Dynamic Guidelines to Statistically Control Peer Review Costs

Constantine M. Koursaris, Ph.D.
Department of Management Sciences
College of Business Worldwide
Embry-Riddle Aeronautical University
Daytona Beach, FL

Richard L. W. Welch, Ph.D.
Chief Statistician & Senior Technical Fellow
Northrop Grumman Corporation
Melbourne, FL

Abstract

During the system/software development lifecycle process (SDLC), peer reviews are conducted throughout to ensure that true-positive defects are detected and flagged for re-work. Often, false-positive defects may be flagged for re-work as well, thus increasing operational costs. This paper examines two possible organizational baseline and operational scenarios sequence recommendations using statistical process control of peer review costs. The optimal organizational operational scenario will be selected based on the analysis of over four years of empirical data and 2517 data points collected during the time of the research. The conclusions will show that the log-transformed data scenario is the best, most optimal, operational scenario to implement within the organization. Although the research was sponsored by a Department of Defense prime contractor to the U.S. Air Force bound software systems, it can be applied in the civilian sector of aviation transportation software systems in the management of the supply chain. Identifying accurate defects and implementing this scientifically proven method of controlling costs within the organization is of tremendous value in both government and civilian aviation transportation systems.

Keywords
Industrial engineering, operations research, statistical process control

1. Introduction

Identifying defects and controlling costs during peer reviews of a product being developed for the U.S. military ensures that quality is not compromised while the product is going through its development life cycle. Several process improvement guidelines have been established, but none specify what type of method to use in order to more accurately product detect defects and thus control costs within the organization developing the product and eliminating unnecessary re-work for the developers. The Capability Maturity Model Integration (CMMI) [1] has provided organizations with process improvement approaches. The application of high maturity statistical models and techniques within a CMMI Level 5 framework has been effectively employed and illustrated as a result of a continuing analysis and exploitation of peer review data. Statistically controlling peer reviews of requirements, software designs, code, and test artifacts, within all phases of the Software/System Development Life Cycle (SDLC) process has proven to have a direct relationship with controlling costs. Reducing and controlling variation directly affects the reduction of peer review cost and improves efficiency. Controlling the total cost of software development is one of the fundamental interests expressed by many organizations. Also, the necessity to improve internal processes within organizations, have given us structured methodologies to use and implement in order to become more efficient, increase productivity, and reduce costs. Using statistical process control (SPC) to our advantage, offers many benefits. Walter Shewhart developed control charts to indicate when special-cause variation exists in a process [2]. One notable benefit of using SPC is that it reduces programmatic risk. It gives superior insight into average performance, and variability of the controlled process. It provides higher confidence estimates. Use of SPC enhances predictability and stability in executing the job [3]. It enables proactive process improvement to meet management or customer performance targets. As a result, the common cause variation is removed from the process [4]. The focus of this paper is to recommend the best organizational baseline and operational scenario for statistically controlling peer review cost using dynamic guidelines. By using statistical process control methods, we can control costs, by first identifying defects. If the
distribution of the process is not normally distributed we will face Chebyshev’s inequality problem which is explained more thoroughly in the Problem Statement section.

2. Problem Statement

The challenge lies in the desire to introduce successful SPC techniques for optimal quantitative project management in the software development lifecycle phases [5]. What drives the need for statistical control of peer review cost? Peer reviews are ubiquitous throughout the SDLC and represent about 10 percent of the total software development effort in a mature organization. Since the peer review is typically the last process step before an artifact is placed into the configuration management system, it offers a significant control opportunity. If peer review sub-processes are not in control and are not capable, it will not be possible to meet baseline budget and schedule allocations. This simply translates to exceeding the amount allotted for the project, and more resources have to be spent to re-work the problems/defects found. The cost of re-work is exponential from the early stages to the completed stages. Software and information technology (IT) have been valued by industry experts at about U.S. $1 trillion in 2006. Data indicate the cost of poor quality (COPQ) exceeds $250 billion and could be as high as $500 billion which means that 25 to 50 percent of every company’s software and IT budget is at risk [9]. Another way to view this problem is that 80 percent of the failures arise from 20 percent of the total items. As Juran coined the term “vital few and useful many” it becomes easier to make improvements in the vital few [6]. Experts such as W. Edwards Deming believe that Management is responsible for 94 percent of all troubles that belong to the system and 6 percent are related to some special event [7]. It is important to realize that once software development has begun, product development is half over. Opportunities to recognize and correct special and common cause variation in the design process are gone. Department of Defense studies indicate that first year decisions determine up to 70 percent of total life cycle cost. Early, proactive, and effective statistical control offers great practical benefit [8]. Control charts are very powerful tools for achieving SPC. By implementing successful SPC techniques not only are we able to improve process consistency and stability, but also create a predictable process that results in a long-term quality improvement for the future.

In studying the use of effective techniques in analyzing data using statistical process control, subject matter experts such as Donald Wheeler [9, 10, 11, 12], and Douglas Montgomery [13, 14, 15] have illustrated conflicting views. The preferred approach and methodology of Wheeler suggests that data be analyzed in their raw state. Contrary to Wheeler’s stance, Montgomery suggests the use of log-transformed data prior to analysis. The big question arises of who is on the right track with this argument. Due diligence motions for the exploration of both methods advocated by Wheeler and Montgomery to compare raw data versus log-transformed data in order to determine which method is actually the best case and operational scenario for organizations to implement. We know that process operational costs can be controlled successfully by implementing CMMI methodologies in organizations that produce products and services. This in turn, promotes stability and consistency in process execution. Process, people, and technology are the determining factors of product, cost, schedule, and quality. Therefore it is essential to first understand the process before we can evaluate the cost factors within an organization.

An effective process, form an organization’s experience practices, has the characteristics of structured collections of practices that fit the process model. As we begin to understand the process model within an organization, we see that it can be used to help set process improvement objectives and priorities, and helps ensure stable, capable, and mature processes. The process model can also be used as a guide for improvement of project and organizational processes. With the aid of an appraisal method, the process model can also be used to diagnose the current state of an organization’s practices.

A stable process operates within the control limits 99.73 percent of the time. A stable process meets budget requirements and offers opportunities for systematic process improvement [16]. 99.73 percent represents the proportion of values that are contained within three standard deviations from the mean for a data set that is normally distributed [17]. However, in reality, data tend to behave in a non-normal manner and appear skewed instead. If the data is not normally distributed then Chebyshev’s Inequality [18] must take precedence. Chebyshev’s Inequality is a serious reliability threat to the proportion of values that are not contained in the distribution. Within three standard deviations we can only account for 88.9 percent of the data, leaving approximately 11 percent of the data that are not accounted for, to be misinterpreted as false positives. See equation (1) for application of Chebyshev’s equation with three standard deviations from the mean (k=3).

\[
P_V = 1 - \frac{1}{k^2} = 1 - \frac{1}{3^2} = \frac{8}{9} \approx 0.89
\]
The use of control charts or process behavior charts show whether we are operating in an “Ideal State” or not. The four requirements for operating in this ideal state are:

1. The process must be inherently stable over time.
2. The manufacturer must operate the process in a stable and consistent manner. The operating conditions cannot be selected or changed arbitrarily.
3. The process aim must be set and maintained at the proper level.
4. The natural process limits must fall within the specification limits.

Only with the use of control charts can these four requirements be met. At the same time, we have to consider the dynamic changes of the data, the frequent process changes. The primary goal is to establish and maintain process stability. Once the process is stabilized by operating in a stable and consistent manner then the process aim has to be set correctly [19]. The organizational strategy for conducting peer reviews (PRs) lies between two control loops. Within the first control loop, the strategy is to control costs by stabilizing the variance first, then reducing it. Stabilizing variance first establishes a level 4 state of the development process capability maturity model integration (CMMI). Reducing the variance secondly, establishes a level 5 CMMI state. After the variance is stabilized and improved, cost reductions follow. Efficiency can be increased as a consequence.

Most often, the cost reduction is functionally linked to the variance reduction. In other words, reducing the variance causes a cost reduction. For example, as we elaborate on the two different organizational baselines and operational scenarios, looking at the lognormal distribution we can see the functional dependency between the mean and variance, the first two moments. Also, from a planning point of view, determining who will participate in the design inspection review, selecting a trained moderator, determining the number of sessions needed to be conducted, and determining if overviews are needed, will have an effect on how well inspections are conducted. Similarly, from the Moderator perspective, being familiar with the process, ensuring that key participants are present, following a checklist, following up and completing tasks at hand, and ensuring procedures followed, are all causes that ultimately have an effect on how well inspections are conducted. By the same token, the preparation that takes place, in keeping a log of errors, preparing for inspectors reviews, having a list of major items for discussion at inspection, and the entire inspection package as a whole, plays a major role in the effectiveness of peer review conducts. Finally, the causes of the inspection meeting taking place, ensuring coverage, defect recording, determining defect origin, leading to the resolution of all major defects, also play a vital role in effective inspections.

3. Literature Review

An overview of the Software Engineering Institute’s Capability Maturity Model Integration emphasizes recommended best practices and solutions covering the product life cycle of development and maintenance from conception through delivery. The Department of Defense (DoD) endorsed the CMMI practices. Six Sigma practices are in place to strengthen management strategies and improve customer value and efficiency. Dr. W. Edwards Deming played a vital role in process improvement methodologies and he coined Total Quality Management (TQM). The IEEE hosts a repository of software engineering standards, as well as a vast number of white papers and publications on improving quality and efficiency within organizations.

Using function point metrics to measure software process improvements [20], Software Productivity Research, Inc., an Artemis company, one important topic tried to answer what it costs to improve software processes by examining data from more than 10,000 software projects dating from 1984. From about 600 companies and government agencies, some of the data examined utilized the Software Engineering Institute’s Capability Maturity Model Integration and some did not. The study concluded that software process improvement programs follow a fixed pattern of six stages in a specific order, preceding by a preliminary stage that actually “set the stage” for the other six stages. These sets of stages began with stage 0 that dealt with the “Software Process Assessment, Baseline, and Benchmark”. Stage 1 dealt with a “Focus on Management Technologies”. Stage 2 focused on “Software Processes and Methodologies”. Stage 3 dealt with a “Focus on New Tools and Approaches”. Stage 4 dealt with a “Focus on Infrastructure and Specialization”. Stage 5 dealt with a “Focus on Reusability”. Stage 6 dealt with a “Focus on Industry Leadership”. It is important to note here that stage 0 was named as such because it does not improve anything, and is outside the six improvement stages. Stage 0 is very important in the sense that defined the CMMI establishment of a baseline. The baseline was used to provide an organization with quantitative basis for productivity, schedules, costs, quality, and user satisfaction in order to judge future rates of improvement. The baseline data serves to indicate future progress.

The Department of Defense (DoD) [21] has established standards in detecting software defects and methods to implement to control costs and software defects. As a result, the CMMI was developed in cooperation with the Department of Defense, industry, the Software Engineering Institute, and sponsored by the National Defense Industrial Association (NDIA). CMMI has become the standard practice for integrated process improvement. The CMMI’s
Koursaris, Welch

Purpose is to provide for improvement and reduction in costs, implement and integrate internal process improvement efforts, and provide a common baseline and lexicon for process improvement.

Prior to the establishment of the Carnegie Melon SEI in 1984 by the U.S. Department of Defense [21], it was a common trend in the software industry to experience cost overruns, schedule delays, and production of poor quality software. Since the establishment of SEI [22], the misconception existed that by using state of the art development processes and advanced software tools, software applications were created faster and with fewer failure errors. One of the problems that surfaced was the lack of empirical data with regards to software process improvement, so therefore it was not possible to calculate the cost of improving software processes.

Motorola started the six sigma quality path. The problem was that Management did not support the six sigma methodology. When a Japanese firm took over a Motorola factory under new management and began producing television sets with 1/20th as many defects and lower costs by using Motorola’s six sigma methodology, under new management, the picture was clear. A common tool that is advocated by today’s standards in employing six sigma methods is to use SPC techniques in order to define, measure, analyze, improve, and control (DMAIC) the process. The process baselines are derived from a description of analyzing the prior processes [23].

Promoted by Dr. W. Edwards Deming, as part of the Total Quality Management (TQM) methodology, is the Deming cycle which is a methodology for improvement. The basis of the Deming cycle is to perform four stages known as Plan, Do, Check (or Study), Act (PDCA). Although not scientific in nature, the Deming cycle focuses on both short term and long-term continuous improvement and organizational learning and it certainly paved the way to quality control. Planning deals with studying and understanding the process. Do deals with a trial implementation of the plan with data collected and documented. Check deals with evaluating the results and addressing any further issues. Act deals with standardizing the improvements and implementing the plan as “current best practice” [24].

It will be noteworthy to also elaborate on the Institute of Electrical and Electronics Engineers (IEEE) [25] software engineering standards which include the most current IEEE software engineering standards. Documenting comparisons and contrasts between the DoD sponsored CMMI SEI and IEEE will add value to the direction that the technology is emerging.

4. Research Objectives and Approach

The objective of this research is to provide a recommendation of the best operational baseline and operational scenario using statistical control of peer review cost. To statistically control peer review costs throughout the Software Development Lifecycle (SDLC), we will reproduce the organization’s engineering review times operation that statistically manage and control their peer review costs. A traditional process of a software development life cycle contains multiple phases in which peer reviews are conducted. The focus will be on the peer review of code phase of the SDLC. Why are peer reviews so important in the software lifecycle? The answer lies in the control of peer reviews. The control of peer reviews offers a high leverage of capability for process control. Statistical properties and characteristics are expected from engineering peer review data. The log-cost model is used to overcome deficiencies in the distribution of cost data that make traditional control charting problematic, when used with software applications.

Another reason lies in the ubiquity of software defects. Many work products reviewed throughout software development lifecycle include design of artifacts, code, test plan, procedures, and reports. Another reason lies in the frequency. High data rates are therefore a consideration. The influence factor states that approximately 10 percent of the software development effort is spent on peer reviews and inspections. Code walkthroughs represent the biggest opportunity for detecting defects, at the early stages of software development, thus less costly to eliminate. If on the other hand, software defects are not caught because of one reason or another, the costs associated with rework are exponential as the software development lifecycle progresses. Peer reviews help to greatly improve the process. The objective is to first reduce variation, then control it. The control of peer reviews offers a high leverage capability for process control. The results are training people on the process, create procedures and checklists, and strengthen process audits. Improving the process also increases the effectiveness by increasing the mean. Training people and creating checklists as a result, reduce waste and re-work, and replicate best practices from other projects.

This research paper will begin with a study, within an organization’s method of using total cost of the peer review process of gains and losses of transformed data versus analyzing raw, untransformed data using historical data. The organization has used a logarithmic transform on peer review unit cost data (review hours per line of code) very successfully in the past, using total cost of the review. Included are preparations, conduct of peer reviews, and error corrections. The applicability of a log-cost model to control software code inspections has been proven by a senior author of the organization [26]. With the log-cost model peer review, we consider a code walkthrough in terms of the number of lines of code reviewed in a certain number of hours. The natural logarithm of the difference in cost between the current and the next peer review will be normally distributed with zero mean and a constant standard
deviation. The cost basis is the number of hours per line of code reviewed. The consequences of the log-cost model during peer reviews show that peer review costs are lognormally distributed. The natural logarithms of the peer review costs follow a normal distribution. Thus, the log-cost data meet the assumptions needed for successful control charting. The log normal equation is shown below with equivalent relationships written to obtain the expected value, variance, standard deviation, and the mean.

\[
f(x; \mu, \sigma) = \frac{e^{-\left(\ln x - \mu\right)^2}}{x\sigma\sqrt{2\pi}}
\]

\[
E(X) = e^{\mu + \sigma^2}
\]

\[
Var(X) = (e^{\sigma^2} - 1)e^{2\mu + \sigma^2}
\]

\[
\sigma^2 = \ln(1 + \frac{Var(X)}{(E(X))^2})
\]

\[
\mu = \ln(E(X)) - \frac{1}{2}\ln(1 + \frac{Var(X)}{(E(X))^2})
\]

We will continue the research and will analyze the problem as if we were trying to control the cost per defect found. We will use preparation and conduct of peer reviews only since error corrections are not part of this analysis. This research has not been performed previously. It will serve as new knowledge gained within this scope of software development and conduct of peer reviews. We will investigate statistical process control techniques used to run the control charts. We will determine the best type(s) of control charts to use, in order to control costs. Any software development company will benefit from this research.

The purpose of peer reviews is to detect code defects during the early stages of software development, so that they can be corrected before turning the software over to user acceptance testing. We know from engineering principles that the costs associated with rework, as the lifecycle progresses, are exponential. In this part of my research during the verification process area, it is essential to detect the defects during the peer review examination, in order to keep costs at a minimal. The software requirements specification (SRS) contains a detailed description of the software system to be developed. From a functional requirements point of view, a set of use cases are included to clarify the roles and interactions of the users with the software system. From a non-functional perspective, requirements pose certain constraints on the design or the implementation of the software system application. These design requirements include performance metrics, quality standards, quality of service attributes, and other design constraints such as specifying the way the system will operate. Software requirements include functionality, usability, reliability, performance, and supportability.

5. Experimental Data Sets
For the purpose of this research, we will analyze 2,517 data points as a result of engineering review times, beginning on March 26th, 2004 and ending on October 9th, 2007, yielding approximately three years and seven months of data. The data represent software code peer reviews from a major software development organization. The numerical values were arbitrarily scaled by a randomly chosen constant to protect proprietary performance information. Thus, the values are shown in units of \( C \times \text{hours/line of code} \), where \( C \) is the arbitrary scaling constant. The data points were measured on a group of projects that used a common, statistically stable development process which has been independently appraised as operating in conformance with CMMI level 5. The data were chosen in an unbiased manner, arbitrarily, and without any specific period in mind. We preconditioned the data to include only closed engineering reviews that were marked as “Closed” and had a “Review Closed Date”. Any reviews that had a value of zero were deleted from the data. In order to calculate the total engineering review time that was used as a basis for the research, we took the total Software Quality Assurance (SQA), preparation time and subtracted it from the total preparation time. We subtracted the SQA review time from the total review time. We multiplied the meeting duration with the meeting attendees (excluding SQA). We subtracted the total SQA post review time from the total post review time. We added the newly calculated results of preparation time, review time, meeting attendees duration, and post review time, to compute the engineering review time for each peer review taken into consideration for this study. The exclusion of SQA times is justified by the reasoning that SQA time does not belong in the calculation of the total engineering time.
The data were then divided into thirteen week periods in order to present the findings in quarterly management report format which represents three months of data analysis. The total weekly timeframe from March 2004 to October 2007 of the data, yielded 187 weeks. A total of 173 thirteen week increments were tabulated. Multiplied by a factor of two to account for the two operational scenarios to be analyzed, resulted in the equivalent of 346 individual cases generated and analyzed amounting to the equivalent of fourteen calendar years of hypothetical company operations. Adding the equivalent number of descriptive statistical data generated yielded 692 control charts and histograms used in this research paper. The first thirteen weeks were analyzed, followed by the second thirteen week period, which consisted of dropping the first week and adding the fourteenth week to the period. This second set of thirteen weeks is termed “weeks 2-14”. Following weeks 2-14, we dropped the second week and added week 15, so now we had weeks 3-15 to analyze. Each successive week followed the same pattern, for a total of 173 thirteen week increments.

The analysis from the data will be based on two cases comparing raw, untransformed, data vs. log-transformed data. From these two cases we will analyze two types of operational scenario data runs:

1) Raw, untransformed, data with no lock on the limits
2) Log-transformed data with no lock on the limits

Each thirteen week data series represents the amount of data in one weekly report with a moving quarterly window. The interest lies whether the findings of the other three organizational baselines and operational scenarios are more optimal after comparing and contrasting them.

6. Research Findings and Analysis

We have performed statistical process control measurements to establish results of the two experimental data sets that this research paper is focused on. These two experimental data sets acting as valid representations of organizational baselines and operational scenarios provided the grounds for analysis of the performances of each filter based on several key factors of consideration. This section will present an analysis of the results from our data runs and demonstrate our superior findings of our log-cost model approach in contrast to the raw, untransformed data. We are interested in answering several performance related questions as a result of these findings, such as:

i. What is in-control between the UCL and the LCL?
ii. What is out-of-control above the UCL?
iii. What is out-of-control below the LCL?
iv. When comparing the flags of the raw untransformed data and transformed data, with dynamic limits (no lock on the limits), are they the same, more, less?

We feel it is imperative that we discuss the results of the first thirteen week runs to determine if a trend appears to be visible in subsequent test runs. We began the analysis with the I-MR charts of the first thirteen weeks, for the raw data and the log-transformed data in order to get a good sense of the interpretation of these data from the starting point of this research as a baseline. The analysis of data runs covered periods between March 27, 2004 and October 10, 2007, which comprise all of the 2517 data points in this sample being studied. With each operational scenario, we displayed the resultant Minitab data runs of the engineering review times showing a graphical summary of the histogram with an overlaid normal curve so that we can get a better idea of how the data fits under a normal curve distribution within 95% confidence interval. The summary chart also displays the mean and standard deviation which we tabulated with each data run. The I-MR charts for the two scenarios considered are also shown along with all points that failed.

Shown below are the summary and the resultant I-MR chart runs from the first thirteen weeks for both the raw and log-transformed data operational scenarios (See Figures 1 & 2.) Desirable results indicate only true-positive flagged defects within three standard deviations from the mean under normally distributed limits. Undesirable results indicate both true-positive and false-positive flagged as defects discussed in detail under Chebyshev’s Inequality problem discussed in the Problem Statement of this paper As a result of the data runs from the first thirteen weeks, with 184 data points reviewed, we can clearly see that the log-transformed data runs produced the most desirable results when compared with the raw untransformed data runs. The percentage of in-control points between the UCL and LCL in the log-transformed runs were higher at 97.83% compared with the results from the runs of the raw untransformed data at 95.11% for the I chart and 90.76% with the MR chart. The mean and standard deviation of the log-transformed data contain more acceptable values when normalized under the log-transform. The percent DPU is lowest with the log-transformed data runs compared to the raw untransformed data runs (See Table 1).
**Test Results for I Chart of Engineering Review Time**

TEST 1. One point more than 3.00 standard deviations from center line. Test Failed at points: 1, 12, 21, 60, 78, 81, 86, 144, 168

**Test Results for MR Chart of Engineering Review Time**

TEST 1. One point more than 3.00 standard deviations from center line. Test Failed at points: 2, 12, 13, 21, 22, 60, 61, 78, 79, 81, 82, 86, 87, 144, 145, 168, 169

Figure 1: Weeks 1-13 Dynamic raw data results

**Test Results for I Chart of ln(Engineering Review Time)**

TEST 1. One point more than 3.00 standard deviations from center line. Test Failed at points: 1, 21, 60, 168

**Test Results for MR Chart of ln(Engineering Review Time)**

TEST 1. One point more than 3.00 standard deviations from center line. Test Failed at points: 2, 12, 60, 169

Figure 2: Weeks 1-13 Dynamic log-transformed data results

**Table 1: Counts and Percentages for the first 13 Weeks**

```
<table>
<thead>
<tr>
<th>Scenarios</th>
<th>μ</th>
<th>σ</th>
<th>In control between UCL &amp; LCL</th>
<th>% In control above UCL &amp; LCL</th>
<th>% Out-of-control above UCL</th>
<th>% Out-of-control below UCL</th>
<th>% Out-of-control below LCL</th>
<th>% DPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>I Chart of Raw Data</td>
<td>1.7682</td>
<td>2.3776</td>
<td>175</td>
<td>95.11</td>
<td>4.89</td>
<td>0</td>
<td>0.00</td>
<td>4.89</td>
</tr>
<tr>
<td>MR Chart of Raw Data</td>
<td>1.67</td>
<td>2.585</td>
<td>157</td>
<td>95.76</td>
<td>17</td>
<td>9.24</td>
<td>0.00</td>
<td>9.24</td>
</tr>
<tr>
<td>I Chart of Log-Transformed Data</td>
<td>0.20308</td>
<td>0.74499</td>
<td>180</td>
<td>97.83</td>
<td>4</td>
<td>2.17</td>
<td>0</td>
<td>2.17</td>
</tr>
<tr>
<td>MR Chart of Log-Transformed Data</td>
<td>0.20308</td>
<td>0.74499</td>
<td>180</td>
<td>97.83</td>
<td>4</td>
<td>2.17</td>
<td>0</td>
<td>2.17</td>
</tr>
</tbody>
</table>
```
As a result of the data runs from weeks 85-97, with 206 data points reviewed, we can clearly see that the log-transformed data run produced the most desirable results when compared with the raw untransformed data run. The percentage of in-control data points between the UCL and LCL in the log-transformed data runs with no locked limits, were higher at 98.54% for the I chart and 97.09% for the MR chart, when compared with the results from the other scenario run. In this case, three data points were out-of-control above the UCL in the I chart and six data points were out of control in the MR chart resulting in the lowest percent DPU of 1.46% in the I chart and 2.91% in the MR chart for the log-transformed data with no locked limits. All of these out-of-control data points were flagged in the other scenarios as well, as shown below from the results of the Minitab I-MR data runs.

The mean of the log-transformed data with no locked limits had the lowest value when compared with the other operational scenario, with \( \mu =0.03611 \). However, the lowest standard deviation value was noted in the I-MR chart of the log-transformed data with no locked limits. In terms of percent DPU, the best operational scenario is contained within the log-transformed data with no locked limits, with the lowest DPU of 1.46% for the I chart and 2.91% for the MR chart run when compared to 1.94% DPU for the I chart. The highest values for mean and standard deviation for these runs are contained in the raw data with no locked limits for the mean with \( \mu =1.9047 \) and in the raw data with no locked limits for the standard deviation with \( \sigma = 3.5992 \). The most undesirable operational scenario for weeks 85-97 is the run of the raw data with no lock on limits because it contains the highest percent DPU of 3.88% for the I-chart and 6.31% for the MR-chart. No data points were found to be out-of-control below the LCL in any of the four operational scenarios. Table 2 below summarizes our findings for weeks 85-97. Following, are the summary and I-MR chart results of the engineering review times for weeks 85-97 (Figures 6 & 7):

**Test Results for I Chart of Engineering Review Time**

**Test Results for MR Chart of Engineering Review Time**

**Test Results for I Chart of ln(Engineering Review Time)**

**Test Results for MR Chart of ln(Engineering Review Time)**

---

**Figure 3: Weeks 85-97 Dynamic raw data results**

**Figure 7: Weeks 85-97 Dynamic transformed data results**
In the end, the analysis consisted of 173 thirteen week data runs that were analyzed two times, raw versus log-transformed data at the point of the data analysis and presented a relative analysis of the data runs results to illustrate this process of testing. The results were analyzed and tabulated as a table of counts and percentages. The entire results of all 173 thirteen week operational scenarios were performed using Statistical Process Control of individual and moving range charts. The transformed data, and each time with one of the operational scenarios under study. The data runs of the two operational scenarios were simulated in this research, and then following is a selection of every thirteen new thirteen week runs. Therefore, from the first thirteen weeks were run through the statistical process control test and analyzed the results. What followed was a drop of the first week and the addition of the fourteenth week and re-run the SPC tests and analyzed the results. This first thirteen weeks were run automatically with each data run, or “dynamically”. A total of 185 weeks were analyzed from an arbitrary period of March 2004 through October 2007. As a customary management preference to view reports in quarterly periods, the sample of 185 weeks were further divided into thirteen weekly sections to simulate three months of data analysis. The first thirteen weeks were run through the statistical process control test and analyzed the results. What followed was a drop of the first week and the addition of the fourteenth week and re-run the SPC tests and analyzed the results. This pattern continued in the fashion of 3-15, 4-16, 5-17, and so on until weeks 173-185.

In the end, the analysis consisted of 173 thirteen week data runs that were analyzed two times, raw versus log-transformed data, and each time with one of the operational scenarios under study. The data runs of the two operational scenarios were performed using Statistical Process Control of individual and moving range charts. The results were analyzed and tabulated as a table of counts and percentages. The entire results of all 173 thirteen week simulation runs are available in their entirety. A selective warm up period of the first twenty thirteen weeks were run through the statistical process control test and analyzed the results. What followed was a drop of the first week and the addition of the fourteenth week and re-run the SPC tests and analyzed the results. This pattern continued in the fashion of 3-15, 4-16, 5-17, and so on until weeks 173-185.

In the end, the analysis consisted of 173 thirteen week data runs that were analyzed two times, raw versus log-transformed data, and each time with one of the operational scenarios under study. The data runs of the two operational scenarios were performed using Statistical Process Control of individual and moving range charts. The results were analyzed and tabulated as a table of counts and percentages. The entire results of all 173 thirteen week simulation runs are available in their entirety. A selective warm up period of the first twenty thirteen weeks are discussed in the research, and then following is a selection of every thirteen new thirteen week runs. Therefore, from week 20-32, we tested weeks 33-45, 46-58, 59-71, 72-84, 85-97, 98-110, 111-123, 124-136, 137-149, 150-162, 163-175, and the last thirteen week section for weeks 173-185. In this paper, we depicted weeks 85-97 about the midway point of the data analysis and presented a relative analysis of the data runs which results to illustrate this process of testing.

### Table 2: Counts and Percentages for Weeks 85-97

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>In-control between UCL &amp; LCL</th>
<th>% In-control between UCL &amp; LCL</th>
<th>Out-of-control above UCL</th>
<th>% Out-of-control above UCL</th>
<th>Out-of-control below LCL</th>
<th>% Out-of-control below LCL</th>
<th>% DPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>I Chart of Raw Data No Lock on Limits</td>
<td>198</td>
<td>96.12%</td>
<td>8</td>
<td>3.88%</td>
<td>0</td>
<td>0%</td>
<td>3.88%</td>
</tr>
<tr>
<td>MR Chart of Raw Data No Lock on Limits</td>
<td>193</td>
<td>93.69%</td>
<td>13</td>
<td>6.31%</td>
<td>0</td>
<td>0%</td>
<td>6.31%</td>
</tr>
<tr>
<td>I Chart of Transformed Data No Lock on Limits</td>
<td>203</td>
<td>98.54%</td>
<td>3</td>
<td>1.46%</td>
<td>0</td>
<td>0%</td>
<td>1.46%</td>
</tr>
<tr>
<td>MR Chart of Transformed Data No Lock on Limits</td>
<td>200</td>
<td>97.09%</td>
<td>6</td>
<td>2.91%</td>
<td>0</td>
<td>0%</td>
<td>2.91%</td>
</tr>
</tbody>
</table>

### 7. Summary

The purpose of this research paper was to evaluate two operational case scenarios with the objective to determine the best and most optimal operational scenario in order to detect true-positive defects when conducting engineering peer reviews during the software development life cycle and minimizing false positive detections as defects. The two cases consisted of raw data and log-transformed data. Within these two cases, one operational scenario was analyzed no lock on the limits, hence the term “dynamic” which indicates that the mean and standard deviation is calculated automatically with each data run, or “dynamically”. A total of 185 weeks were analyzed from an arbitrary period of March 2004 through October 2007. As a customary management preference to view reports in quarterly periods, the sample of 185 weeks were further divided into thirteen weekly sections to simulate three months of data analysis. The first thirteen weeks were run through the statistical process control test and analyzed the results. What followed was a drop of the first week and the addition of the fourteenth week and re-run the SPC tests and analyzed the results. This pattern continued in the fashion of 3-15, 4-16, 5-17, and so on until weeks 173-185.

7. **Summary**

The purpose of this research paper was to evaluate two operational case scenarios with the objective to determine the best and most optimal operational scenario in order to detect true-positive defects when conducting engineering peer reviews during the software development life cycle and minimizing false positive detections as defects. The two cases consisted of raw data and log-transformed data. Within these two cases, one operational scenario was analyzed no lock on the limits, hence the term “dynamic” which indicates that the mean and standard deviation is calculated automatically with each data run, or “dynamically”. A total of 185 weeks were analyzed from an arbitrary period of March 2004 through October 2007. As a customary management preference to view reports in quarterly periods, the sample of 185 weeks were further divided into thirteen weekly sections to simulate three months of data analysis. The first thirteen weeks were run through the statistical process control test and analyzed the results. What followed was a drop of the first week and the addition of the fourteenth week and re-run the SPC tests and analyzed the results. This pattern continued in the fashion of 3-15, 4-16, 5-17, and so on until weeks 173-185.

In the end, the analysis consisted of 173 thirteen week data runs that were analyzed two times, raw versus log-transformed data, and each time with one of the operational scenarios under study. The data runs of the two operational scenarios were performed using Statistical Process Control of individual and moving range charts. The results were analyzed and tabulated as a table of counts and percentages. The entire results of all 173 thirteen week simulation runs are available in their entirety. A selective warm up period of the first twenty thirteen weeks are discussed in the research, and then following is a selection of every thirteen new thirteen week runs. Therefore, from week 20-32, we tested weeks 33-45, 46-58, 59-71, 72-84, 85-97, 98-110, 111-123, 124-136, 137-149, 150-162, 163-175, and the last thirteen week section for weeks 173-185. In this paper, we depicted weeks 85-97 about the midway point of the data analysis and presented a relative analysis of the data runs which results to illustrate this process of testing.

### 8. Conclusion, Contributions, and Future Work

Weeks one through thirteen produced 9 defects with the raw data, and 4 defects with the log-transformed data. The period between weeks 85-97 produced 8 defects with the raw data, and 3 defects with the log-transformed data. This shows us that the organization saves 62.5% of the rework time that would have been spent with the raw data analysis method instead, in both cases. It also tells us that the true defects required 37.5% of the rework time spent with the log-transformed analysis method. In evaluating the two cases with the two operational scenarios that were analyzed in this research paper, it is evident that the clear “winner” is the second case with the log-transformed data.

Out of the log-transformed operational scenario the outcome was very interesting. In terms of percent defects per unit, the most desirable outcomes for the first 16 quarterly data runs, that is from weeks 1-13 up to and including weeks 16-28 were clearly those of the I-MR charts of the log transformed data with locked limits which were presented at the 2013 ISERC findings, with more stable process results in the MR charts. However, what makes this outcome very interesting is that from weeks 17-29 onwards to the end, clearly the “winner” and best operational scenario has been consistently the log-transformed data with no locked limits. We say “interesting” because as in any test bed scenarios using simulations all weekly periods covering up to weeks 17-29, there is a warm-up period and we think that the first sixteen weekly periods were a warm up for our entire 173 week periods simulated in this research. Further research could examine a similar operational scenario using a lean and agile process in the development life cycle of a product.
References
8. Institute of Electrical and Electronics Engineers. http://www.ieee.org/
25. Institute of Electrical and Electronics Engineers. http://www.ieee.org/