software products. In this paper, we plan to address the problem of process instability and causes of process anomalies. These problems can be addressed using one of the powerful statistical techniques known as statistical process control (SPC). Also, control chart would be used in our study as it has been proved to be one of the suitable tools of SPC in monitoring process stability. As we know, the more defects we found during SDP, the less quality of the software product. Therefore, this study considers defect density as the metric to be use due to its significance in determining product quality.

Key words—Software development process, Statistical process control, Control Charts, Defect density.

I. INTRODUCTION

Software development process has been defined by many authors within software engineering domain. These definitions are the same in respective of the type software product, architecture of the products as well as the tools used in developing the product. Basically, software development process involves the arrangement of people (mostly software engineers) with related facilities or tools for development of good quality software product. Due to the high intensive or involvement of people in SDP, it is possible to have so many factors that can contribute to the process instability. For example, lack of proper arrangement of facilities can have a great impact on the process behavior.

According to the authors in [2], causes variations are inevitable in any process. These variations may be either as a result of common cause’s variations or special (assignable) cause’s variations. The common causes may be as a result of common interactions between human and material resources that are involved in the process. In other words, the normal variation of process that exists because of normal interactions among different components (people, machines, methods and so ‘on) of the process is referred to as common cause variation and sometimes is also called controlled variations. Assignable causes of variations on the other hand, may be as a result of events that are not included in the process. This event changes the process behavior, have significant impact on the process performance and the quality of the product. Also, this type of variation is referred to as uncontrolled variations [9]. Since common causes of variations are also referred to as controlled variations, then identifying and eliminating uncontrolled variations (assignable causes of variations) is one of the successful paths towards achieving process control and stability. The technique that we can use to achieve control and stable process is called statistical process control.

The idea of SPC was initiated by Shewart in 1920’s [2]. During that time, he was working or conducting research in Bell Telephone Laboratories on how to improve process quality and lower costs. This leads to the founding of SPC. Based on his studies, SPC can be used to monitor and control process using various statistical tools. Among these tools are: pareto charts, histogram, control charts and so ‘on. This study focus on control charts as it has proved to be effective statistical tool for detecting out-of-control situation. That is to say, many researchers in the field of software engineering performed several studies with this statistical process control (SPC) technique and proved to be effective for process control and improvements. In line with this, this study would be additional effort by contributing with the statistical idea of ensuring software process stability.

However, the remainder of this paper is organised as follows; section 2 contains the motivation of our study. Section 3 presents related work. In section 4; we describe the problem statement of this study. Section 5 describes our research methodology. Section 6 presents our initial result. In section 7; we discussed about the problem we encountered in this study and section 8 contains summary and future work.

II. MOTIVATIONS

Today, the world is changing to a digital world. Many activities of our daily life are carried out with support of information technology or software systems. In our global
world, various sectors contribute their quota towards improving our economy and other sectors. Therefore, software engineering sector should be considered in terms of economy improvements, educational improvements and so on.

Basically, appreciating the status and contributions of software systems to support human activities is one of the motivations of this study. To support and improve the use of software systems in all our activities, we can contribute significantly by monitoring SDP. This will enable software engineers to investigate process behavior over time and quickly react to any detected process anomalies so as to enhance the quality of their products. However, many researchers share their studies on using SPC in the field of software engineering community.

III. RELATED WORK

Software organizations use standard models of process improvement purposely to improve their software development process, and to be sure that certain level of quality has been achieved during the development of the software [1], [6], [11]. This will enhance the product quality and bridge the gap between customer and software engineers. Example of these models is capability maturity model (CMM) which was later superseded by capability maturity model integration (CMMI). This study focused on CMM because, CMM is more specific on process assessment and it implicitly directs software organizations (at CMM level/stage 4) to use quantitative techniques such as statistical process control to quantitatively managed their software development process.

In line with this, many researchers share their idea or experiences on using SPC for software process improvement. Recently, in the studies of [6], control chart of SPC were used to improved process performance of software development life cycle. During their study, they focused on requirement process and they visualized how statistical process control can be effectively and efficiently applied to the processes in order to achieve good quality software products. Additional effort was done by researchers in [11], [2], [16], they emphasized that in order to increase software process efficiency and effectiveness so as to have good quality product; we need to enhance our process measurements skills. Good measurements are suitable for SPC implementations and enable us to evaluate process performance.

Moreover, in the work of [22], they use SPC to monitor software reliability. But, monitoring software reliability is not a very simple activity. These authors make a good effort in using SPC on inspection and review process so as to improve software reliability and quality. The authors used control chart which is one of the important tools of SPC to monitor the time between failures, when the time between failures fall outside control limits specified by these authors, then causes of variations may be presents and need to be addressed so as to ensure software reliability. Similarly, in the study of [1], SPC can be use for continuous software process improvement. He proposes an SPC-based approach that implements SPC technique, and used it for software process improvements. Also, the author in [19]; discovered his experience of using SPC in software development process. He conducted a case study on Hitachi Software Engineering (HSE) in Japan. After he conducted several use of control charts to investigate HSE processes e.g peer review process, he discovered many process anomalies through the use of control charts. He concluded that SPC is a very useful quantitative tool that can be use in software development process.

Similar work was done by researchers in [15], [18], [16]; they used control charts to monitor software quality evolutions for defects. They used the chart to find out the quality evolution of Eclipse and Gnome systems. They measure software quality by of these two open source software systems by number of defects and use control charts on changes in defect over time. They clearly found process deviations of these two systems through the use of control charts and concluded that it very challenging to keep software quality under control during evolutions. But on the other hand, they believed that when control charts are effectively used, software organizations can monitor quality evolutions over a long period of time.

Furthermore, recently some researchers [8],[22], compare Modified Maximum Likelihood Estimation (MMLE) and half logistic Distribution estimates using SPC in order to find the control limit for assessing software reliability. However, because software reliability measure is a one of the important of software quality attributes, these authors concluded that either exponential distribution or HLD is preferable for the estimations of data that is in the form of inter failure times and SPC is a very powerful quantitative technique that can be used to control the estimates in order to ensure software reliability.

Furthermore, another contribution was made by authors in [20] in which they used SPC to automatically detect performance regression. Performance regression is very important in software development process but often consume time. These authors proposed an approach (control charts) to analyze performance counters across test runs using statistical process control technique of control charts. The result they obtained at the end clearly verifies that the proposed approach can be used to identify performance regression in software systems. Similarly, additional work has been done another researcher [9]; He presents practical application of statistical process control. He found that SPC helped him understand and predict product quality and the development process controlling the quality. He also added that the technique (SPC) provided him with the hard data he needed to justify his result at the end.

Moreover, researchers in [3]; added that statistical and process thinking lead to the use of SPC in order to determine stability of processes used to develop Software. However, because software process improvement is a continuous activity, controlling and monitoring software process is still an issue within software engineering domain.
IV. PROBLEM STATEMENT

Based on the studies we have seen so far, most of them addressed issues of process monitoring using different control charts of SPC. But on the other hand, some of the studies focused on the interpretations of SPC instead of proper monitoring of the software process. Monitoring software process stability remain an open issue that when properly handle, we can enhance the quality of the software product. However, there are some issues such as; process performance deviation that affect software process, by carefully monitoring these issues; we can improve the stability and capability of the process. Therefore, in this section, we describe some problems which we plan to address at the end of our study. These are as follows:

1. Problem of detecting software process deviations as a result of variations.
2. Problem of investigating the causes of variations in software process.

In order to address the above mentioned problems, some silent questions should be considered.

a. Are CMM lower maturity level software industries matured enough to use or apply SPC for process monitoring and control?

b. How can we achieve successful SPC implementation at the CMM lower maturity level organizations?

To responds to these questions effectively, our studies will go further to conduct a case studies at CMM lower maturity level industry. Also, as we seen from the literature in the previous section, SPC was used by many of the researchers and proved to be effective statistical method of process control. Therefore, it can help us to address the above mentioned problems and evaluate whether the process is under control or not by using control charts. When the process is affected with either special or assignable causes of variations, control charts will play a vital role of identifying or detecting these causes through the use control limits. On the other hand, when these causes exceed control limits, the process is said to be unstable and the causes must be identify and eliminated.

V. RESEARCH METHODOLOGY

Statistical process control is a very strong quantitative technique for managing and improving process stability. As we discussed earlier, our studies focused on control charts of SPC as the best tool to be used to carry out our analysis. This is because of its significance in detecting any process anomalies by calculating control limits and plotting the data on a chart to visualized the result. However, Figure 1 represents a simple diagram of control charts.

Control charts are used in software engineering in either variable data or attribute data. Xbar chart, R-chart, S-chart and XmR charts are all examples of variable data control charts. P-charts, u-charts, c-charts and XmR-charts on the other hand, are all examples of attribute data control charts. This study focused on u-chart because of its significance on our selected metric (defect density). A defect is a flaw in a component or system that can cause the component or system to fail to perform its required function [23]. For example, an incorrect statement or data definition is a defect. This clearly implies that, defects can be found in any phase of the software development life cycle. Therefore, defect density is the ratio of number of defects discovered in a component or system and the size (functional size) of the component or system as expressed below:

\[
\text{Defect Density} = \frac{\text{Number of Defects}}{\text{Functional Size}}
\]

In view of the above therefore, the functional size can be source lines of code (SLOC) or lines of code (LOC). However, defect density is usually used in order to measure the quality of the work product. In other words, when the defect density is low during the phases of development, the quality of the product will be high where as when the defect density is very high, it implies that the software product quality will be very low. In line with this, defect density will provide us with the idea or answer of whether the customer will be satisfied with the product or not, how much rework is required and so on. As we stated earlier, attribute control charts such as u-chart can be used to monitor and control defect density in a process.

The u-chart attribute control chart shows how the frequency of defect changes occurred over a period of time. The procedure for calculating control limits using u-chart is as follows:

\[
\begin{align*}
UCL &= \bar{u} + 3\sqrt{\frac{1}{n} \sum ai} \\
CL &= \bar{u} \\
LCL &= \bar{u} - 3\sqrt{\frac{1}{n} \sum ai}
\end{align*}
\]

Where by \( \bar{u} \) is the average rate over all areas of opportunity \( (ai) \) and is obtained by using the following formula:
Furthermore, the area of opportunity should be calculated for each sample and can be found by using the following formula:

$$A = \sum_{i=1}^{n} f_i$$

Whereby, $f_i$ represents the functional size or (SLOC) for each sample. Similarly, the lower control limit (LCL) as well as the upper control limit (UCL) can be calculated using the following formulas:

$$LCL = \bar{x} - 3 \cdot \frac{s}{\sqrt{n}}$$

$$UCL = \bar{x} + 3 \cdot \frac{s}{\sqrt{n}}$$

Moreover, the presence of $a_i$ in the both lower and upper control limits indicates that each limit should be calculated or computed for each sample. By using these control limits, the u-chart is very sensible in detecting any deviation in the process based on the defective measures of the process. Despite the differences of control charts on our data, Figure 2 illustrates the step-by-step procedure of implementing control charts for process monitoring and control.

Based on the above Figure 2, to achieve real successful implementation of this approach in process investigations, three major steps should be considered. These are: collecting data, analyzing data and acting on results. Our study move forward by collecting data from CMM lower maturity level software industry (i.e CMM level 2). Due to the high confidentiality of the data, the name of this organization and the data cannot be stated in our study. In other words, our study will only reveal the result we obtained after collecting and analyzing the data.

VI. INITIAL RESULT

As seen from the literature, the use of control chart within the software domain is gaining more and more attention. The technique is usually used to quantitatively manage or monitor software process especially in CMM level 4 software industries so as to achieve process aims and objectives. In this section, we are going to justify the applicability of SPC for process monitoring and control at CMM lower maturity level software industries. In other words, the result we obtained by implementing or using control chart of SPC in CMM level 2 (repeatable) software industry for process control and improvement especially from the code peer review process point of view. We selected this process because of its significance in SDP. Without this process, many of our software products will lack certain quality attributes that will
leads to customer’s dissatisfaction and inability to achieve business goals.

However, the data we collected consists of nine samples or modules that were completed in 2008 by the software organization. These samples have different features regardless of their types, architecture, size, quality attributes and so on. However, in this data we are able to identify the functional size as well as the defects that are associated or found in each sample. Similarly, we calculated the centre line, lower control limit and upper control limit for each sample using u-chart attribute control chart. The result of this data was plotted using the chart as is shown in Figure 3.

As seen from the figure, based on the principles of statistical quality control, sample number 1 fall outside control limits (i.e upper control limits) and therefore the process is not stable or out of control. This clearly implies that the root causes (assignable causes) should be investigated for proper process improvement. In other words, we can say that it is clear the process was not under control when it was started and this indicates that the product will lack certain quality attributes. Moreover, the purpose for any out of control situation should be investigated so as to have stable process that would yield positive result. When we can identify the causes and quickly react to them, therefore the process will definitely change.

That is to say, detecting and eliminating process anomalies so as to bringing back the process under control would significantly enhance the quality of the software product. This is because; the quality of our software products depends largely on the stability of the SDP involved. Therefore, since we found the special cause of variation that makes the process out-of-control, we can simply remove it and examine the process behavior again. That is to say, we first remove project one and reinvestigate the process behavior by recalculating control limits and plotting new control chart. Figure 4 presents the u-chart of the process after anomaly was removed.

As we have seen in Figure 3 and Figure 4, we can simply say that the process is now stable (under control) and is defect free to produce quality software product. The high defect rate of the process was removed and the process is now in statistical control. However, sometimes unstable process may occur as a result of so many factors such as: lack of insufficient coding knowledge, violations of various coding rules and so ‘on. In such cases, appropriate measures such as adequate staff training and rework will play a significant role in achieving process stability. Finally, based on what we discovered in our study, the data used in the first module is different with the remaining modules. It is strongly important to use control charts for process that have similar data or activities.

VII. PROBLEM ENCOUNTERED

Implementation of SPC in lower level CMM software industries is not a very straightforward activity. In this study, we encountered a problem of process measurement that requires effective SPC utilization in respect of these maturity levels. Improper measurement would yield undesirable results. In order to implement SPC, software engineers should be equipped with various tools or methods that will enable them to carry out effective and efficient software process measurement.

However, addressing this problem would contribute to a large extent on process improvements and product quality enhancements. In line with this, additional effort is required on proposing new methods for effective process measurement especially at the CMM lower maturity levels so as to achieve successful SPC implementations and preparing these industries for higher CMM maturity levels.

VIII. CONCLUSIONS AND FUTURE WORK

In this study, we have seen the idea of SPC on software process control. In responding to our stated questions, question 1 was addressed by detecting process anomaly and we achieved process stability using u-chart attribute data control chart. Question 2 on the other hand, was addressed by discovering the root causes of the process variation. This is as a result of mixing dissimilar process activities as a result of
improper process measurement. Finally, based on the case study we conducted in this research, we can conclude that lower CMM maturity level software industries can utilize SPC for effective process control and monitoring. Our future work is to propose an approach or strategy to achieve successful SPC implementation at the CMM lower maturity level organizations. This will answer our last question in this study.

ACKNOWLEDGMENT

Our sincere appreciations to the entire software engineering research group (SERG), Universiti Technologi Malaysia for their humble support.

REFERENCES