Optimum Control Limits for Employing Statistical Process Control in Software Process

Pankaj Jalote, *Senior Member*, *IEEE*, and Ashish Saxena

Abstract—There is an increased interest in using control charts for monitoring and improving software processes, particularly quality control processes like reviews and testing. In a control chart, control limits are established for some attributes and, if any point falls outside the limits, it is assumed to be due to some special causes that need to be identified and eliminated. If the control limits are too tight, they may raise too many "false alarms" and, if they are too wide, they may miss some special situations. Optimal control limits will try to minimize the cost of these errors. In this paper, we develop a cost model for employing control charts to software process using which optimum control limits can be determined. Our applications of the model suggest that, for quality control processes like the inspection process, the optimum control limits may be tighter than what is commonly used in manufacturing. We have also implemented this model as a web-service that can be used for determining optimum control limits.

Index Terms—Software metrics, software process improvement (SPI), statistical process control (SPC), control charts, inspections/reviews, software quality control.

1 Introduction

A process is an organization of man, machine, and methods into work activities to produce desired outputs [10]. The outputs produced by a process can be characterized by some quality attributes, the values of which generally show some variation. The causes of variation can be classified as natural causes (also called common causes) or assignable causes (also called special causes). Natural causes are those that are inherent in the process and that are present all the time. Assignable causes are those that occur sometimes and that can be prevented. A process is said to be under statistical control if all the variation in the attributes is caused by natural causes [23], [33].

To keep a process operating under statistical control, it is essential to continuously monitor its performance and identify when it goes out of control. Control charts are common tools that have been used for decades for monitoring manufacturing processes. In a control chart, some quality attribute is chosen and the values of the attribute for samples taken from the production at some time intervals are plotted. Some control limits are established and, if a point falls outside the control limits, an assignable cause is assumed to be present. The selection of control limits determines how frequently "false alarms" will be raised (i.e., a point falls outside the control limits even though there is no assignable cause) and how frequently assignable causes are missed (i.e., an assignable cause is present but the point does not fall outside the control limits). The control limits in manufacturing processes are generally set to 3σ (where σ is the standard deviation) around the mean. These control limits aim to minimize the overall loss due to out of control processes and false alarms.

Though designed for manufacturing processes, SPC concepts can be applied to software process and there is now increased interest in the use of control charts for software processes. Currently, the approach is to use some control chart for some miniprocesses within the overall software process. The process for which control charts are being most commonly used is the inspection/review process [8], [11], [12], [31]. SPC concepts have also been used for testing [3], [4], [19], [31], maintenance [30], [32], personal process [27], and other problems [5], [10]. The use of control charts for software processes is likely to continue to grow, particularly since frameworks like CMM [26] expect some usage of control charts at higher maturity levels.

In software processes, as data points are not that frequent, generally, each data point is individually plotted and evaluated. Hence, charts like the XmR or the U charts are more suitable for software [14], [31], [34] and are the most commonly used charts, as reported in the survey [28]. On the other hand, in manufacturing, the $\bar{X}R$ chart, which employs a sampling based technique, is most commonly used. Consequently, modeling and analysis for selection of control limits for optimal performance has also focused on $\bar{X}$R charts (a survey of some of the models for economic design of control charts is given in [16], [24]). In addition, there are two other differences between manufacturing and software processes that have a bearing on proper design of control charts:

- The primary focus of using control charts in manufacturing is to bring the process back in control by removing assignable causes so that the future production losses are minimized. With software

- *P. Jalote is with the Department of Computer Science and Engineering, Indian Institute of Technology, Kanpur, India - 208016. E-mail: jalote@cse.iitk.ac.in.*
- *A. Saxena is with VERITAS Software, India Pvt Ltd, Symphony, S. No. 210 A/1, Range Hills, Pune India 411020. E-mail: asaxena@veritas.com*

Manuscript received 11 Apr. 2001; revised 7 Jan. 2002; accepted 27 Mar. 2002.
Recommended for acceptance by L.C. Briand.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number 113974.

processes, besides improving the process, an important objective of using control charts is to control the product also. For example, when using control charts for an inspection process, if a point falls outside the control limits, besides the process improvement actions like improving the checklist, inevitably, product improvement actions like rereviews, scheduling extra testing is also taken. In [14] (which is perhaps the first paper on the use of SPC in software), Gardiner and Montgomery suggest "rework" as one of the three actions that management should take if a point falls outside the control limits The use described in [8] clearly shows this aspect of product control. The survey of high maturity organizations also indicates that project managers also use control charts for project-level control [21]. Due to this use for product-control, project managers are more likely to want potential warning signals to be pointed out, rather than miss such signals, even if it means more false alarms.

- The cost parameters that affect the selection of control limits are likely to be quite different in software processes. For example, if a manufacturing process has to be stopped (perhaps because a point falls outside the control limits), the cost of doing so can be quite high. In software, on the other hand, the cost of stopping a process is minimal as elaborate "shutdown" and "startup" activities are not needed. Similarly, the cost of evaluating a point that falls outside the control limits is likely to be very different in software processes as compared to manufacturing processes.

Due to these differences, it is reasonable to assume that, to get the best results, control charts will need to be adapted to take into account the characteristics of the software process. In this paper, we examine the issue of setting control limits when control charts are used in software processes. We will focus our attention on the quality control processes, in particular, the review/inspection process. (The reader is referred to [13], [15] for further information on the inspection process.)

We develop a model for applying control charts to the review process. Using this model, the total cost of process control as a function of control limits is determined. This cost function is then used to numerically compute the optimum control limits at which the total cost is minimized. We are now making the software available as a web service that a process designer can use to determine the optimum control limits by giving the values of the different parameters. By using the model, a process designer can set the control limits such that the overall cost of employing control charts is minimized.

In the next section, we provide an overview of control charts and their use in software process, including an example. In Section 3, we describe our model and the assumptions we make. The overall cost of using a control chart and determining the control limits that will minimize this cost is discussed in Section 4. The section also describes how the model is numerically solved and gives the URL of an experimental software that can be used by process designers to apply this model to their process parameters. Section 5 gives a few examples to illustrate the use of the model. One example uses the data from a real organization and discusses how that data was collected. Sensitivity analysis of the model is discussed in Section 6 and Section 7 contains the conclusion.

2 Statistical Process Control

The basis of statistical process control (SPC) is that, if a process is used consistently, it will demonstrate consistent results in key process attributes like quality, productivity, etc. [5]. Consistency does not mean that the same results will always be achieved—the results will vary as there are some normal variations in the performance that are inherent to all processes. The variation that is inherent in the process is called noise or *common cause variation*. It is not possible to control the variation due to common causes in a process—to reduce the variation further, the process itself has to be changed.

However, there are situations in which special factors are at play when the process is executing. That is, besides the common or inherent causes, there are *special causes*. These special causes, when present, generally cause a large variation in process performance. This change in performance due to special causes can be thought of as the signal through which these special causes can be identified and later removed.

If the variation in performance of a process is only due to common causes, then it is said that the process is under statistical control, or that it is a stable process. The bounds on the performance of such a process can be predicted. On the other hand, if the process is not stable, its performance cannot be predicted as the variation due to special causes is unpredictable.

The key problem of SPC is how to identify the special causes whenever they are present. This means that, from the behavior of the process, the noise has to be separated from the signal whenever there is a signal. Control charts are a means to achieve this goal.

There are various types of control charts. In manufacturing, the most common type of control chart is the $\bar{X}R$ chart [23], [33]. For this chart, at regular time intervals, a sample consisting of a few outputs of the process is taken. For a sample, the mean ($\bar{X}$) is the mean of the value of the attribute of interest for the outputs in this sample. The range R is the difference between the maximum and the minimum value of the attribute for the outputs in the sample. The mean values for the samples collected at different times are plotted on one chart, giving rise to the ($\bar{X}$) chart. The range for the samples is plotted as the R-chart. These charts are used to identify the presence of any special causes.

The control limits are generally set at 3σ around the mean (where σ is the standard deviation). That is, the upper control limit (UCL) is the mean value of $\bar{X}$ from the samples plus 3σ, while the lower control limit (LCL) is mean value of $\bar{X}$ minus 3σ. With these control limits, the chances that a point will fall outside the control limits, even when there is no special cause, is only 0.27 percent [23], [33]. Hence, whenever a point falls outside the control limits, it is highly

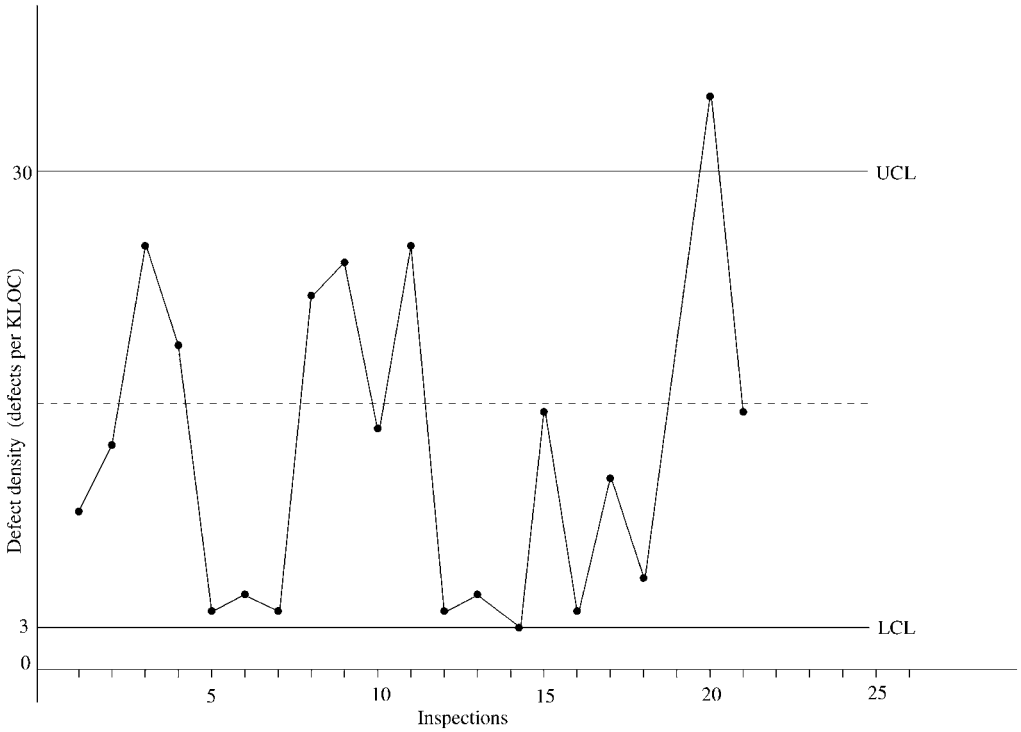

Fig. 1. A control chart for defect rates in inspection.

likely that some assignable cause is present in the process. Therefore, a point falling outside the control limits is taken to signify the presence of of an assignable cause. (Actually, there are other rules also for identifying the presence of an assignable cause [10], [23], [33].)

Note that control charts are only used to identify the presence (with high probability) of special causes. They are silent on what should be done in such a situation.

In the $\bar{X}R$ chart, the mean of the values in a sample is plotted. In software, as outputs are fewer, it is usually desirable to consider and plot the attribute value for each ouptut separately. To achieve this, the XmR chart and the U-chart are well-suited [23], [33]. In the XmR chart, the value of the attribute is plotted individually to form the X-chart. For range, the moving range of two consecutive points is plotted. In the U-chart, the individual data point is plotted, but there is no general control limit—a different control limit is established for each point, depending on the size of the work product (or the "area of opportunity"). However, for a process like the inspection process, approximate U-charts can be built with a single control limit by working with the average size of code that is inspected at a time [8].

Let us illustrate the use of control charts for the inspection process through an example. We will focus on the X-chart. The attributes generally plotted for inspections are the preparation or inspection rate (LOC/hour) or density of defects detected during the inspection (defects/LOC). Suppose we plot defect density. Then, after each inspection, the defect density will be plotted, giving rise to a run chart. From data from previous inspections (either from the same project or from across the organizations), control limits are established. Suppose the control chart is as shown in Fig. 1 (adapted from [8]).

As we can see, the defect density for inspection 20 is more than the upper control limit. For this inspection, other parameters, like coverage rate, preparation rate, previous history of the module being reviewed, etc., are considered. It may be found (as in [12]) that the assignable cause is that new coding standards were introduced and that is why the number of defects is too high. The action following this identification could be to train the programmers in the new coding standards, as was done in [12].

On the other hand, the examination might reveal that all parameters for the process are as expected and the reason the point fell outside the control limits is that the module being inspected is defect prone (as in [8]). In that case, to further control the quality of this module, action might be taken to redesign or rework the module, reinspect it, or schedule more testing [8].

In general, in software processes, if a point falls outside the control limits, actions might be taken to improve the work product and/or to change the process to remove the special cause.

3 System Model and Assumptions

For control charts to be effective, control limits have to be chosen carefully so they separate the normal variation from the variation caused by an assignable cause. The distribution of variation due to natural causes (often assumed to be normally distributed) is, generally, such that there is a nonzero probability for the attribute to have any value. Hence, it is not possible to set control limits that will perfectly and reliably separate the two cases. Regardless of what control limits are set, two types of errors are possible [23], [33]:

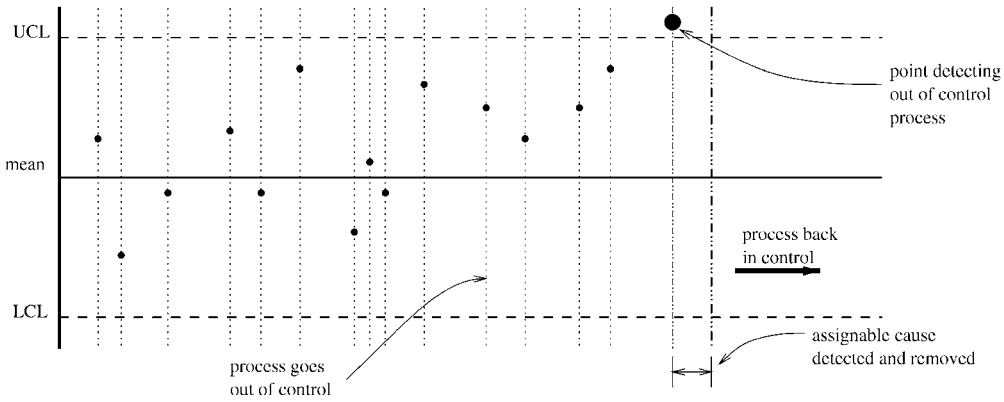

Fig. 2. A control cycle of quality process.

1. Type 1 error: This error occurs when a point falls outside the control limits because of variation due to natural causes themselves. This forces us to search for assignable cause (when none exists) and adds extra cost to the process.
2. Type 2 error: This error occurs when a point falls inside the control limits, although some assignable cause is present in the process. In this case, we ignore this point rather than taking corrective actions and this leads to extra cost in downstream processes.

It is not possible to reduce the probability of these two errors to zero. In fact, there is a tradeoff—if one increases, the other decreases. Hence, setting the control limits is a balancing act. Generally, economic factors regarding the costs of these two types of errors are considered to select the control limits.

For formulating the economic model of a quality control process, we make some assumptions about the process behavior. We assume that the process starts from an incontrol state, i.e., initially, all the variations in the process performance are due to natural causes. We assume that the main attribute being monitored is observed defect density (ODD), which is the number of defects detected per unit size (assume that size is measured in KLOC). For defect density, U-charts are more suitable as they can easily accommodate the different sizes of document/code being reviewed. However, as suggested in [10], as control limits often do not vary much for different data points, XmR charts are a reasonable and simpler alternative. Here, we consider the X-chart of ODD.

We assume that ODD is distributed normally with mean μ_0 defects per KLOC and standard deviation σ. That is, the average number of defects detected by this process is μ_0 per KLOC and the actual defect density of a review is normally distributed with the mean μ_0 and standard deviation σ.

The process is considered as a series of control cycles. Each cycle begins with the quality process in an in-control state. After operating for some time in the in-control state, an assignable cause occurs and the process goes out of control. We assume that, when out of control, the mean of ODD shifts by $\pm\delta\sigma$, but the variance remains the same. For some time, the presence of this assignable cause goes undetected as the data points continue to fall within the control limits. Eventually, a point falls outside the control limits, generating a signal. On this signal, analysis is done to identify the cause and actions are taken to "repair" the process. Following a repair, the process returns to the in-control state and a new cycle begins. This behavior of the process is shown in Fig. 2.

We can also view this whole process as a failure-repair process [29]. The process starts with an in-control state. In this state, if a data point falls outside the limits, some analysis is done, but the process continues to remain in the in-control state. At some point, some assignable cause occurs and the process goes in out-of-control state. It continues in this state until a data point falls outside the control limits that have been set for the process. On this signal, some analysis is done. As the process has some assignable cause, we assume that the analysis will reveal the presence of the assignable cause and the process will be repaired by removing the assignable cause. Once the assignable cause is removed, the process goes back to the in-control state. A state diagram of a control cycle is shown in Fig. 3. One control cycle is: Starting from the initial in-control state, it remains in this state for some time, goes to the out-of-control state, and remains in it until a point falls outside the control limits, then returns back to an in-control state after process and product repair. A control cycle is thus an interval between two successive repairs. An optimization over a single control cycle will causes the optimization of the whole quality process since the process is a repetition of these control cycles.

A cycle in the process life consists of three different stages. These stages are: process operating in an in-control state (before arrival of assignable cause), process operating

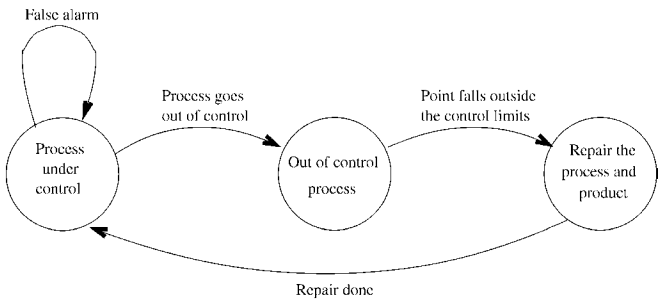

Fig. 3. State diagram of a control cycle.

in an out-of-control state (between the arrival and detection of assignable cause), and searching and repairing of the assignable cause. All three stages add some cost to the control cycle. (We do not model examining a false alarm as a separate state as the cost of false alarms is taken as part of the in-control state.) The cost in the in-control stage is due to a Type 1 error that may occur and the cost in an out-of-control stage is due to Type 2 errors. In the third stage, the cost is due to process repair.

We assume that the average cost of analyzing a false alarm is T and the average cost of correcting an out-of-control situation is W. We assume that the process stays in an in-control state for an average of λ reviews. That is, on average, an assignable cause occurs after every λ controlled reviews.

To compute the cost implications of not detecting an out-of-control situation, we need to understand what the cost is of removing a defect in the current stage versus what it will cost to remove it in later stages. We assume that the mean cost of removing a defect in this stage is C_1 and the mean cost of removing it later is C_2. To compute the total cost, we assume that the average size of the work product is η KLOC. To summarize, we use the following parameters:

1. Control limit parameter k (the actual control limits are $\mu_0 \pm k\sigma$).
2. When the process is operating normally, the average ODD is μ_0 and its standard deviation is σ.
3. Amount of shift, δ, in ODD due to occurrence of an assignable cause (the shift is $\delta\sigma$).
4. Average number of reviews done in an in-control state, λ, before the process goes out of control.
5. Average cost of analyzing false alarm, T.
6. Average cost of correcting an out of control situation, W.
7. Average cost of fixing a defect in this phase, C_1.
8. Average cost of fixing a defect in later stages, C_2.
9. Average size of a work product being reviewed, η.

4 Setting Control Limits

The total cost of a cycle is sum of cost of all three stages. The contribution of these costs to total cost depends on the probability of their occurrences of corresponding errors which in-turn depend on control limits. The probability of a Type 2 error increases with control limit and probability of a Type 1 error decrease and vice versa. The sum of these costs forms a U-shaped curve, which has a minimum at some value of control limits. This cost minimizing value of control limits economically balances the two costs and minimizes the total cost.

Cost minimization is perhaps the most important criteria while setting the control limits. Several methods for the design of economically optimal control charts for manufacturing processes have been suggested in the quality control literature. A detailed study of these models, most of which focus on $\bar{X}$R charts, is given in [7], [9], [25]. Here, we present a simple cost model which is used to determine the optimal control limits in software processes.

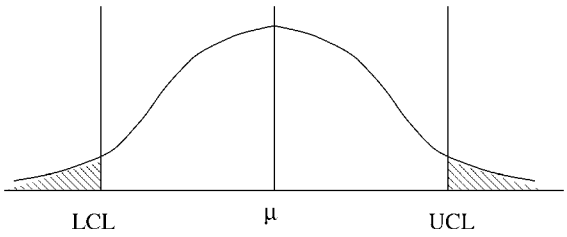

Fig. 4. False alarm.

4.1 False Alarm Cost

When the process operates in an in-control state, with each observed data point, there is an associated probability that it may fall outside the control limits. The points which fall outside the control limits, even when the process is actually in an in-control state, are called *false alarms* or *false positives*. On occurrence of a false alarm, the process is examined to check whether the process is in the under-control state or not. This extra examination is the False Alarm Cost (FAC).

If the control limits are set at $\pm k\sigma$ distance from the mean μ_0, the probability that ODD of a quality process exceeds the control limits, even though the process in control is essentially the area in the probability density function from ∞ to LCL plus the area from UCL to ∞. (A parameter-like defect density does not have negative values, leading to a truncated distribution. We assume that errors due to this truncation on the probabilities are negligible.) As the function is symmetric, this probability, α is given by the following equation:

$$\alpha = 2\Phi(-k), \tag{1}$$

where Φ is the cumulative distribution function of a standard normal variable and is given by

$$\Phi(z) = \int_{-\infty}^{z} \frac{\exp^{-x^2/2}}{\sqrt{2\pi}} dx. \tag{2}$$

Pictorially, α can be represented as area under parts of the normal distribution curve, as shown in Fig. 4. It can be shown that, if control limits are set to 3σ, then α is 0.0027; if the limits are 2σ, then α is 0.0455; and, if the limits are 1σ, then α is 0.3173.

As the process remains in an in-control state for an average of λ reviews, and the expected cost of handling a false alarm is T person hours, the expected number of reviews done in the in-control state is $T \times \lambda$. Hence, the FAC in one control cycle is

$$\text{FAC} = \alpha \times T \times \lambda \text{ person hours}. \tag{3}$$

Note that, for computing FAC, we do not need to consider the full length of a cycle, but only that part of the cycle during which the process is operating normally. After the shift takes place, the cost incurred till it is detected is considered next. Here, we considered that the average duration of the process staying in control is λ reviews. From a modeling perspective, it is perhaps better to consider "going out of control" as a Poisson process with rate. However, we have found, in our experiments, that, by doing this, the final results do not change significantly. Hence, we work only with the average.

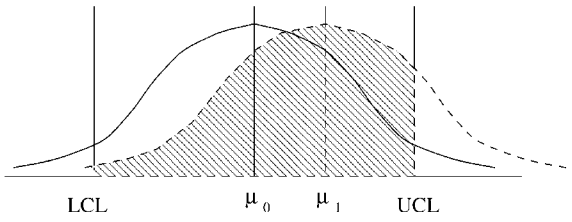

Fig. 5. Type II error.

4.2 Undetected Failure Cost

The reviews done between the occurrence and detection of an assignable cause are the ones in which the process is not operating normally. Due to the process being out of control, these reviews may fail to detect the expected number of defects from the work product. These defects get transferred to later phases of development where the cost of removal is higher. In general, the later defects are detected, the higher the cost of fixing them. If the average cost of fixing a defect in current development phase is C_1 person hours and the expected cost of fixing a defect after the current phase is C_2 person hours, then the additional cost per such review due to this error is

$$M = \delta \times \sigma \times (C_2 - C_1) \times \eta \text{ person hours}, \tag{4}$$

where $\delta \times \sigma$ is the shift in ODD when the process is out of control and η is the average size of work product being reviewed in KLOC ($\delta \times \sigma \times \eta$ is the expected number of defects that pass through this stage).

To get the total cost in a cycle, we need to determine the expected number of reviews conducted while the process is in an undetected out-of-control state. Let P be the probability that, when an assignable cause is present, the ODD will fall outside the control limits. P represents the ability of the control chart to detect an assignable cause and is $(1 - \text{Probability of Type II error})$. The probability of a type II error represents the area in the shifted curve that falls within the control limits, as shown in Fig. 5. Mathematically,

$$P = \Phi(-\delta - k) + \Phi(\delta - k). \tag{5}$$

If the shift is known, then P can be computed for a control limit. For example, if the shift is 1σ, then P is 0.022 with control limits set to 3σ, 0.16 with control limits set to 2σ, and 0.52 with control limits set to 1σ. Similarly, if the shift is 2σ, then P is 0.158 with control limits set to 3σ, 0.50 with control limits set to 2σ, and 0.842 with control limits set to 1σ.

The expected number of reviews that take place with the shifted process (i.e., after the shift occurs but before it is detected) is $1/P$. Therefore, the expected undetected failure cost (UFC) for a cycle is

$$UFC = M/P \text{ person hours}. \tag{6}$$

4.3 Repair Cost

When an assignable cause is found, certain actions are taken to remove the assignable cause and bring the process back into control. We assume that this cost is fixed at W person hour. As there is only one repair in each cycle, the total repair cost (RC) is W.

4.4 Optimal Control Limits

Total cost (TC) of a control cycle is the sum of the false alarm cost, undetected failure cost, and repair cost,

$$\begin{aligned} \text{Total Cost}, TC &= FAC + UFC + RC \\ &= \alpha T\lambda + M/P + W \\ &= 2T\lambda\, \Phi(-k) + \frac{M}{\Phi(-\delta - k) + \Phi(\delta - k)} \\ &\quad + W \text{ person hours}. \end{aligned} \tag{7}$$

It is clear that the total cost depends on many parameters and, if the value of the parameters are known, we can find the cost of a cycle for a given control limits. The dependence of the total cost on various parameters is shown in Fig. 6.

With the function for total cost known, we now address the question of minimizing the cost. It should be clear that the control limit parameter, k, is the one that influences the cost the most. Furthermore, when using a process, k is a parameter that is fully under the control of engineers and they decide what it should be, unlike most of the other parameters that are the properties of the process. For an engineer or a process designer, then, the main question is what value of k should be selected. Given the cost function, the obvious answer is to select k that will minimize the total cost. The value of k that achieves this is the optimal control limit k_{opt}.

It is hard to analytically differentiate the cost with respect to k and then determine the value of k_{opt}. We therefore do it numerically. We have written a program that, given the value of parameters, computes the cost for different values of k and plots it. Besides the plot, it also gives the value of k_{opt}.

In one version of this implementation (available as a service at www.cse.iitk.ac.in/research/software), to simplify the use of the model, for each k, we compute the cost for different values of shift (between 0.5 and 3.0 times the standard deviation) and then take the final cost as the average cost. This cost is then used to determine the optimum.

5 Examples

Let us now illustrate determining the optimal control limits through some examples. First, let us take the data given in [10] for a code review process. The average size of code during review is 0.32 KLOC and the review process detects on average 20.2 defects/KLOC with standard deviation 7.2. We assume that the process shifts on an average after every 40 reviews. That is, the code review process remains stable for on an average 40 reviews before some assignable cause occurs. We assume that the cost of investigating a false alarm is 10 person hours and for finding and repairing a process is an additional 10 person hours. This assumption says that, if the performance of a code review falls outside the control limits, even when the review process is operating normally, it takes 10 person hours to examine all the data for that review and declare that there is no assignable cause. And, if the analysis shows that there is an assignable cause, the activities that need to be undertaken to modify the review process are an additional 10 person-hours. We

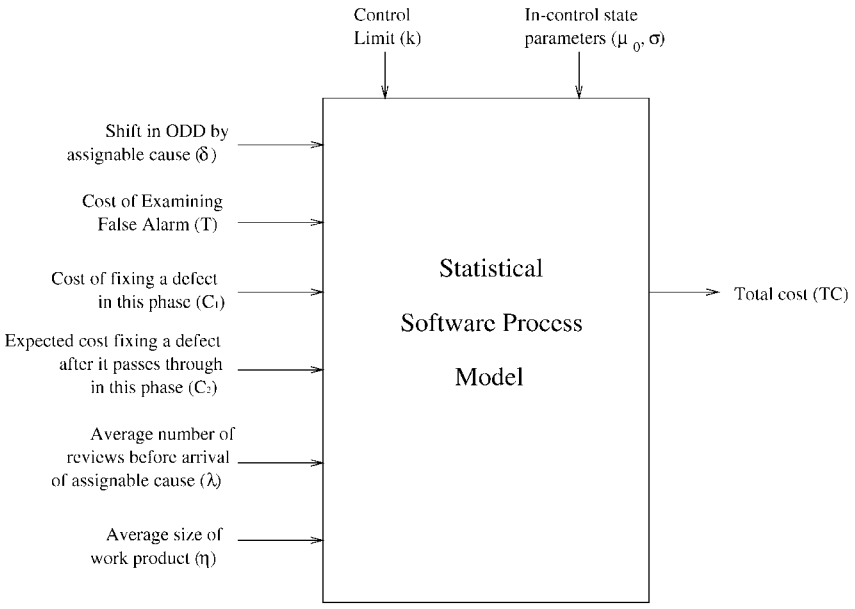

Fig. 6. Input and output parameters of our model.

further assume that the average cost of fixing a defect in the current phase is 15 person hours, while the expected cost of fixing it after the current phase is 50 person hours (these costs are based on data given in [1], [2]).

Now, we compute the cost of one control cycle at different control limits [$0.1 \leq k \leq 3.0$, steps of 0.05] and draw a graph between cost and value of k. Fig. 7 shows the graphs for the different cost components and the total cost when the process mean shift is 2σ.

It is clear from the figure that the optimum value of k is 1.45. (The optimum value of k is 1.10, 1.15, and 1.3 for shifts of 0.5σ, 1.0σ, and 1.5σ, respectively.) In other words, the optimal control limits for this process are $(20.2 \pm 1.45 \times 7.2)$ defects/KLOC. It is also worth noting that, if the control limits are set at 3σ, which is the usual chosen number for manufacturing, the total cost is about 3.5 times the cost at optimal control limit. Clearly, choosing control limits without considering their impact on cost can have a significant impact on cost.

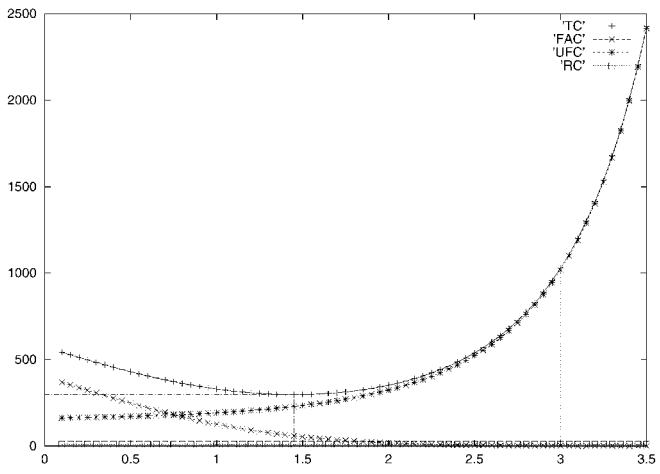

Fig. 7. Cost as a function of k.

Let us now take the example of the code review process from a real organization. Infosys is a high maturity organization that heavily uses metrics for managing its processes and projects. It uses some tools to record effort and defect data. For most activity, it collects the activity effort separately from the rework effort. The methods for effort data collection and the data collected in reviews are discussed in [19]. The data used here is from a set of projects of one of the business units.

The average size of the code being reviewed, the average defect density, and the standard deviation are easily obtained from the review data of these projects. These parameters are 0.9 KLOC, 9.2 defects/KLOC, and 7, respectively. The average number of reviews for which the process remains in control was estimated by discussing with project managers how frequently they find the review data to be too unexpected to warrant their attention and further analysis. This was estimated to be about 10 reviews.

The average cost of fixing a defect during code reviews is easily obtained from the rework effort after review and the number of defects detected in reviews. This cost is 3.9 person-hours. The average cost of fixing a defect after code review can be computed as the average cost for finding a defect in unit testing, system testing, or acceptance testing. This cost is approximately 4.8 person-hours.

Estimating the cost of false alarm and cost of bringing a process back in control required more thought. In Infosys, projects employ a structured defect prevention process [20]. This requires analysis of defect data, followed by a root cause analysis, followed by implementing actions to prevent the defects from occurring again. We estimated that the cost of bringing the review process back in control, which will involve analysis of defects and then implementation of the preventive actions, will be similar to the cost of doing defect prevention as similar tasks with similar objectives are performed in defect prevention. From the defect prevention data, this came to about 5.2 person-hours. We further estimated that the cost of a false alarm will be

about 75 percent of this cost (as it will involve only analysis but no further action), which came to 3.9 person-hours.

For this data, the optimum control limits come to be 1.45 times the standard deviation. That is, the upper control limit should be set to 19.5 defects per KLOC and the lower control limit should be set to 0 KLOC. (The optimum value of k is 1.20, 1.25, 1.40, and 1.55 for shifts of 0.5σ, 1.0σ, 1.5σ, and 2.0σ, respectively.)

We have tried various variations of published data and the data obtained from Infosys as examples. Those examples indicate that the optimum limits value k increases with increase in shift δ, false alarm cost T, and number of reviews in control state λ, and decreases with undetected failure cost M. We also saw that, in these examples, the optimum control limits were consistently lower than the frequently used 3σ value and the cost at 3σ limits is much higher than the optimum cost. The examples also suggest that, for software, the 3σ limits are suitable if the shift due to assignable cause is large (in the examples, more than 5.5σ), if the average number of inspections before the process goes out of control is large (in our examples, above 15,000), if the cost of examining a false alarm is high (above 2,500 person-hours in our examples), or if the cost of missing an out-of-control review is low (below 0.48 person-hours in our examples). These suggest that 3σ may not be optimum for software processes.

It should, however, be clear that having a tighter control limit implies that, even though fewer process shifts will pass by, there will be more false alarms. The excessive false alarms may have a detrimental effect on the people or the process beyond the direct cost of analyzing these signals. If this is the situation, then the cost of false alarms should be suitably enhanced to accommodate the "loss" due to these factors. Optimal control limits can then be established for these new costs.

6 Sensitivity Analysis

Sensitivity analysis in a model is the study of the variation of its output with respect to its inputs. It is generally used to determine which input parameters have a significant effect on the output and their relative ranking.

In our model, there are five main factors that affect the selection of optimal control limits. In many situations, the value of these parameters will have to be estimated and may not be accurately known. The sensitivity analysis will help a process designer in understanding which parameters need to be estimated more accurately and the impact of inaccuracies on the outcome.

To perform the sensitivity analysis on our model, we used the statistical approach suggested in [17], [18]. In this approach, first, a set of values for the input parameters are generated using hypercube sampling as follows: Each of the input parameters is divided into some nonoverlapping intervals. One value for each interval is selected randomly. The values generated for the different variables are then combined randomly to generate the input vectors. For the generated input vectors, the output for the model is then computed.

For ranking the input parameters, a stepwise regression is performed on this set of input vectors and the corresponding outputs. In stepwise regression, first, the variable having the highest simple correlation with the output is selected as having the highest rank (i.e., a rank of 1). Then, of the remaining variables, the one having the highest partial correlation (i.e., the variable that has the highest correlation after adjusting for the variables that have already been selected) is selected at the next rank. This process continues till all variables have been added.

TABLE 1
Input Parameter Ranking by Sensitivity Analysis

Rank	Process Attribute
1	Shift in ODD by assignable cause, δ
2	Undetected failure cost, M
3	False alarm cost, T
4	Number of reviews in control state, λ

The ranking of the main input variables in our model is shown in Table 1.

This ranking says that the optimal cost varies more by a change in M and δ than by changes in λ and T. In other words, an error in estimating the values of undetected failure cost or shift will cause a larger deviation in the cost and, thus, is more harmful than error in other parameters. So, we have to be more careful about these two parameters when using this model for setting control limits. Note that the fifth factor W does not appear here as it is constant with respect to k.

7 Conclusion

This paper discusses the use of statistical process control and control charts to software quality control processes like reviews and testing. As the use of control charts in software has begun in recent years, standard charting techniques from manufacturing are used. We argue that, for applying control charts, software processes differ from manufacturing processes since the focus is also on project level control and the cost parameters are different. Due to these, the 3σ control limits that are frequently used in manufacturing can become unsuitable for software.

To find the optimum control limits, a cost model of software quality process is presented. The model estimates the cost of a failure-repair cycle of quality process using some process parameters like average cost of examining a false alarm, average cost of detecting and fixing a defect, expected cost of detecting and fixing a defect if it passes through the current phase, applied control limits, etc. Then, for a given set of process parameters, we compute the control limits at which the total cost of process is minimum.

We have applied the model to some published data and data on code reviews from one high-maturity software organization. These examples suggest that using 3σ may not be suitable for software processes—in software processes, tighter control limits are perhaps better.

During this analysis, we have made some simplifying assumptions, like defect density for reviews having a normal distribution, and have considered only the basic

rule of a point falling outside the control limits for detecting an out-of-control situation. We have not considered the impact of supplementary rules for detecting an out-of-control situation, which can make control charts more sensitive to process shifts [6].

We are currently web-enabling our analysis software, such that any process designer can input the parameters for a process and get the cost curve and the optimum control limits. This facility will be publicly available shortly. A trial version is available at www.cse.iitk.ac.in/research/software.

ACKNOWLEDGMENTS

The authors would like to thank Professor Ashok K. Mittal of the Industrial and Management Engineering Department at IIT Kanpur for helping us with various aspects of SPC and its modeling. They also extend a special thanks to Infosys for agreeing to provided the data on the review process and to Anoop Kumar for collecting and summarizing the data from projects. Finally, they thank the referees for giving detailed feedback that considerably improved the paper.

REFERENCES

[1] V.R. Basili and D.M. Weiss, "Evaluation of a Software Requirements Document by Analysis of Change Data," *IEEE Trans. Software Eng.,* vol. 10, no. 6, pp. 728-738, 1981.
[2] B. Boehm, *Software Engineering Economics.* Prentice Hall, 1981.
[3] D. Card and R.A. Berg, "An Industrial Engineering Approach to Software Development," *J. Systems and Software,* vol. 10, pp. 159-168, 1989.
[4] D. Card and R.L. Glass, *Measuring Software Design Quality.* Prentice Hall, 1990.
[5] D. Card, "Statistical Process Control for Software," *IEEE Software,* pp. 95-97, May 1994.
[6] C.W. Champ and W.H. Woodall, "Exact Results for Shewhart Control Charts with Supplimentary Run Rules," *Technometrics,* vol. 29, no. 4, pp. 393-399, 1987.
[7] W.K. Chiu and G.B. Wetherill, "A Simplified Scheme for the Economic Design of $\bar{X}$ Charts," *J. Quality Technology,* vol. 6, no. 2, pp. 63-69, Apr. 1974.
[8] R.G. Ebenau, "Predictive Quality Control with Software Inspections," *Crosstalk,* June 1994.
[9] E. von Collani, "A Simple Procedure to Determine the Economic Design of an $\bar{X}$ Control Chart," *J. Quality Technology,* vol. 18, no. 3, pp. 145-151, July 1986.
[10] W.A. Florac and A.D. Carleton, *Measuring the Software Process: Statistical Process Control for Software Process Improvement.* Addison-Wesley, 1999.
[11] W.A. Florac, A.D. Carleton, and J.R. Bernard, "Statistical Process Control: Analyzing a Space Shuttle Onboard Software Process," *IEEE Software,* pp. 97-106, July/Aug. 2000.
[12] A. Florence, "CMM Level 4 Quantitative Analysis and Defect Prevention," *Crosstalk,* Feb. 2001.
[13] D.P. Freedman and G.M. Weinberg, *Handbook of Walkthroughs, Inspections, and Technical Reviews—Evaluating Programs, Projects, and Products.* Dorset House, 1990.
[14] J.S. Gardiner and D.C. Montgomery, "Using Statistical Control Charts for Software Quality Control," *Quality and Reliability Eng. Int'l,* vol. 3, pp. 40-43, 1987.
[15] T. Gilb and D. Graham, *Software Inspections.* Addison-Wesley, 1993.
[16] C. Ho and K.E. Case, "Economic Design of Control Charts: A Literature Review for 1981-1991," *J. Quality Technology,* vol. 26, no. 1, pp. 39-53, Jan. 1994.
[17] R.L. Iman, J.C. Helton, and J.E. Campbell, "An Approach to Sensitivity Analysis of Computer Models: Part I—Introduction, Input Variable Selection and Preliminary Variable Assessment," *J. Quality Technology,* vol. 13, no. 3, pp. 174-183, July 1981.
[18] R.L. Iman, J.C. Helton, and J.E. Campbell, "An Approach to Sensitivity Analysis of Computer Models: Part II—Ranking of Input Variable, Response Surface Validation, Distribution Effect and Technique Synopsis," *J. Quality Technology,* vol. 13, no. 4, pp. 232-240, Oct. 1981.
[19] P. Jalote, *CMM in Practice : Processes for Executing Software Projects at Infosys.* Addison-Wesley, 1999.
[20] P. Jalote, *Software Project Management in Practice.* Addison-Wesley, 2002.
[21] P. Jalote, "Use of Metrics in High Maturity Organizations" *Proc. Software Eng. Process Group Conf. (SEPG '00),* Mar. 2002.
[22] P. Jalote, S. Raghavan, M.R. Bhashyam, K. Dinesh, and S. Ramakrishna, "Quantitative Quality Management through Defect Prediction and Statistical Process Control" *Proc. Second World Quality Congress for Software,* Sept. 2000.
[23] D.C. Montgomery, *Introduction to Statistical Quality Control,* third ed. John Wiley & Sons, 1996.
[24] D.C. Montgomery, "The Economic Design of Control Charts: A Review and Literature Survey," *J. Quality Technology,* vol. 12, no. 2, pp. 75-87, Apr. 1980.
[25] D.C. Montgomery, "Economic Design of an ($\bar{X}$) Control Charts," *J. Quality Technology,* vol. 14, no. 1, pp. 40-43, 1982.
[26] Software Engineering Institute, *The Capability Maturity Model for Software: Guidelines for Improving the Software Process,* M.C. Paulk et al., principal contributors and eds., Addison-Wesley, 1995.
[27] M.C. Paulk, "Applying SPC to the Personal Software Process," *Proc. 10 Int'l Conf. Software Quality,* Oct. 2000.
[28] R. Radice, "Statistical Process Control in Level 4 and 5 Organizations Worldwide," *Proc. 12th Ann. Software Technology Conf.,* 2000, also available at www.stt.com.
[29] K.S. Trivedi, *Probability & Statistics with Reliability, Queuing, and Compute Science Applications.* Prentice-Hall, 1982.
[30] E. Weller, "Applying Statistical Process Control to Software Maintenance," *Proc. Applications of Software Measurement,* 1995.
[31] E.F. Weller, "Practical Applications of Statistical Process Control," *IEEE Software,* pp. 48-54, May/June 2000.
[32] E. Weller, "Applying Quantitative Methods to Software Maintenance," *ASQ Software Quality Professional,* vol. 3, no. 1, Dec. 2000, also available from www.stt.com.
[33] D.J. Wheeler and D.S. Chambers, *Understanding Statistical Process Control,* second ed. Knoxville, Tenn.: SPC Press, 1992.
[34] R.E. Zultner, "What Do Our Metrics Mean?" *Cutter IT J.,* vol. 12, no. 4, pp. 11-19, Apr. 1999.

Pankaj Jalote received the BTech degree from IIT Kanpur and the PhD degree in computer science from the University of Illinois at Urbana-Champaign. He is a professor and head of the Department of Computer Science and Engineering at the Indian Institute of Technology, Kanpur. He was formerly an assistant professor in the Department of Computer Science at the University of Maryland, College Park, where he also had a joint appointment with the Institute of Advanced Computer Studies. From 1996 to 1998, he was the vice president (quality) at Infosys Technologies Ltd., a large Bangalore-based company providing software solutions worldwide, where he spearheaded Infosys' move to high maturity levels of the CMM. He is the author of *CMM in Practice—Processes for Executing Software Projects at Infosys* (Addison Wesley) and of the recent book *Software Project Management in Practice* (Addison Wesley 2002). He has previously authored two other books—*An Integrated Approach to Software Engineering* (Springer Verlag) and *Fault Tolerance in Distributed Systems* (Prentice Hall) and about 60 research articles. His area of interest is software engineering and distributed computing. He is a senior member of the IEEE.

Ashish Saxena received the bachelors degree in computer science and engineering from Barkatullah University, Bhopal, in 1999, and the MTech degree in computer science and engineering from the Indian Institute of Technology, Kanpur, in 2001. He is currently an associate software engineer at Veritas Software India Pvt Ltd, Pune. His areas of interest are operating systems, software process and software quality, and algorithms.