CONTROL CHARTS FOR IMPROVING THE PROCESS PERFORMANCE OF SOFTWARE DEVELOPMENT LIFE CYCLE

S. Selva Arul Pandian & P. Puthiyayagam
Department of Statistics, Subbalakshmi Lakshmipathy College of Science, Madurai - 625 022, India.

ABSTRACT
In this paper the new ways discussed to cut costs for delivering high-quality product in the today’s global economy. Control charts technique is used to study the process stability and reduce the number of defects in the software product. The Process Performance Model (PPM) is also discussed for the project performance over time among various factors and to predict the range of variation of the outcome that is processes. This work is limited to the software development cycle which follows the Waterfall Model or the Classical life cycle model.

Keywords: Process Performance Model (PPM), Control Chart, Defect density outliers, PPM Analysis.

1. INTRODUCTION TO SPC IN SOFTWARE DEVELOPMENT
Statistical Process Control is an analytical decision making tool which allows to know when a process is in control or not. Variation is inevitable in any process, deciding when the variation is natural and when it needs correction is the key to quality control. The foundation for Statistical Process Control was laid by Dr. Walter A. Shewhart working in the Bell Telephone Laboratories in 1920s conducting research on methods to improve the quality and lower cost. According to Dr. Donald J. Wheeler and David S. Chambers, “SPC is a way of thinking which happens to have some tools attached”. It did have a profound impact within the world of physical product manufacturing, and continues to today. A main concept is that, for any measurable process characteristic, the notion that causes of variation can be separated into two distinct types 1) Common or Chance causes of variation and 2) Assignable or Special causes of variation. SPC allows us to detect the Assignable causes of variation when present. If the assignable causes of variation can be detected, potentially they can be identified, investigated and removed if necessary. That is

Total Variation = Assignable Causes + Chance Causes

In literature, there are several concepts and development on the usage of statistical techniques for software development. A brief review on the recent development by various researchers utilizes SPC techniques for software industry had been summarized. They suggest control charts for both project level process control and organizational level process improvement purposes. SPC technique in the software industry has been growing continuously by extending maturity levels of process improvement models. These models facilitate software companies to implement SPC as a crucial step for achieving process maturity levels.
Humphrey (1989) can be regarded as a reflection of quality management on software engineering discipline. He describes a framework for software process management, outlines the actions to provide higher maturity levels and acts as a basic guide to improve process in a software organization. He emphasizes that measures should be robust, suggest an improvement strategy and be a natural result of the process. He also mentions that it is essential to have a model, but believing it too implicitly can be a mistake.
Kan (1995) provides detailed information about software metrics, software reliability models, and models and analysis of program complexity. He emphasizes the difficulty in achieving process capability in software domain and is cautious about SPC implementation. He mentions that the use of control charts can be helpful for a software organization especially as a supplementary tool to quality engineering models such as defect models and reliability models. He also underlines the necessity of maturity for achieving process stability in software development.
Burr and Owen (1996) describe the statistical techniques currently available for managing and controlling the quality of software during specification, design, production and maintenance. The main focus is given to control charts as beneficial SPC tools and guidelines are provided for measurement, process improvement and process management within software domain.
A similar work is performed by Florac and Carleton (1999). They represent PPM understanding on the utilization of Statistical Process Control for software process improvement. It mostly focuses on the benefits of control charts depending on Shewhart’s principle. It also discusses some issues related to the application of
control charts in software development and incorporates the experience gained from manufacturing industries to software processes.

Jakolte and Saxena (2002) provided idea of 3 sigma control limits and propose a model for the calculation of control limits to minimize the cost of type1 and type2 errors. It is a pioneering one as it questions an accepted practice for control charts and the results of the example studies are encouraging. However, the study is more academic rather than a practical one as it includes too many parameters and assumptions.

Oddur Benediktsson, Darren Dalcher and Helgi Thorbergsson (2006) described an experiment in software developed comparable software products using different development approaches (V-model, incremental, evolutionary and Extreme Programming). Extensive measurements were taken to assess the time, quality, size, and development efficiency of each product. He presented the experimental data collected and its impact on the project and the quality of the results.

Jasmine K.S, and Dr. R. Vasantha (2010) brief the gains in product quality, productivity, cost reduction, cycle time reduction, and even customer satisfaction are offered to corporate decision-makers to justify investment in adoption of the CMMI. He provides an approach and mechanisms for making CMMI investment decisions based on impact on ROI by proposing a new process based capability maturity model for reuse based development process.

James R. Evans, and Rupa Mahanti (2012) Studied Indian software industry and implementing quality management techniques, a better understanding of the implementation of SPC can provide companies with a stronger competitive advantage. Statistical process control (SPC) is a powerful technique for managing, monitoring, analyzing and improving the performance of a process and understanding the critical success factors (CSFs) for successful implementation in the software industry.

2. PROCESS PERFORMANCE MODEL

The CMMI definition of Process Performance Model is “a description of the relationships among attributes of a process and its products that are developed from historical process performance data and calibrated using collected process and product measures from the project and that are used to predict results to be achieved by following a process.” In the early 1990s, the SEI™ Capability Maturity Model® for Software included SPC as an integral component. These concepts remain in today’s Capability Maturity Model Integrated (CMMI®). The following statements from the CMMI for Development, version 1.2 [SEI, 2006] demonstrate the degree of emphasis placed upon SPC concepts:

Under a section titled “Maturity Level 4: Quantitatively Managed”
“Special causes of process variation are identified and, where appropriate, the sources of special causes are corrected to prevent future occurrences.”

Under a section titled “Quantitatively Managed Process”
“Quantitative management is performed on the overall set of processes that produces a product. The sub processes that are significant contributors to overall process performance are statistically managed. For these selected sub processes, detailed measures of process performance are collected and statistically analyzed. Special causes of process variation are identified and, where appropriate, the source of the special cause is addressed to prevent its recurrence.”

Under a section titled “Maturity Level 5: Optimizing”
“At maturity level 5, an organization continually improves its processes based on a quantitative understanding of the common causes of variation inherent in processes.”

Under a section titled “Optimizing Process”
“In a process that is optimized, common causes of process variation are addressed by changing the process in a way that will shift the mean or decrease variation when the process is stabilized. These changes are intended to improve process performance and to achieve the organization’s established process improvement objectives.” To support this concept, the Quantitative Project Management (QPM) process area in the CMMI, Specific Goal #2 exhorts “Statistically Manage Sub process Performance” with Specific Practice 2.2® stating “Apply Statistical Methods to Understand which processes are analyzed, special causes of process variation are identified, and performance is contained within well-defined limits.”

Further, CMMI Generic Practice #4.2 states “Stabilize Sub process Performance” and elaborates with “A stable sub process shows no significant indication of special causes of process variation. Stable sub processes are predictable within the limits established by the natural bounds of the sub process.” Whenever one speaks about the identification of special or assignable causes of variation or the understanding of common or chance causes of variation, he or she is talking, without question, about the use of SPC since SPC is the only known technique capable of such identification/understanding. The CMMI for Development, version 1.2 uses the term “variation” 83 times, the term “special cause” 39 times, and the term “common cause” 19 times.
Clearly, there is a significant amount of emphasis placed upon SPC within the CMMI. So much emphasis, that an organization cannot exceed a CMMI process maturity level rating of 3 without performing SPC. The CMMI doesn’t emphasize any other measurement and analysis technique to this degree. Essentially, isn’t the CMMI placing SPC on a pedestal as the Holy Grail of Measurement and Analysis Techniques?

3. PROBLEM DEFINITION
In the current scenario project manager are in difficult positions to take the appropriate decisions upfront in the projects based many factors such as people, process, product and risk. Metrics across the projects are analyzed to arrive how Defects are injected in the project during the execution. It is difficult to forecast how much defects are introduced in the product and how planning place a crucial role in the reduction of defects injection in the projects. Defects are impacted by the people experience, execution methodology and the planning are briefly described in the case study. The number of defects in a work product is an important measure as it gives us an intuition about how much the customer will be satisfied, how much rework will be performed, how efficient our inspection processes work, which processes need improvement, which components of the system are error prone etc. Therefore, defect counts provide evidence not only on the quality of the product, but also on the quality of the related processes. The terms defect, fault, failure and error are usually used interchangeably.

Defect: a product anomaly. (1) Omissions and imperfections found during early life cycle phases, and (2) faults contained in software sufficiently mature for test or operation.

In order to set the boundaries of this case study, it was determined to get answers for the following

1. Are the processes stable over a period of time?
2. How can SPC be applied to reduce injected defect?

4. CASE STUDY
In order to visualize SPC implementation in real time, a case study was conducted in a software organization. In this study, how SPC can be effectively applied to the processes of a software development project by using its existing measures was investigated. The basis of the case study analysis in this paper is to build a process performance model (PPM) and it is used to understand the variation of factors (People, Process and Product) and to predict the range of variation of the outcomes i.e. processes. The advantage of this model is applying SPC technique for software life cycle process by set it in control when the project meets in-process targets and reaches end product quality goals.

In this paper, the case study is structured in the following order. First up all, the data is to be collected relevant to different metrics from various projects over a period of time. Secondly, the control chart will be used to detect the nonconformities and in visualizing process behavior. Thirdly, the stability of the process performance model (PPM) will be constructed using multiple regressions analysis model for future estimate of project process cycles

\[ y = b_0 + b_1X_1 + b_2X_2 + \ldots + b_kX_k \]

Where Y is the value of the response and Constant \((b_0)\) or regression line intercept is the value of the response variable when the predictor variable\((s)\) is zero. Predictor\((s)\) \((X)\) is the value of the predictor variable\((s)\). Coefficients \((b_1, b_2, b_3)\) represent the estimated change in mean response for each unit change in the predictor value. Fourthly, the process performance model will be validated by box plot and probability plot technique. Fifthly, the data will be studied whether it is capable of producing desired results for the prediction analysis. Finally, the regression will be statistically verified through normality by adopting model sanity test.

5. DATA COLLECTION AND METHODOLOGY
Software process must be assessed based on its capability to develop software that is consistent with user’s requirements. Actually, there is no specific software measure showing the extent to which customer requirements are met. However, there are processes and products that influence production life cycle. The data of all projects found during a review, test, or audit have been collected and tracked through Problem Reports and Document Change Requests. Although the trouble reports evolved within time, the basic defect information such as the subject work product, related project phase, defect priority, initiation and closure date are recorded for all the projects. Considering the following people, product and process parameters across the projects are collected.

In the best possible conditions, a software firm with well-defined processes can be found for a number of projects using the measurement system. Specific metrics are defined and collect data in detail for a long period of
time can be analyzed the data using control charts and evaluated how successful SPC is in detecting process nonconformities and in visualizing process behavior. At the end, the effects of different parameters can be estimated different types of control charts. As a result, it can be understood which metrics represent the desired stable characteristics of the process and which process of SPC is meaningful. The process performance model is considered for the metric of defect density. Where Injected Defect Density is defined as

**Injected Defect Density = Total Internal Defects/Actual Development Effort**

Injected Defect density of Candidate Measures is a relative number of defects in a software product and it is used to measure the defects injected when developing work products. The different measures of metric and its calculation procedure are listed in the following table.

<table>
<thead>
<tr>
<th>Metric/Measurement Description</th>
<th>Calculation formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Injected Defect Density</td>
<td>Injected Defect Density =\frac{\text{Total internal defects}}{\text{Actual development Effort}}</td>
</tr>
<tr>
<td>Average of Developers’ Domain experience</td>
<td>Developers Average domain experience = \frac{\text{Sum of the domain experience of the Developers}}{\text{Total number of the Developers}}</td>
</tr>
<tr>
<td>Average of Developers’ Verification experience</td>
<td>Developers Average verification experience = \frac{\text{Sum of the verification experience of the Developers}}{\text{Total number of the Developers}}</td>
</tr>
<tr>
<td>Staff Turnover ratio</td>
<td>\left(\frac{\text{Total number of resource replacements}}{\text{Total number of resources Assigned}}\right) * 100</td>
</tr>
<tr>
<td>PL - Developer ratio</td>
<td>\left(\frac{\text{No. of PLs}}{\text{Total number of Developers}}\right) * 100</td>
</tr>
<tr>
<td>Fresher’s-Team Member ratio</td>
<td>\left(\frac{\text{No. of Fresher’s(\leq 12 months exp)}}{\text{No. of Team members}}\right) * 100</td>
</tr>
<tr>
<td>Requirement Tool usage</td>
<td>Tools Used (1 or 0)</td>
</tr>
<tr>
<td>Requirement Analysis Effort / Requirement</td>
<td>Requirement Analysis Effort per Req = \frac{\text{Requirement Analysis effort}}{\text{No of requirements}}</td>
</tr>
<tr>
<td>TC Dev Effort / Requirement</td>
<td>TC Dev Effort / Requirement = \frac{\text{Test Coding effort}}{\text{No of requirements}}</td>
</tr>
<tr>
<td>TD Dev Effort / Requirement</td>
<td>TD Dev Effort / Requirement = \frac{\text{Test Design effort}}{\text{No of requirements}}</td>
</tr>
<tr>
<td>TD Review Effort/Requirement</td>
<td>TD Review Effort per Requirement = \frac{\text{Test Design Review Effort}}{\text{No of Requirement}}</td>
</tr>
<tr>
<td>RSI Coding Phase</td>
<td>Requirement Stability index = \left(1 - \frac{\text{Total number of initial baselined requirements}}{\text{No of added + modified + deleted requirements}}\right) * 100</td>
</tr>
<tr>
<td>Project</td>
<td>IDD</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>Project 1</td>
<td>0.04115</td>
</tr>
<tr>
<td>Project 2</td>
<td>0.01532</td>
</tr>
<tr>
<td>Project 3</td>
<td>0.01721</td>
</tr>
<tr>
<td>Project 4</td>
<td>0.03534</td>
</tr>
<tr>
<td>Project 5</td>
<td>0.00983</td>
</tr>
<tr>
<td>Project 6</td>
<td>0.07174</td>
</tr>
<tr>
<td>Project 7</td>
<td>0.07096</td>
</tr>
<tr>
<td>Project 8</td>
<td>0.06847</td>
</tr>
<tr>
<td>Project 9</td>
<td>0.02430</td>
</tr>
<tr>
<td>Project 10</td>
<td>0.01396</td>
</tr>
<tr>
<td>Project 11</td>
<td>0.03419</td>
</tr>
<tr>
<td>Project 12</td>
<td>0.04686</td>
</tr>
<tr>
<td>Project 13</td>
<td>0.08794</td>
</tr>
</tbody>
</table>

Source: Primary Data Nov-2011 to Jan-2012
5.1 CONTROL CHART FOR DATA OUTLIER ANALYSIS

From the collected set of data, the outliers are identified. Outliers are having high variation among the group of the given sample. By carrying out the outlier analysis, it is identified which project is outlier in the sample. Control chart is constructed for all the metric for performing the outlier analysis and the outliers visualized only for defect density.

5.1.1 OUTLIER OF TEST DESIGN DEVELOPMENT PER REQUIREMENT

<table>
<thead>
<tr>
<th>TD Dev Effort / Reqt</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
<th>P7</th>
<th>P8</th>
<th>P9</th>
<th>P10</th>
<th>P11</th>
<th>P12</th>
<th>P13</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.34</td>
<td>0.33</td>
<td>0.32</td>
<td>0.56</td>
<td>0.13</td>
<td>0.52</td>
<td>0.38</td>
<td>0.24</td>
<td>1.28</td>
<td>0.38</td>
<td>0.3</td>
<td>0.11</td>
<td>0.04</td>
</tr>
</tbody>
</table>

In the project 9, Development effort per requirement is high among the projects in that group. Team has taken more effort for design of the requirements. A total requirement provided by the customer is 753 and the development effort put for the requirements is 3600 hrs. In this project there are 7 resources deployed and out of that only 4 resources are available in the peak period.

5.1.2 OUTLIER OF TEST CODE DEVELOPMENT PER REQUIREMENT

<table>
<thead>
<tr>
<th>TC Dev Effort / Reqt</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
<th>P7</th>
<th>P8</th>
<th>P9</th>
<th>P10</th>
<th>P11</th>
<th>P12</th>
<th>P13</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.32</td>
<td>0.2</td>
<td>0.26</td>
<td>0.29</td>
<td>0.27</td>
<td>0.52</td>
<td>0.39</td>
<td>0.89</td>
<td>0.51</td>
<td>0.37</td>
<td>0.29</td>
<td>0.47</td>
<td>0.35</td>
</tr>
</tbody>
</table>

In the project 8, Development effort per requirement is high when compared with the projects in that group. Team has taken more effort for coding of the requirements. A total requirement provided by the customer is 359 and the development effort put for the requirements is 915 hrs spent. In the project only one resource is deployed which performs the activity for both designing and coding.

5.2 STABILITY ANALYSIS

Box plots summarize information about the shape, dispersion, and center of the data. They can also help to spot outliers. If the data are fairly symmetric, the median line will be roughly in the middle of the IQR box and the whiskers will be similar in length. Data are skewed, the median may not fall in the middle of the IQR box, and one whisker will likely be noticeably longer than the other. In the box plot of the precipitation data the median is centered in the IQR box, and the whiskers are the same length. This indicates that except for the outlier (asterisk), the data are symmetric. This is a good indication that the outlier may not be from the same population as the rest of the sample data. In the sample data there are no outliers since the projects of the same type for the analysis are considered. The nature of activity is the same for all the 13 projects and the work product level there are more common between the projects. Hence expected defect density will be the same for almost all the project in those categories. In this analysis, the data will be stable enough to carry out the prediction analysis.
5.3 CAPABILITY ANALYSIS
After analyzing the data for stability whether it is capable of producing desired results for the prediction analysis is studied. A capable process is able to produce products or services that conform to specifications. It is important to make sure the process is in control before you assess capability. The performance of an in-control process is predictable; therefore, the ability of the process to produce units that are “in spec” and predict the number of parts out-of-spec can be assessed. Capability is determined by comparing the process spread to the specification spread. In other words, the width of the process variation is compared to the width of the specification interval. Ultimately it is wanted to see whether the process spread is smaller and contained within the specification spread. Capability indices are ratios of the process spread and specification spread. They are unit less values so that one can use them to compare the capability of different processes. Some capability indices consider the process mean or target. Many practitioners consider 1.33 to be a minimum acceptable value for capability indices; and few believe a value less than 1 is acceptable. In this capability analysis the data shows it is not capable because of process variability such as process, people and product capability for the upper limit (CPL) is 1.00 but the mean value is 0.70 some variable factor influencing the defect density which is skewing the value.

5.4 PROCESSES CAPABILITY ANALYSIS MODEL:
In the first iteration, the processes capability analysis model is fitted by multiple regression equations. By using Minitab software, the regression equations for injected defect density versus Ave Developers’, Ave Developers’ …., are fitted and all regression coefficients are estimated and tested by analysis variance techniques and it is given as
The regression equation is
\[
\text{IDD} = 0.192 - 0.0252 \text{ Ave Developers’ Domain exp}
+ 0.00791 \text{ Ave Developers’ Verifn exp} - 0.000349 \text{ PL - Developer ratio}
+ 0.00075 \text{ Fresher-Team member ratio} - 0.000478 \text{ Staff Turnover ratio}
\]
- 0.0382 Requirement Tool usage - 0.00102 RSI - Coding Phase
+ 0.0513 TD Dev Effort / Reqmt - 0.0272 TC Dev Effort / Reqmt
- 0.102 Reqmt Analysis Effort / Reqmt - 0.353 TD REV Eff/ Reqmt

After fitting it has been tried to eliminate the unimportant variables by adopting the following rules of regressions in the way of iteration procedure.

- R-sq Adj should be > 70%
- All individual p-value should be < 0.05
- Direction given by Co-efficient of correlation should match the logical direction
- All individual VIF should be < 10
- Overall p-value < 0.05
- Normality plot of residuals should have p-value > 0.05
- Scatter plot should not show any specific pattern

From the above test and iterations rules, Average Developers’ Verification experience X is found to be failing in individual P & VIF values as well as in direction of correlation. Therefore it is concluded to remove Average Developers’ Verification experience variable X and recommended to redo the regression. Using above iteration procedure of rules of regression, the project life cycle process is extended to next iteration and the unimportant variables are eliminated. The final iteration result is the outcome of this step by step iteration processes i.e. The regression equation.

$$\text{IDD} = 0.0962 - 0.000382\ PL - 0.0445\ \text{Requirement Tool usage}$$
- 0.503 TD REV Eff/ Reqmt

In the final iteration, the Model R² Adj value, P value and Individual X’s P Value, VIF values found to be satisfying the rules and it has recommended to Perform the sanity test for the model is carried out to test whether the regression equation is under normality. Minitab software results are consolidated the iterations procedure.

### 5.5 Model Sanity test

After the iterations, regression equation is statistically proved through normality and results are validated.
6. CONCLUSIONS
In this study, the problems in software industry are analyzed by using statistical process control with real time scenario. The processes are well defined and established in the long term. Multiple Regression model is constructed for the Injected Defect Density for the different combination of a People, Process and product based on the seven rules. The model will be used at different phases for prediction in the on-going or the future projects. The data reveals that in 3 projects predicted values are within the control limit. In the Project 4, sample values happen to be the outliers as they are above the upper control limit and recommended to increase the PL-Developer ratio above 50% and a developer has to review below 430 requirements a day. In the project 8, Development effort per requirement is high when compared with the projects in that group. Team took more effort for coding of the requirements. In the project there are 7 resources deployed and out of that only 4 resources are available in the peak period. Frequent resources changed in the project which leads to more effort spent for designing the requirements. In the project 9, Development effort per requirement is high among the projects in that group. Team took more effort for design of the requirements. A total requirement provided by the customer is 753 and the development effort put for the requirements is 3600 hrs. Only one resource deployed for the project for Designing and Coding, then the designing may be sent to external experts for reviewing and the rework on the code development will be high. The study can further extended by Bayesian or Markovian model for prediction. The advanced control charts such as CUSUM or EWMA can be used to study the impact.

7. REFERENCES