Monitoring the Stability of the Processes in Defined Level Software Companies Using Control Charts with Three Sigma Limits

G.VIJAYA
Information Technology,Anna University
KGiSL Institute of Information Technology, Saravanampatti
Coimbatore-35, TN, INDIA
vijaya_ravichandran@yahoo.com

S.ARUMUGAM
Nandha College of Engineering, Erode, TN
INDIA

Abstract— Monitoring the stability of the software process in the lower level companies is a challenging issue to software engineers. In this paper, SPC is applied to software metrics. Defect Density, Review Performance and Rework percentage and results after applying the SPC to various processes of software are discussed and analyzed, using control charts, the most sophisticated tools of SPC. The difficulties in the application of Statistical Process Control to lower level software organization are observed and relevant suggestions are provided.

Keywords: *Control Chart, Statistical Process control (SPC), Defect density, Inspection Performance, Rework percentage,, CMM.*

1 Introduction

A solution that is adapted by software industries for maintaining and improving the software processes is called Statistical Process Control (SPC). SPC is a set of tools for managing the processes, and hence, determining and monitoring the quality of the outputs of an organization. SPC is a time-tested and effective control scheme used for process capability analysis and process monitoring. Even though, SPC is used since 1930 with the idea of applying SPC to software development it became effective only from the middle years of 1990's. Earlier, it was mainly used by the manufacturing companies; but today, it is involved in the software development companies too.

2 CMM Level and SPC

The Software Engineering Institute (SEI) and Capability Maturity Model (CMM) mandate that SPC can be used in Level 4 organizations. As the maturity process of software organization is too long, many organizations stopped using SPC. It would imply that emerging organizations have to wait till they reach maturity stage before applying SPC techniques to their software processes.

It is possible to perform SPC in CMM level 1[13]. The authors base their claim on the idea that if a process is defined and performed consistently, the outcome of SPC would be meaningful. Presently a case study is performed on the application of SPC techniques using existing measurement data in a CMM Level 3 software organization. The control chart ('u' chart) with 3-sigma control limits is used to demonstrate the practical evidence on the utilization of SPC.

The application of SPC techniques for software is rare due to prerequisites such as high maturity, rational sampling, and effective metric selection. The existing study reports result from their own implementations and provide suggestions for success. During this study, approaches used for assessing the suitability of software process and metrics for starting SPC implementation via control charts are assessed. The approach includes the guidance given to identify rational samples of a process and ways to select process metrics.

This paper gives the solution to the problem of CMM Level 3companies as to how they could manage their processes using control charts with three sigma limits.

3 Significance of the Research

The study relied on the case study method. This method was found to be useful in low level companies to achieve the CMM level 4. Three software metrics, namely defect density, inspection performance and rework percentage were selected. The metric data were collected from the trouble

report of requirement documents and design documents. A total of seven projects were selected for SPC analysis. From the results obtained, a set of guidelines were formulated to help the programmers, analysts and organizations to use SPC in their organizations successfully.

As per the CMM level standards, SPC can't be utilized in lower level companies to maintain the process stability. Three metrics have been selected and analyzed for the existing data in the CMM level 3 companies and it was found from the analysis that it can be applied for low level companies.

4 Analyses and Interpretation

U-chart and 3-sigma limits are used extensively in the present study. U-chart is a data analysis technique for determining if a measurement process has gone out of statistical control. The u-chart is sensitive to changes in the normalized number of defective items in the measurement process. Here, 'normalized' means the result of number of defectives divided by the unit area. The U in u-chart stands for units as in defectives per lot. The U-control chart consists of:

- Vertical axis = the normalized number of defectives (number of defectives/area for sub-group = u) for each sub-group;
- Horizontal axis = sub-group designation.

A sub-group is a time sequence frequently (e.g., the number of defectives in a daily production runs where each day is considered a subgroup). If the times are equally spaced, the horizontal axis variable can be generated as a sequence.

U charts consist of three guidelines, centre line, a lower control limit and an upper control limit. The center line is the average number of occurrences-per-unit and the two control limits are set at plus and minus three standard deviation. If the process is in the statistical control, virtually all subgroup occurrences-per-unit should be between the control limits and they should fluctuate at random about the center line. The sample control chart is given below in Fig 1.

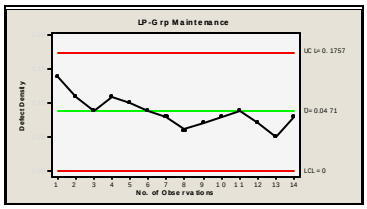

Fig1 Sample Control Chart

Presently, eight tests were defined to identify out of control software processes and are given in Table I.

TABLE 1 TEST DEFINITIONS

Test	K	Definition
a	3	1 Point > K standard deviations from centre line
b	9	K points in a row on same side of centre line
c	6	K points in a row, all increasing or all decreasing
d	14	K Points in a row, alternating up or down
e	2	K out of K+1 points > 2 standard deviations from centre line on the same side
f	4	K out of K+1 points > 1 standard deviations from centre line on the same side
g	15	K points in a row within 1 standard deviation of centre line (either side)
h	8	K points in a row > 1 standard deviation of centre line (either side)

4.1 Defect Density

Defect density data was obtained from the review, test and audit meetings. Data were obtained mainly through Trouble Reports. Two types of defects were considered.

(i) Code defects
(ii) Document defects

These defects were obtained from the requirements and design documents. The data collected were restricted to requirements and design documents and were obtained from documents listed below.

1. Software Requirements Specification and IRS-Interface Requirements Specification)
2. Design Documents (SDD and IDD): The number of pages used to compute size. (SDD- Software Design Description and IDD-Interface Design Description)
3. Requirements documents (SRS and IRS): The number of requirements is used to compute size. (SRS - Problem Reports and Document Change Request (DCR) Reports were the two main sources of defect data collection reports for code and document defects respectively.

The data collected was restricted to requirements and design documents and were obtained from documents listed below.

1. Software Requirements Specification and IRS-Interface Requirements Specification)
2. Design Documents (SDD and IDD): The number of pages used to compute size. (SDD- Software Design Description and IDD-Interface Design Description)
3. Requirements Documents (SRS and IRS): The number of requirements is used to compute size.

(SRS- Problem Reports and Document Change Request (DCR) Reports were the two main sources of defect data collection reports for code and document defects respectively.

The reports had basic information like work product, related project phase, defect priority, initiation and closure date were collected. Defect density was consolidated according to the priority assigned to it.

There were five priority levels maintained by the company, namely, Urgent, High, Medium, Low and Not Applicable. As the number of samples with Urgent priority was very small and was insufficient for SPC analysis, Urgent and High priorities were grouped together. The 'Not Applicable' priority was assigned to defects that were not related to the project and hence were ignored during defect density calculation. Thus, during consolidation, the defect priorities were grouped into three main categories, namely, 'High (HP-Grp), 'Medium' (MP-Grp) and 'Low' (LP-Grp). Few graphs of metric defect density are presented for analysis and interpretations.

4.1.1 Requirements Documents

4.1.1.1 HP-Grp: Implementation & Maintenance

Figure 4.1 shows the control chart for HP-Grp defects (High Priority Defects) obtained from defect density measures of requirement documents for implementation phase and it can be noted that there are two situations which show deviations from centre line.

Situation 1: Test Failed at points: 5, 6 (Test No. 1)
Situation 2: Test Failed at point: 11 (Test No. 3)
A similar pattern was observed in the maintenance phase also (Fig 4.2). The control chart takes into consideration the defects obtained from the IRS and SRS documents for the seven projects selected.

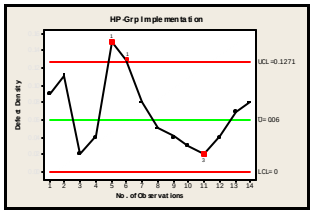

Fig4.1:HP-GrpImplementation-Requirement Documents)

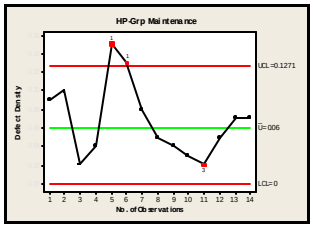

Fig4.2:HP-GrpMaintenance (Requirement Documents)

Interpretation

As per the fig 4.1, the point 5 and 6 refer to project 3 IRS and SRS respectively, while point 11 refers to project 6 IRS document. While trying to find the reason for such behaviour, it was found that both project 3 and 6 are from the same customer and the company was lenient towards their demand for new additions and modifications during requirement analysis phase. Modifications and requirements were done according to the adhoc requests and these were reflected in future defects in requirement documents.

As with requirement documents, the number of documents were consolidated into three groups, HP-Grp, MP-Grp and LP-Grp consolidated according to their priority. As mentioned earlier, these data were collected from the SDD and IDD documents and as the data is collected cumulatively from one phase to the other, the graphs for implementation and maintenance phases looked quite alike.

4.1.2.1 HP-Grp : Implementation & Maintenance

Figures 4.3 and 4.4 show the control charts for High priority group obtained from Software and Interface Design Documents.

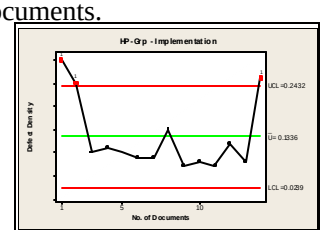

Fig 4.3 : HP-Grp Implementation (Design Documents)

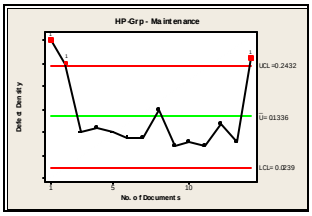

Fig 4.4: HP-Grp Maintenance (Design Documents)

In the figure 4.3 and 4.4, points 1, 2 and 14 are depicted as out of control limit conditions as they failed in Test No. 1 with deviations from centre line. These points belonged to the SDD and IDD of Project 1 and IDD of Project 14.

Interpretation

While investigating for the reason for the deviations located in the figures 4.3 and 4.4, it was found that there were major structural changes in Project 1 and 7 system level documents and the software components were divided into different builds and this resulted in a high defect density. When the components were divided, the integration

needed between subsequent builds also became complex and products turned out to be more defective. One another reason, is that the staff turn over rate was high for project 1 and 14, which was one another important reason for high defect density. The frequent change in staff team was because both the projects were in-house projects, and were frequently interrupted. These interruptions showed high time delay during the development process. The same trend was observed for HP-Grp maintenance also. Obtaining similar charts prove that most of the defects in this group were detected before maintenance phase itself.

4.1.2.2: MP-Grp: Implementation & Maintenance

The trend of defects on medium priority group on implementation and maintenance phases are shown pictorially in Fig 3.11 and 3.12.

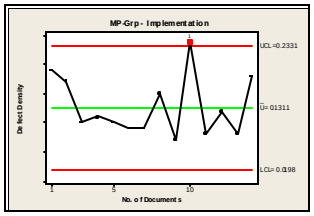

Fig3.11:MP-Grp Implementation(Design Documents)

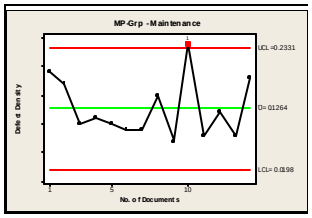

Fig 3.12: MP-Grp Maintenance (Design Documents)

Interpretation

From the figure, it can be clearly seen that, the Interface Design Description document of Project 5 has raised defects which is reflected as out of control limit (Point 11).

While analyzing reasons for this out of control behaviour, it was found that the analyzers designed an inapplicable data schema or structure and the resulting data model did not match with customer requests. This was reflected as a huge number of defects in the problem document and is exposed correctly in the control chart. To further analyze the situation, this data was removed from the dataset and new control limits were calculated. Another control chart with newly calculated control limits was drawn (Fig 3.13).This resulted with a control chart with all points in stable conditions. A very similar trend was obtained for Implementation also.

4.1.2.3 LP-Grp: Implementation & Maintenance

The charts in Fig 3.14 and 3.15 show the low priority group defects for implementation and maintenance.

Interpretation

The figures reflect that all points are within the control limit and therefore they are in stable conditions. This indicates a stable maintenance and implementation process.

In conclusion, it was found that a very high variability in defect density values of different documents existed, which indicates that the software processes does not offer stability even though the processes are well defined and defects are correctly recorded.

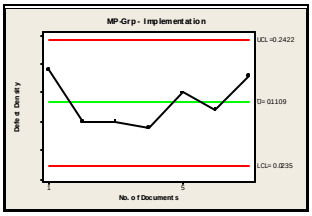

Fig3.13:MP-Grp Implementation (SDD)

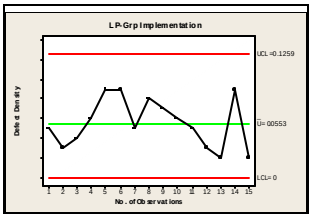

Fig3.15:LP-Grp Implementation (Design Documents)

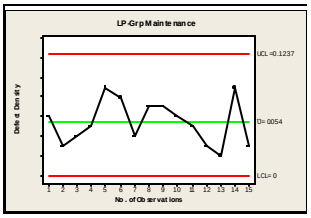

Fig3.16:LP-Grp Maintenance (Design Documents)

On the other hand, the control charts proved to provide a better understanding of the product status and allowed comparison with respect to the control limits.

4.2 Inspection Performance

Inspection in software engineering refers to peer review of any work product by trained individuals who look for defects using a well defined process (Wong, 2006). Software inspections have proven very effective in capturing defects early enough to avoid the cost of rework.

The inspection process was developed by Michael Fagan in the mid-1970s and it has later been extended and modified. There are two related

inspection types, namely, code review and peer review.

Peer Reviews are considered an industry best-practice for detecting software defects early and learning about software artifacts (Cohen, 2006). The elements of Peer Reviews include the structured review process, standard of excellence product checklists, defined roles of participants, and the forms and reports. The most commonly used analysis tool in inspection / review process is the control charts (Jalote and Saxena,2002).

In the present work, peer review data was used to analyze the inspection performance in the company. Peer review is the use of other people to examine an existing work product. The company had three main reasons for conducting peer review and is listed below.

1. Imparting information
2. Gaining approval or consensus
3. Locating issues and defects

The company had separate reviewers to review the product and the review process was carried out by reviewers and at the end of each review a "Review Report" was prepared. The documents such as SRS, SDD, IRS, IDD and program code were inspected during this process. The "Review Report" had details like, name of the reviewer, time of review, number of problems found, review results, etc.Three types of peer reviews conducted and were performed at different points of products life cycle. They were,

(i)Initial Review (IR)
(ii) Additional Review (AR)
(iii) Verification Inspection (VI)

The goal of IR is to perform an ad-hoc type review, usually conducted in 1-2 hours types. This is conducted in-house before the software product is released to the customer.

The goal of AR is to bring the customer and the software product and to verify whether the finished product is according to the customer requirement. The outcome of this review is a

(i) Positive Review (no defect and hence product can be released)
(ii) Negative Review (defects identified)

The second outcome initiates the preparation of the Trouble Report with suggestions from customer, which will be reviewed in VI review for correctness and appropriateness.

As most of the defects are fixed during a IR, the product is expected to have fewer defects before entering the AR. Therefore, the reviewers will most probably find fewer defects during AR in contrast IR, and the review effectiveness will seem to be worse. On the other hand, different trouble reports are used in order to record the defects that are found during IR and AR. As the defects found during a IR are not categorized with respect to their priorities, the analysis is separated for IR and AR.

Moreover the reviewers will not find even a Single defect in most of the change peer reviews. Therefore, it will not be rational to judge a Change peer review as ineffective although no defect has been found. As the aim of using this measure is not to evaluate the product but the review process, Change peer review is left aside in the analysis. A code review can be done as a special kind of inspection in which the team examines a sample of code and fixes any defects in it. In a code review, a defect is a block of code which does not properly implement its requirements, which does not function as the programmer intended, or which is not incorrect but could be improved (for example, it could be made more readable or its performance could be improved)

In addition to helping teams find and fix bugs, code reviews which are useful for both cross-training programmers on the code being reviewed and for helping junior developers learn new programming techniques. For these reasons, reviews for code and different document types are analyzed separately.

The collected review data were ordered according to time for each review type for each product. This method was followed in accordance with Pandian (2004), who suggested that for an effective analysis, the data has to be arranged in some chronological order, which results in giving vital meaning and power to control charts.

The review effectiveness was calculated by dividing the number of defects by review times (minutes) for each review. The u charts with 3-sigma control limits were constructed separately for each type review and for SDD, SRS, UITD (Unit Integration Test Description), UTD (Unit Test Description) documents and for code.

While conducting the analysis, Project 1 and Project 7 were not taken into consideration, as they were in-house projects and reflected huge variations. Projects 3 and 6 were developed for the same customer and had interrelated processes. The AR had to be combined, which was time consuming and could not be taken into consideration during the study period. The review process for Projects 2 and 4 were not completed and therefore the analysis was restricted to a single project (Project 5) and the results are discussed in the following sections.

- **IR – SDD**

The initial review performance on SDD document for Project 5 is shown in Figure 3.17.

Interpretation

From the chart, it can be seen that Point No. 1 is out of control and further investigation revealed the following facts.

Project 5 has some major interface design changes that reflected major changes in the design and coding. It can be remembered that the defect density control chart for MP-Grp for Project 5 also showed a deviation (Figures 3.5 and 3.6). While probing into the reasoning for this change, it was found that there were changes in data schema that did not match customer requirement, and further it was found that the main reason attributed to this change was the frequent employee turnover rate during the development process. These changes in the interface design reflected with a huge number of critical modifications during subsequent implementations.

- **AR - SDD**

Fig 3.18 shows the control chart drawn to determine the performance of AR conducted using SDD document.

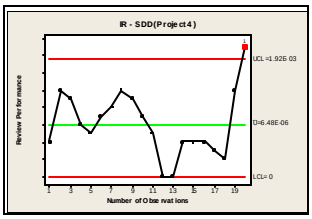

Fig 3.17: Initial Review on SDD Document

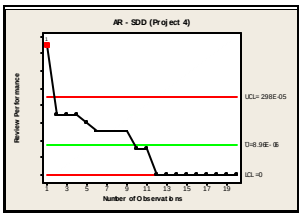

Fig 3.18: Additional Review on SDD Document

Interpretation

From the figure, it can be seen that the review performance has gradually stabilized with Point No. 1 showing out-of-control situation. Upon investigation, it was found that the reviewer is very experienced and had found maximum errors. The decrease in the number of errors is also another proof on the accomplishment of the first review and indicates that most of the errors were found in Review No. 1 itself. Thus, this chart shows the importance of an experience reviewer for an effectiveness and efficiency during a review process.

- **AR and IR on UTD**

The performance of the initial review conducted using UTD document is illustrated by the control chart in Figure 3.19 and Figure 4.20 shows control chart for the UTD Additional Review.

Interpretation for IR and AR on UTD

The lower and upper control limits for IR on UTD are 0 and 3.0E-4, respectively. Therefore, the review performance points are expected to fall between 0 and 3.0E-4. The center line is 6.98E-05.

The lower and upper control limits for AR on UTD are 0 and 1.83E-04, respectively. Therefore, the review performance points are expected to fall between 0 and 1.83E-4. The center line is 3.23E-05.

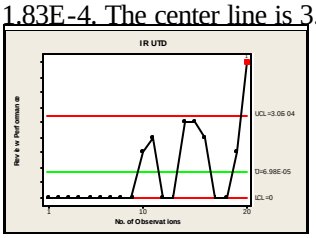

Fig 3.19 : Initial Review on UTD Document

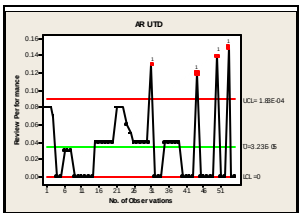

Fig 3.20 : Additional Review on UTD Document

In Figure 3.19, one point failed Test No. 1 and five tests in Figure 3.20 were detected as special causes. All these points are located above the upper control limit and failed Test 1 because they are more than 3-σ away from the center line. As Test 1 is the strongest indicator of an out-of-control process, it is concluded that the chart needs further analysis.

Further analysis of the Review Summary Reports revealed that the majority was minor defects, such as formatting / grammatical, which even though is important resulted in a huge number of defects. The reviewers reported this as lack of consistency. It was also found that these reviewers were non-technical managerial persons and were concentrated more on non-technical issues.

As more critical defects are found normally with technical issues, it was decided to modify Figure 3.20 to concentrate on technical issues only, by removing the out-of-control points. The resulting graph is shown in Figure 3.21.

Redrawing Figure 3.20 to Figure 3.21, revealed five other points (Points 1, 2, 21, 22 and 23) failing in Test 1. Analysis of this situation leads to the fact that the project is not using common code implementations. This increased the number of errors, as they are repeated again and again in the project. Errors found in such functions were high and were directly proportional to the number of

times it was being used. It was also found that the higher final peer review performances achieved by the reviewers who also took part in the previous draft peer reviews. As they worked on the document before, they were more efficient in the second review.

- **IR and AR on UITD**

Considering the UITD documents produced the control charts in Figures 3.22 and 3.23 for the Initial Review and Additional Review documents.

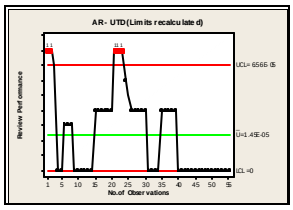

Fig 3.21: Modified AR UTD

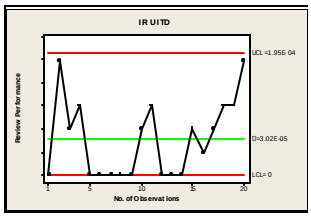

Fig 3.22 : IR UITD

Interpretation

It can be seen from Figures 3.22 and 3.23, no points are outside control limit and hence can be concluded that the UTTD review performance was highly satisfactory.

- **IR and AR on Code**

The control charts constructed for the code review performance analysis for IR and AR are shown in Figures 3.24 and 3.25 respectively.

Interpretation

The figures indicate that there are six out of 3-σ limits situations. On probing for reason, it was found that most of the errors found were similar to that of UTD and UITD. According to Li *et al.* (2006), while performing reviews, if the same reviewers perform the process repeatedly, they tend to concentrate on the same defects. He also pointed out that since they were more familiar with the defects they could find the defects quickly.

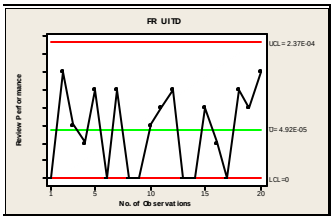

Fig 3.23 : AR

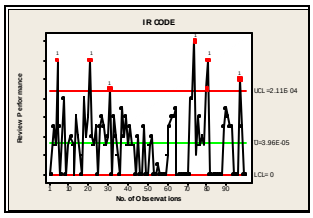

Fig 3.24 : IR Code

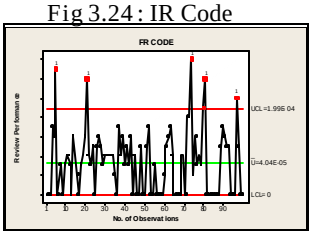

Fig 3.25 : AR Code

The study conducted reveals the fact that SPC techniques and in particular, the usage of control charts is effective in identifying out-of control situations and stable conditions. Further, as expected, it could also be seen that the number of errors detected during IR is more than AR.

4.3 Rework Percentage

Rework is generally considered to be potentially avoidable work that is triggered to correct problems or to tune an application (Butler and Lipke, 2000). The cost of rework can approach or exceed 50% of total project cost (Haley *et al*., 1995). Research on rework has focused on minimizing the amount of rework that a project may incur (King and Diaz, 2002). This is typically done through the introduction of earlier, more frequent, and more formal reviews, inspections, and tests; these aim to detect and enable the correction of problems as early in the life cycle as possible. Despite successes in reducing rework, it is generally accepted that rework cannot be eliminated entirely. Moreover, not all rework-inducing problems can be detected as soon as they occur; some problems will only be caught some distance downstream. Thus, some amount of rework is inevitable. In this research the problem of rework is attacked through the use of control charts (Cass *et al*., 2003).

Rework is defined as any modification to configuration items after IR of the first release and changes to internal/external baselines. The data for rework analysis is obtained from Problem Reports and Document Change Requests, as it is believed that if any defect recorded on a Problem Reports and Document Change Requests, is a cause rework. The total problem resolution time, which is the rework effort in terms of man-hours, is recorded on these forms. Thus, the total rework effort can be calculated by summing up the efforts on trouble reports within the dates corresponding to related

project phases. The timesheet data is also collected daily for each individual and effort amounts in project level can be obtained and therefore the study is limited to measure rework percentages within specific time intervals.

As the rework effort is calculated for more than three years in the company and the results are utilized for process improvement studies, it is assumed that there is a firm understanding on the meaning of measurement. However, as the causes of defects are not recorded on the trouble reports, the opportunity to make an analysis on the rework percentages related to different project phases was not available. Therefore, the present study was limited to measure rework percentages within specific time intervals.

Trouble Report Document (TRD) was the main document from which data was collected. A TRD includes details of many defect items on the same form. The data was analyzed to be in two stages.

1. TRD open for a period less than a week.
2. TRD open for more than a week.

For the first category, the rework and total effort is recorded for each week for each project. For the second category, an assumption that the effort amount is uniformly distributed among different work days is made and thus, a weighted effort for the weeks is calculated, where the weight is determined by the number of days that the trouble report stayed open in the corresponding week. For instance, if the initiation date of a trouble report is 22nd of May - Thursday and closure date is 27th of May - Tuesday, 33.3 % of total effort is counted for week 2, and 66.6 % for week 1 (4 days from Thursday to Sunday; 2 days from Monday to Tuesday).

As the rework amounts increase during inspection and testing periods, the variation with respect to different points in the life cycle can be regarded natural. In order to smooth out the effect of this natural variation, it is decided to perform the analysis on a four-week period through which the rework percentage is assumed to be within certain limits. Therefore, four consecutive weekly rework and total effort amounts are summed up separately and rework percentages for each of these four-week periods are derived. Moreover, the documentation and coding rework percentages are analyzed separately as they would possess different trend characteristics.

- **Analysis of Project 1**

Figures 3.26 and 3.27 shows the control charts for project code rework data and document rework percentage data.

Interpretation

The control charts reveal the fact there are 3 out-of-limit points in Figure 3.26 (code rework) and 2 out of limit points in Fig 4.27(document rework) The reason behind such behavior is that, major structural changes and coding practices were performed during this period.

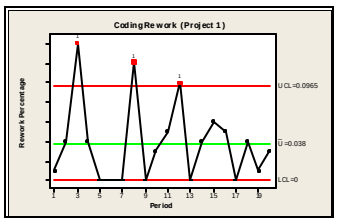

Fig 3.26 : Code Rework Performance

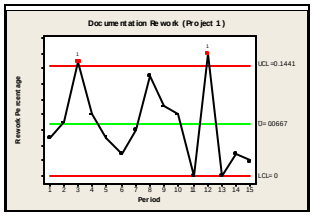

Fig 3.27 : Document Rework Performance

Interviewing the senior programmer and Systems Analysts exposed the fact that, as this project is an in-house project, the organization wanted to incorporate all the latest and efficient techniques, which resulted in such a change.

- **Analysis of Projects 2 to 6**

The behaviour pattern with respect to code rework percentage and document rework percentage is analyzed for the projects from 2-6.

Interpretation

All control graphs show no out-of-control situation in code and documentation rework data and therefore are considered to be stable and not presented here.

- **Analysis of Project 7**

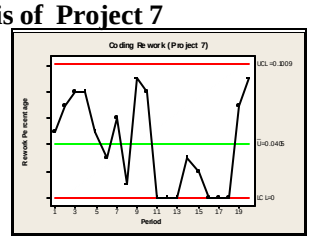

Fig 3.38 : Project 7 Coding Rework

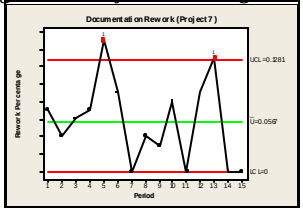

Fig 3.39 : Project 7 Documentation Rework

Figures 3.38 and 3.39 show the control chart for code rework and document rework of project 7 respectively.

Interpretation

No out-of-control situation is observed in code rework data (Figure 3.38). However, the documentation rework exceeds the upper limits at two points (Figure 3.39). Again a meeting with the technical persons revealed the following facts:

- Project is temporarily suspended quite a few times due to unavailability of staff
- Delay in development caused changes in requirements

With all these findings, the rework percentage data can be safely regarded as an outlier. From the analysis it can be deduced that:

- Changes in the structure of a project (the life cycle and architecture) result in high rework amounts.
- Scope changes result in great amounts of rework.
- High turnover rate is an important reason for high rework amounts.
- Internal projects and the projects that are performed to partner organizations are apt to more rework in the later phases. Therefore, special attention should be devoted to these projects considering the customer's relaxed approach.
- The lack of experience causes high rework.
- User interface prototypes without navigation may mislead the customer and cause more rework in the later phases.

The findings of this metric are obvious and thus, prove the successful implementation of control charts to detect out-of-control situations that cause high rework in the projects. The results provide two-way advantage on the usage of control chart.

1. Increases the reliability on control charts to capture deviation in software processes.
2. Act as a basis for improvement of processes to improve employee morality (as they show deviations in scientific manner) and becomes the part of overall preventive action activities if it is applied continuously on ongoing projects.

5 Case Study Results

The research work uses 3-sigma to calculate the upper and lower limits to draw a control chart. To evaluate the performance efficiency obtained by using 3-sigma limits, all the experiments were conducted twice, with ±3 and with ±2standard deviation limits. The experiments are conducted with all the three metrics. Results obtained for each of the metric selected is consolidated according to the number of tests failed and is presented in Table2, 3 and 4 for metrics defect density, inspection performance and rework percentage respectively.

Table 2 : Defect Density

S.No.	*Software Metric*	±2 s	±3 s
REQUIREMENT DOCUMENTS			
1	*HP-Grp Implementation*	4	2
2	*HP-Grp Maintenance*	6	2
3	*MP-Grp Implementation*	0	0
4	*MP-Grp Maintenance*	0	0
5	*LP-Grp Implementation*	1	0
6	*LP-Grp Maintenance*	1	0
DESIGN DOCUMENTS			
7	*HP-Grp Implementation*	5	2
8	*HP-Grp Maintenance*	5	3
9	*MP-Grp Implementation*	4	1
10	*MP-Grp Maintenance*	3	1
11	*LP-Grp Implementation*	0	0
12	*LP-Grp Maintenance*	0	0

Table 3 : Inspection performance

S.No.	Software Metric	±2 s	±3 s
INITIAL REVIEW			
1	SDD	1	1
2	UITD	0	0
3	Code	14	7
ADDITIONAL REVIEW			
4	SDD	9	5
5	UITD	2	0
6	Code	12	6

Table 4 : Rework Percentage

Project No	Code Rework		Document Rework	
	±2 s	±3 s	±2 s	±3 s
1	4	3	4	2
2	1	0	0	0
3	0	0	0	0
4	2	0	3	0
5	4	0	6	0
6	0	0	0	0
7	1	0	8	2

Two kinds of mistakes are commonly encountered while using control charts with SPC. The first is the mistaken identity of common cause as special cause variation and the second is the mistaken identification of special cause variation as a common cause variation. Both situations of mistaken identifications are costly and will result in substandard software product. In either case, the economic loss is large and the creation of substandard software product is inevitable. From the results shown, it is evident that the usage of 3-sigma minimizes the number of false alarms, thus

indirectly reduces the total cost from both overcorrecting and under-correcting.

6. Conclusion

The study provides a practical insight to the debate on whether it is necessary and sufficient to have a high maturity level for successful SPC implementation. Some metric data was collected when the organization is at CMM Level 2 and the results produced for these data were also successful. This proves that having a high maturity level might not be necessary for utilizing SPC in a software organization. We are able to achieve the following: Determine what the core process are and determine if they in control, Measure the behavior of those processes that are out of control, Use different types of control charts for different applications, Interpret variation in processes and develop strategies to reduce variation.

Experimental results proved that the charts are efficient in the maintenance of software quality and can be used by lower level software industries, EDP managers and senior programmers to achieve CMM level 4 before attain the maturity period.

References

[1] Sutherland, J., Devor, R., **Chang, T.,** ***Statistical Quality Design and*** *Control.* Prentice Hall Publishing Company, 1992. ISBN: 002329180X.

[2] Card, D.,"*Statistical Process Control for Software?*" IEEE Software, May 1994, PP 95-97.

[3] Kan, S.H.,*Metrics and Models in Software Quality Engineering.* Addison-Wesley Publishing Company, 1995. ISBN 0-201-63339-6.

[4] Crosby, P.B., *Quality is Free: The Art of Making Quality Certain*, Penguin Book USA Inc, January 1980. ISBN: 0-451-62585-4.

[5] Paulk, M. C., Curtis, B., Chrissis, M. B., Weber, C. V. *Capability Maturity Model for Software, Version 1.1 (CMU/SEI-93-TR-024, ADA 263403).*Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University,1993.

[6] Carleton, A., *"Statistical Process Control for Software (Software Technology Review)*". Carnegie MellonUniversity,2001.(URL: http://www.sei.cmu.edu/ str/descriptions/spc_body.html).

[7] Florac, A.W., Carleton A.D., *Measuring the Software Process: Statistical Process Control for Software Process Improvement.* Pearson Education, 1999. ISBN 0-201-60444-2.

[8] Jakolte, P., Saxena, A., "*Optimum Control Limits for Employing Statistical Process Control in Software Process"*. IEEE Transactions on Software Engineering, Vol: 28, No: 12, December 2002, PP 1126-1134.

[9] Romine, J., *"Using Statistical Techniques to Manage Software Projects with Data*", SEPG 2002 Conference, February 18-21, 2002.

[10] Radice, R., *"Statistical Process Control for Software Projects*". 10th Software Engineering Process Group Conference. Chicago, Illinois. March 1998.

[11]Weller, E., "*Practical Applications of Statistical Process Control"*. IEEE Software, May/June 2000, PP 48-55.

[12]Humphrey, W., *Managing the Software Process.* Reading, Mass.: Addison-Wesley Publishing Company, 1989. ISBN 0-201-18095-2.117

[13]Shewhart, W.A., *Statistical Method: From theViewpoint of Quality Control*, Lancaster Press Inc., 1939.

[14]Deming, W.E., *Out of the Crisis*, First MIT Press, 2000. ISBN: 0-262-54115-7.

[15] Fenton, N.E., Pfleeger, S.L., *Software Metrics.* PWS Publishing Company, 1997. ISBN 0-534-95425-1.

[16]Fenton, N.E., Neil M., "*Software Metrics: successes, failures and new directions*". The Journal of Systems and Software, 47, 1999, PP 149-157.

[17]K.U.Sargut, O.Demirors, *Utilization of statistical process control (SPC) in emergent software organizations: Pitfalls and suggestions*, Volume 14, Issue 2 (June 2006) Pages: 135 - 157.

[18]*Abdul Sattar Jamali, Li Jinlin "False Alarm Rates for the Shewhart Control Chart with Interpretation Rules"* Proceedings of the 6th WSEAS International Conference, Hangzhou, China, April 16-18, 2006 (pp238-241).

[19]CHUN-HUI, WU *"An Exploration of the Relationship between Organizational Learning and Software Development Process Maturity"*Proceedings of the 6th WSEAS International Conference on Applied Computer Science, Hangzhou, China, April 15-17, 2007

[20]Chiu-Yao Ting, Chwen-Tzeng Su *"Using Concepts of Control Charts to Establish the Index of Evaluation for Test Quality "* WSEAS transaction on communications, Issue 6, Volume 8, June 2009 PP 535-544.