Presentation
Paper
Bio
Return to Main Menu

PRESENTATION

T12

Thursday, February 15, 2001
11:00AM

Statistical Process Control (SPC) for Software Inspections

Don Porter
Motorola

International Conference On
Software Management & Applications of Software Measurement
February 12-16, 2001
San Diego, CA, USA

Statistical Process Control for Software Inspections

Don Porter
Senior Statistician
Motorola

Statistical Process Control for Software Inspections

Cost of Poor Quality

- Software defects that escape the creation phase can be very costly. These costs include:
  - **Rework** (**$8,000 to $40,000 ***)
  - **Overhead Expenses**
  - **Lost Customers**
  - **Job Satisfaction** (Fixing Problems, Late Nights)
- A Software Organization can Lose Millions of on Each Release

We cannot afford to miss defects!!!

* According to Barry W. Boehm, ("Software Engineering Economics", Prentice Hall, October 1981)

Statistical Process Control for Software Inspections

F Control Charts E

- **The time tested effective tool to catch defects. Keep track of:**
  - **Preparation Effort = Prep Time/Amount**
  - **Inspection Effort = Inspection Time/Amount**
  - **Defect Density = (Major + Minor)/Amount**
- **NOTE: Effort variables are the inverse of traditional preparation and inspection rates.**

Statistical Process Control for Software Inspections

- But, traditional control charts cannot be effective for inspections!
- They generate control limits by adding and subtracting 3 standard deviations from the mean.
  - – Lower Control Limit = m - 3 * s
  - – Center Line = m
  - – Upper Control Limit = m + 3 * s

So ... why won’t this work for inspections?

Inspection Control Charts - First Attempts

- First control charts used statistically sound approaches. Unfortunately, managers ignored them because they:
  - possessed variable control limits, or
  - variables lacked recognizable reference distributions, or
  - were designed for normally distributed (bell-shaped) measures

Statistical Process Control for Software Inspections

Example - Charting Faults

- Faults obey a Poisson probability law:
  - **constant probability of discovering faults over lines in a code module, say:**

    l = 0.1
  - **Mean faults (the Poisson parameter) for a code of size S (say S=50 LOC) equals:**

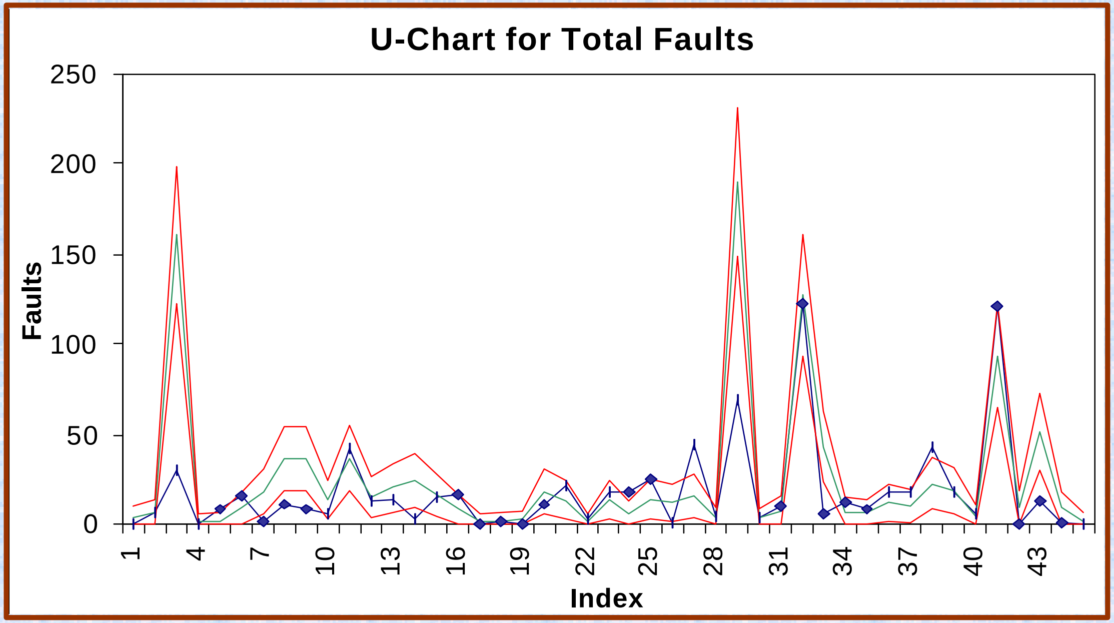
    m = l * S = 0.1 * 50 = 5
  - **Standard deviation =square root of mean:**

    s = Öm = Ö5 = 2.24

NOTE: Mean and standard deviation change for different sizes of code modules!

Statistical Process Control for Software Inspections

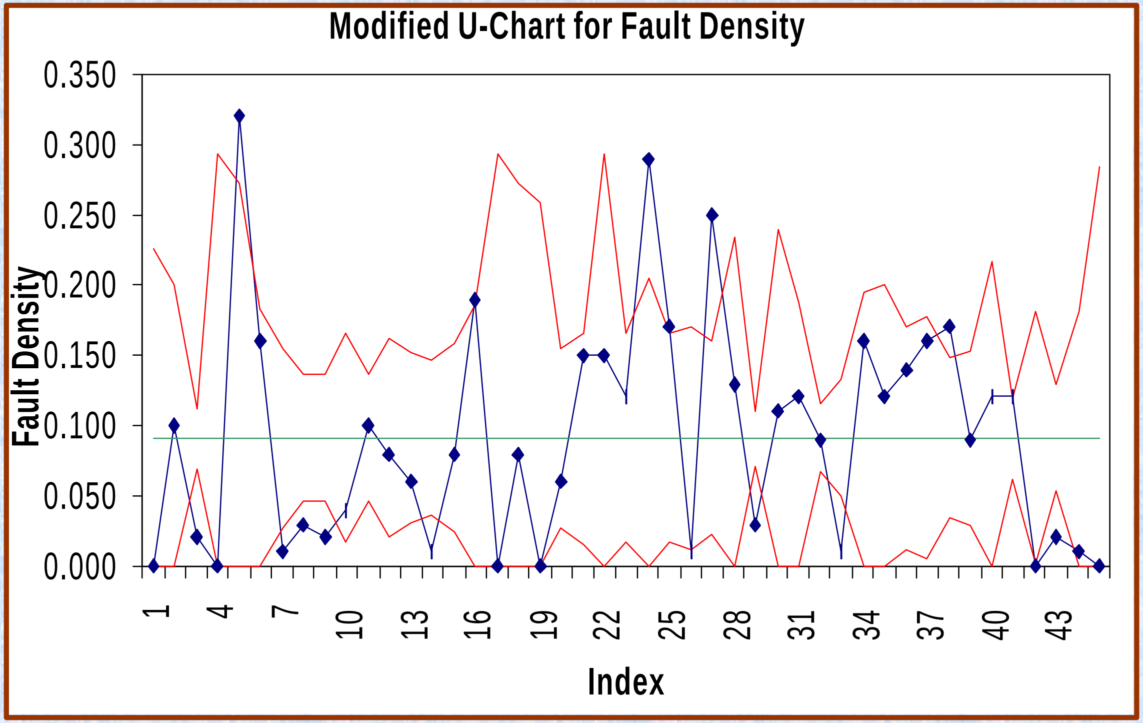
U-Charts Have Variable Centerlines and Control Limits - Too Ugly!



Control limits depend on the module size, so it is difficult to even see the centerline (green) and control limits (red). It's impossible to determine if there are any time trends.

Statistical Process Control for Software Inspections

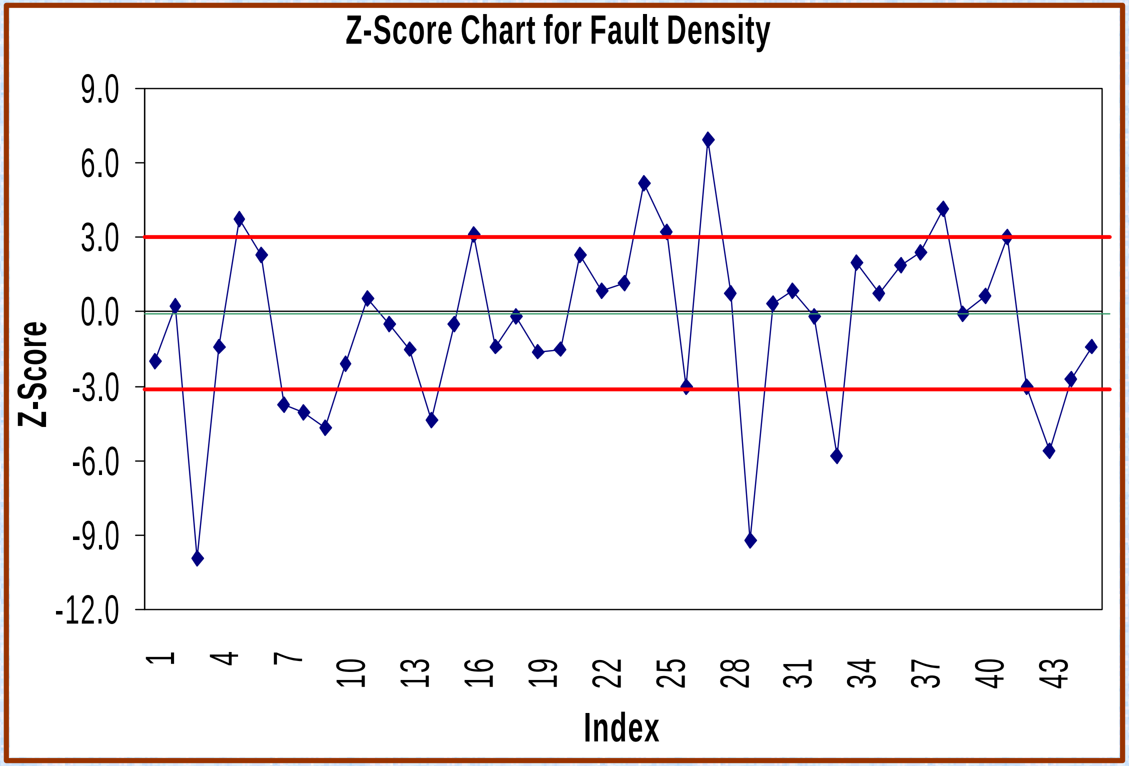
Modified U-Charts For Fault Density Has Variable Control Limits - Still Too Ugly!



This chart at least has a constant centerline. But variable control limits are still difficult for the user to interpret. And it is still very difficult to assess trends.

Statistical Process Control for Software Inspections

Z-Charts - Inappropriate for Fault Density



This chart has constant centerline and limits, but is designed for a symmetric variable. Poisson variables are decidedly NOT symmetric. Far too many “out of control” signals.

Statistical Process Control for Software Inspections

Four steps to building control charts

- 1. Find stable data to characterize the process when it is in control
- 2. Identify the distribution type (Normal, Poisson, etc.)
- 3. Establish the central value
- 4. Determine control limits based on cost considerations

Statistical Process Control for Software Inspections

Distribution Fitting: Preparation Effort

Mean ± k*Sigma Just Doesn't Add Up!

Table 1. Control Variable Statistics

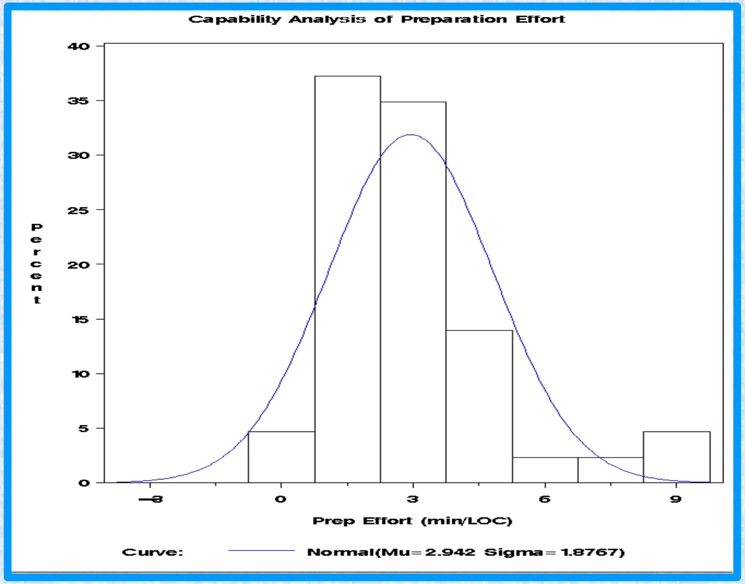
				Percentiles				
Control Variable	Mean	Standard Deviation	Skewness	1st	5th	50th (Median)	95th	99th
Preparation Effort	**2.94**	**1.88**	1.51	0.39	**0.83**	**2.40**	**6.75**	9.00
Inspection Effort	2.78	1.58	0.72	0.64	0.76	2.40	6.00	6.52
Fault Density	0.094	0.080	0.82	0.000	0.000	0.088	0.251	0.320

Statistical Process Control for Software Inspections

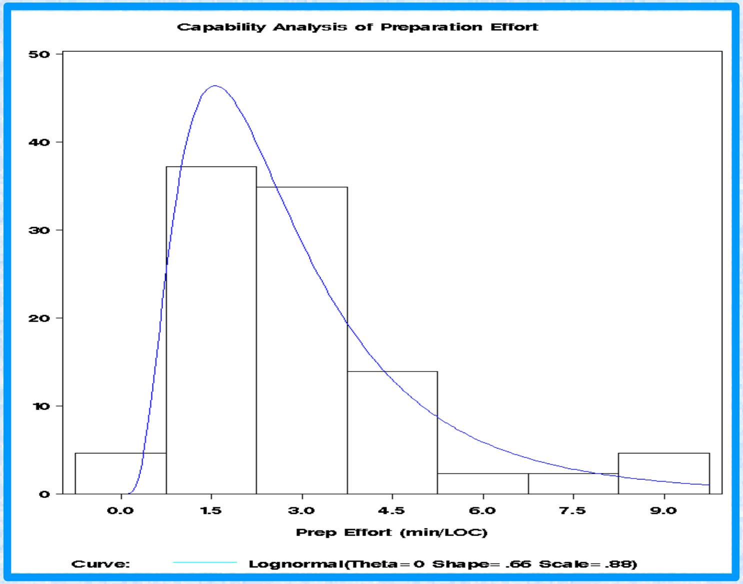
Distribution Fitting: Preparation Effort

Control Variable is Skewed to the Right

Normal Fit to Preparation Effort



Lognormal Fit to Preparation Effort



Which Distribution Would You Use, Normal or Lognormal?

Statistical Process Control for Software Inspections

Distribution Fitting: Preparation Effort

Find the probability distribution that best fits the control variable

Table 2A. Distribution Fit Statistics for Preparation Effort

Test Statistics	Percentile	Actual	Exponential	Gamma	Log Normal
Exponential	1	0.399	0.030	0.373	0.517
C^2 = 15.50	2.5	0.444	0.074	0.544	0.660
p = .0084	5	0.825	0.151	0.735	0.813
Gamma	25	1.656	0.846	1.632	1.548
C^2 = 8.79	50	2.400	2.039	2.591	2.422
p = .0665	75	3.620	4.078	3.873	3.790
Log Normal	95	6.750	8.813	6.348	7.215
C^2 = 4.91	97.5	8.400	10.85	7.322	8.893
p = .2964	99	9.000	13.55	8.565	11.34

Statistical Process Control for Software Inspections

Setting Control Limits

How "tight" should control limits be?

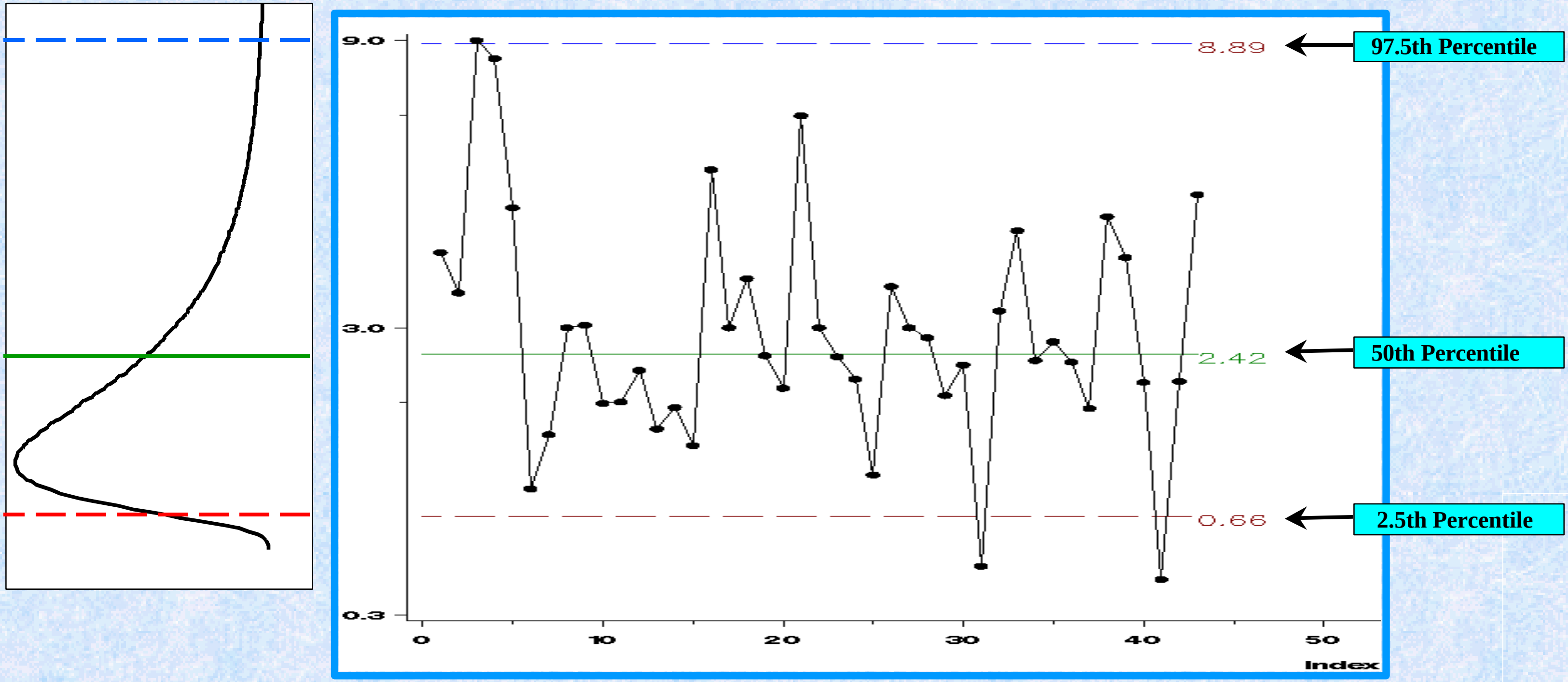
- Wide limits:
  - increase risk of passing a bad product
  - decrease risk of unnecessary re-inspection.
- Narrow limits:
  - decrease risk of passing a bad product
  - increase risk of unnecessary re-inspection.

- Q. Which is the most costly error?
- A. Allowing bad product to pass inspection!

Statistical Process Control for Software Inspections

Setting Control Limits

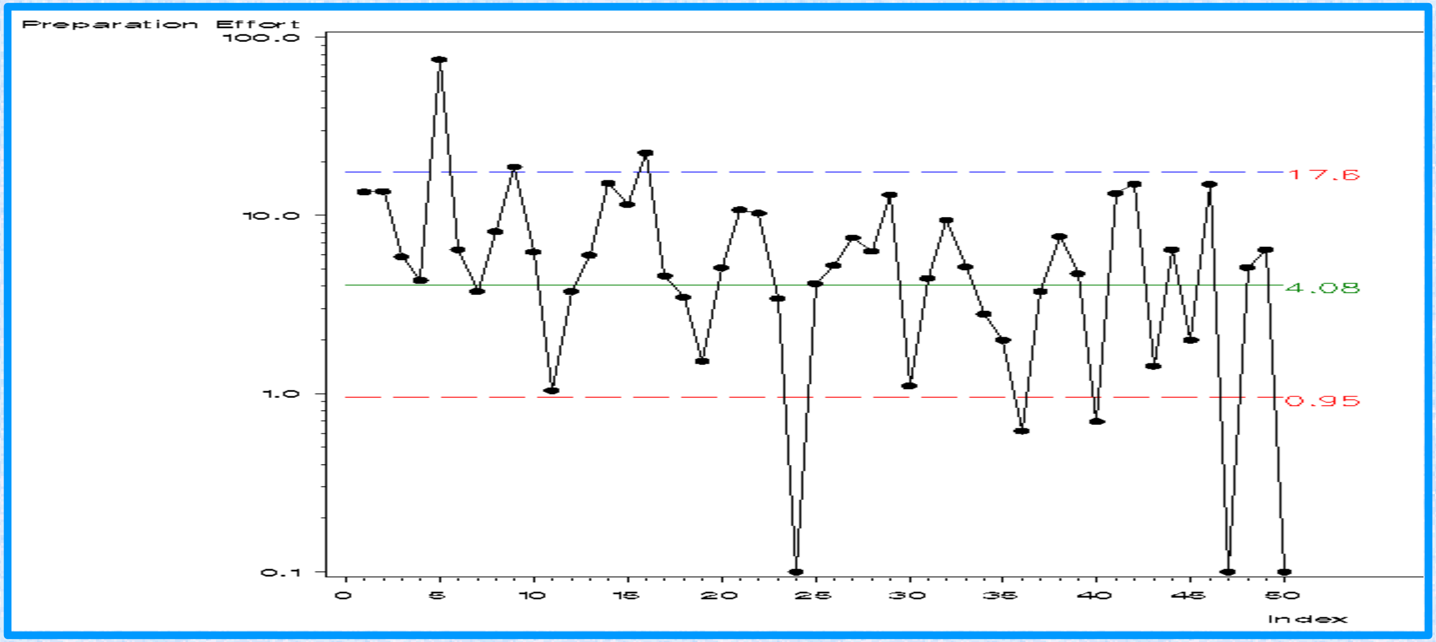
95% Control Chart for Preparation Effort



Statistical Process Control for Software Inspections

Control Charts

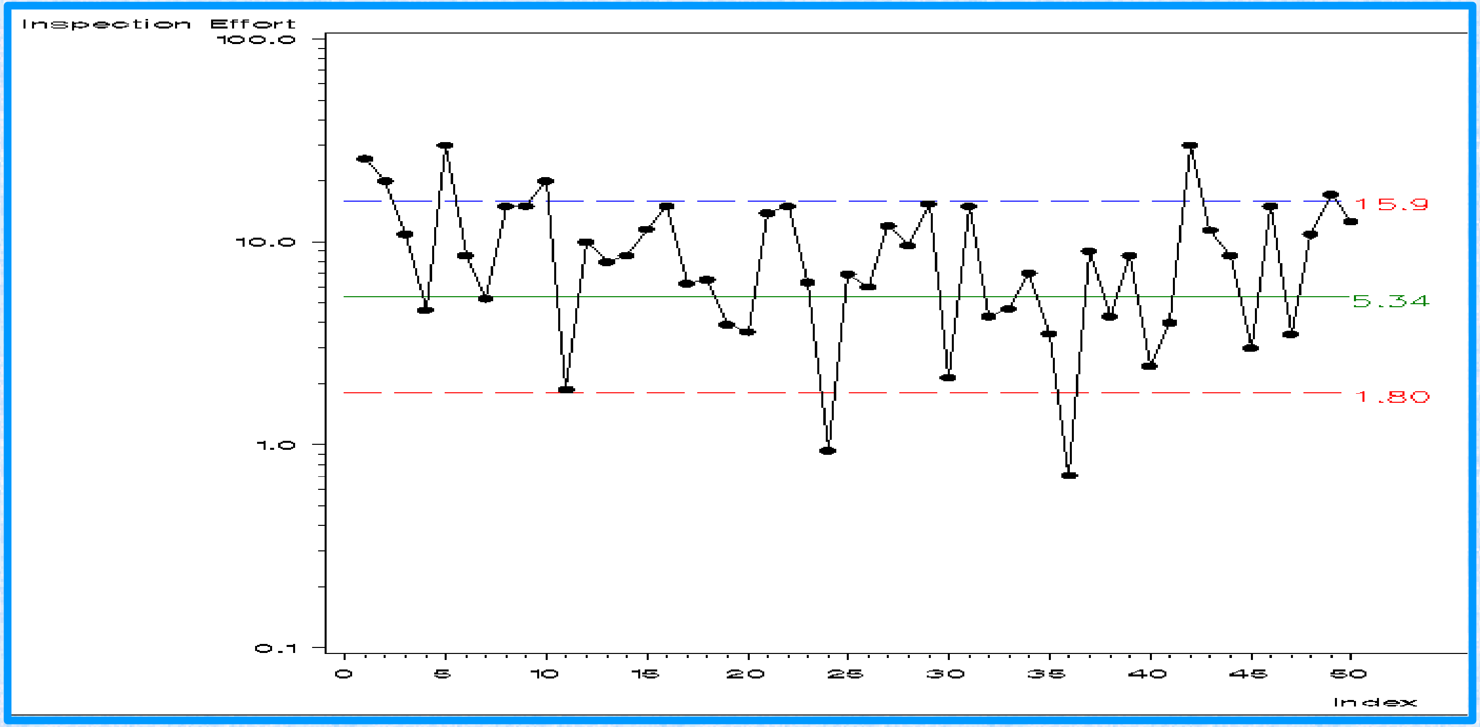
95% Control Chart for Preparation Effort (min/Page)



Statistical Process Control for Software Inspections

Control Charts

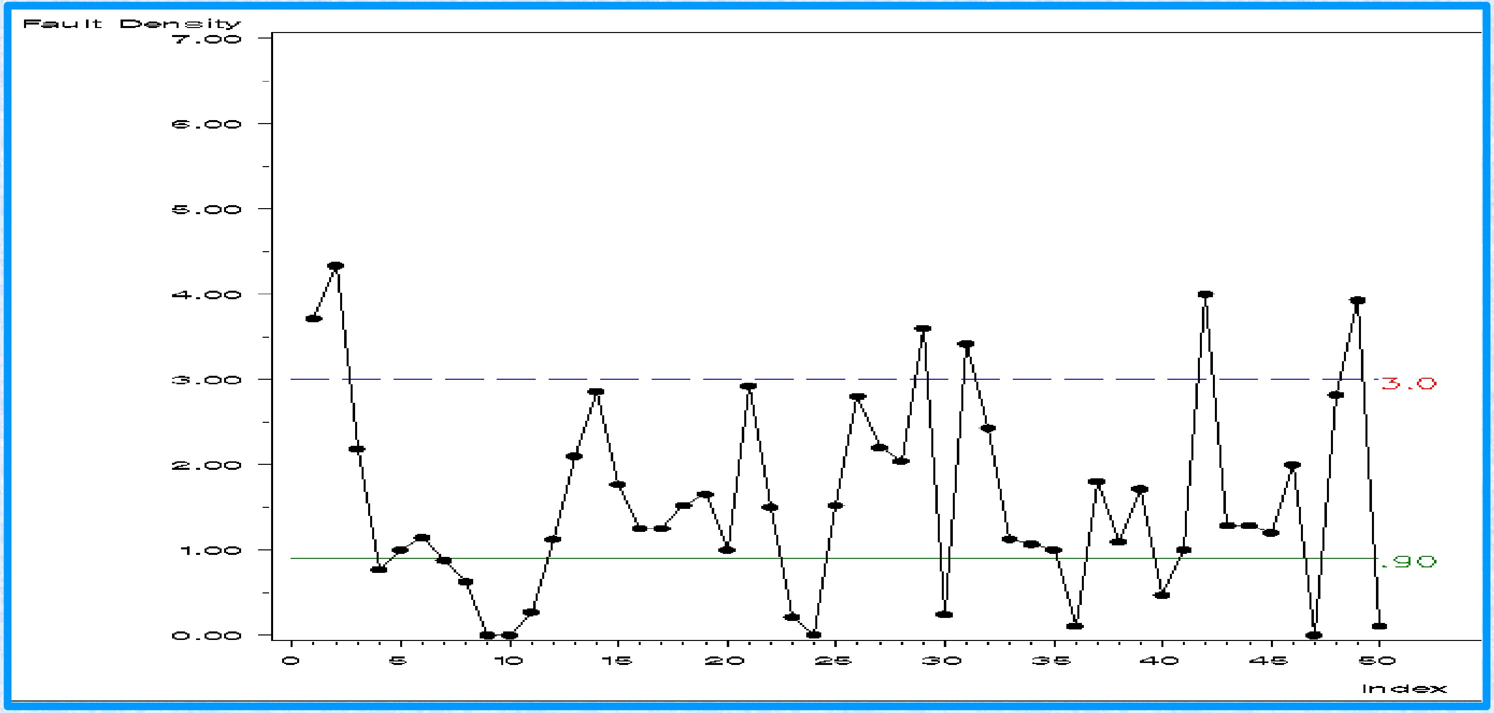
95% Control Chart for Inspection Effort (min/Page)



Statistical Process Control for Software Inspections

Control Charts

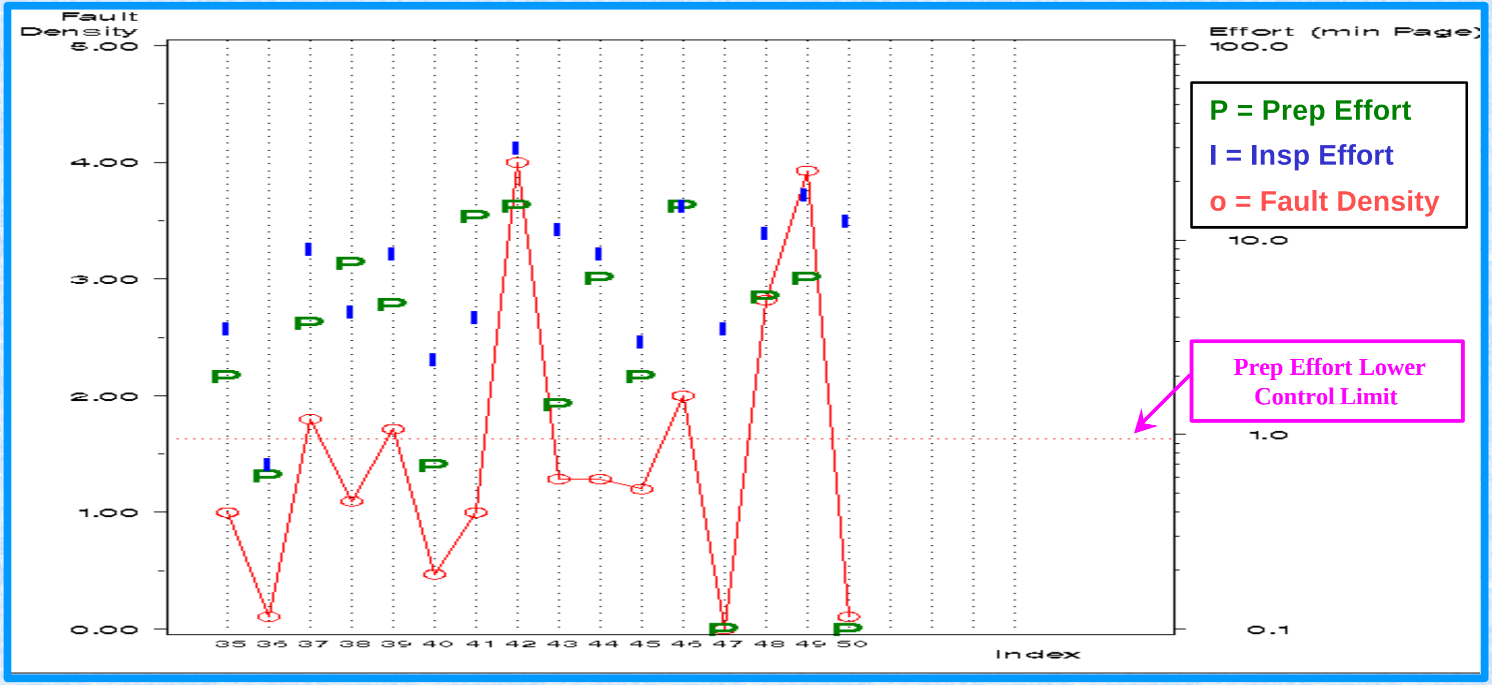
95% Control Chart for Fault Density (Faults/Page)



Statistical Process Control for Software Inspections

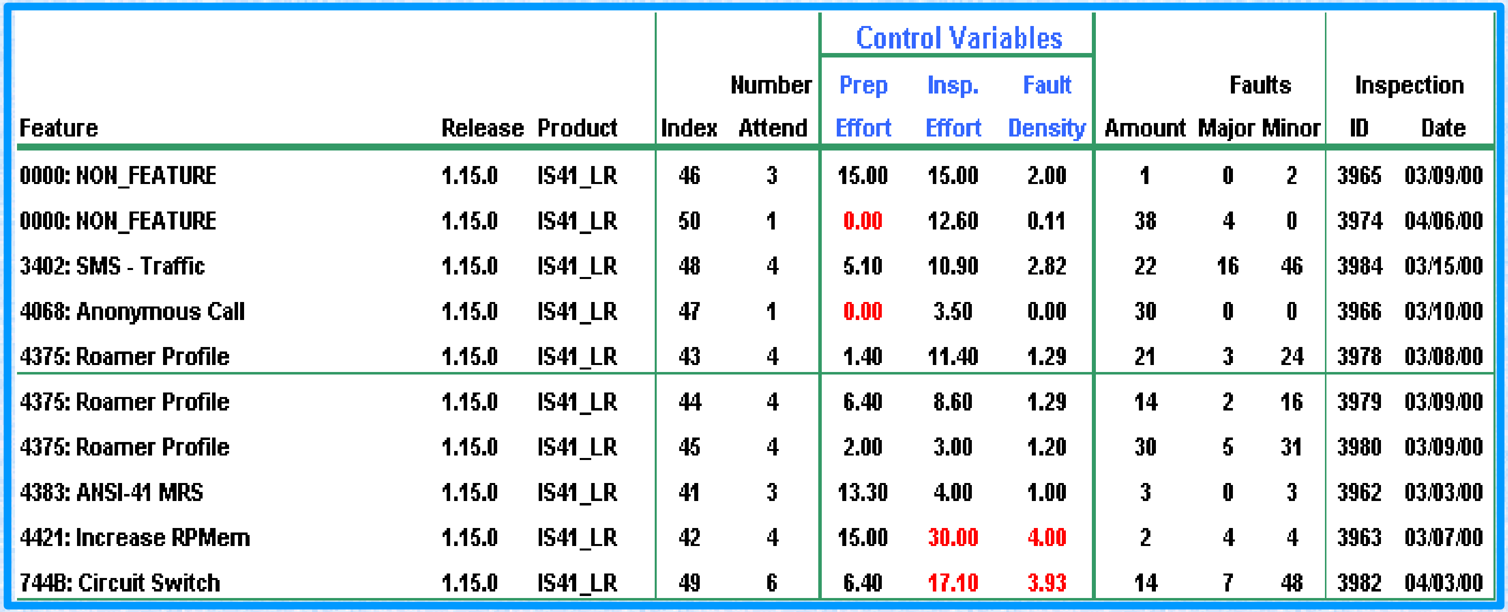
Control Charts - Overlay Chart

Fault Density versus Effort Measures



Statistical Process Control for Software Inspections

Control Chart Report



					Control Variables							
				Number	Prep	Insp.	Fault		Faults		Inspection	
Feature	Release	Product	Index	Attend	Effort	Effort	Density	Amount	Major	Minor	ID	Date
0000: NON_FEATURE	1.15.0	IS41_LR	46	3	15.00	15.00	2.00	1	0	2	3965	03/09/00
0000: NON_FEATURE	1.15.0	IS41_LR	50	1	0.00	12.60	0.11	38	4	0	3974	04/06/00
3402: SMS - Traffic	1.15.0	IS41_LR	48	4	5.10	10.90	2.82	22	16	46	3984	03/15/00
4068: Anonymous Call	1.15.0	IS41_LR	47	1	0.00	3.50	0.00	30	0	0	3966	03/10/00
4375: Roamer Profile	1.15.0	IS41_LR	43	4	1.40	11.40	1.29	21	3	24	3978	03/08/00
4375: Roamer Profile	1.15.0	IS41_LR	44	4	6.40	8.60	1.29	14	2	16	3979	03/09/00
4375: Roamer Profile	1.15.0	IS41_LR	45	4	2.00	3.00	1.20	30	5	31	3980	03/09/00
4383: ANSI-41 MRS	1.15.0	IS41_LR	41	3	13.30	4.00	1.00	3	0	3	3962	03/03/00
4421: Increase RPMem	1.15.0	IS41_LR	42	4	15.00	30.00	4.00	2	4	4	3963	03/07/00
744B: Circuit Switch	1.15.0	IS41_LR	49	6	6.40	17.10	3.93	14	7	48	3982	04/03/00

Red indicates control limit violation. Especially troubling are inspections with inadequate preparation effort.

Statistical Process Control for Software Inspections

Concluding Remarks

- What should we "take home" from this presentation?
- 1. Missing defects is **EXTREMELY** costly.
- 2. Control charts that ignore distribution shape are:
  - Difficult to understand
  - Lead to erroneous conclusions
- 3. The four basic steps to building control charts.
- 4. The high cost of escaped defects require setting narrow control limits.

Statistical Process Control for Software Inspections

Don Porter
Motorola

dporter1@email.mot.com

847-632-6430

Statistical Process Control for Software Inspections

Don Porter

ABSTRACT

Attempts to create user-friendly statistical process control (SPC) charts for software inspections often have failed. A principle cause of these problems is the failure to recognize the asymmetric distributions of the critical control variables, and to incorporate this fact into control chart design.

This paper provides innovative guidelines for inspections SPC. The innovations include:

- *Tracking fault density = faults/(amount inspected) rather than the number of faults. Since the number of faults follows a Poisson probability law, the fault density obeys a related exponential distribution.*
- *Tracking preparation and inspection efficiencies rather than rates. These are defined as (preparation time)/(amount inspected) and (inspection time)/(amount inspected). These variables fit gamma or lognormal distributions quite well.*

The fault density chart is easy to read because the control limits remain constant for products of varying sizes. This helps the user compare inspections over time. The effort measures provide many advantages over their more familiar inverses. They are defined when time is zero, and their probability distributions are recognizable.

Analysis is facilitated because all control variables are consistently normalized by amount inspected. Thus, plot overlays help pinpoint inspections that fail to find faults, and indicate the reasons. These benefits have overcome the disdain that some in the development community have shown for the unwieldy charts in the past, and has increased the stature of SPC in their eyes.

1. Introduction

The estimated cost to fix a software defect found after release is 100 times greater than a defect found in the phase of origination. In software organizations, this repair cost has been estimated to range from $8,000 to $40,000, depending on the severity and size of the defect.[1] In addition to these direct costs, there is the cost in lost customers.

If defects can be detected at an early phase of the software life cycle, our customers will benefit from more timely delivery of a higher quality product. Obviously, the benefits to Motorola extend beyond improved customer satisfaction, and include large reductions in internal costs.

In order to improve the efficiency of the development and inspection processes, key associated variables must be measured. When abnormal values of these variables are detected, a re-inspection may be required. Further investigation may reveal and remove an assignable cause for the out of control signal. Statistical process control (SPC) methods have been used for decades to improve quality and processes. Implementing SPC for the inspection process will provide a substantial payoff.

The target audiences are the metrics and quality analysts who implement SPC and the decision-makers that investigate malfunctions of the development or inspection processes. It is hoped that a thorough application of SPC will provide both cost savings and insight into potential avenues for improving our processes.

2. Modifying Standard SPC for Code and Document Inspections

The basic ideas behind SPC can be applied to the inspection process. However, software products are much different from the manufacturing products used in most SPC applications, which are mass-produced by machines. Instead of a uniform product, our software engineers write and inspect products with widely varying purposes and levels of sophistication. While the inspection process remains the same, each observation is unique.

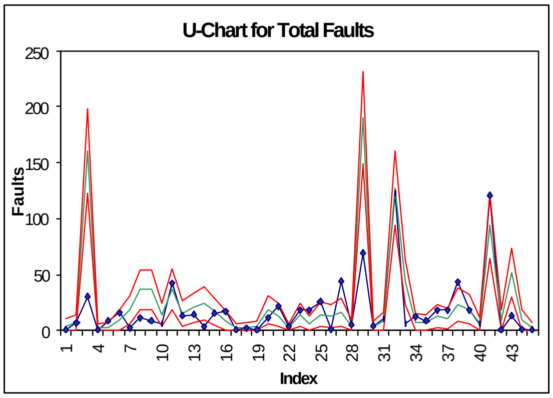
SPC was developed for environments where measurements are at least approximately normally distributed. But the inspection variables that we monitor will not, in general, have such nice, bell shaped distributions. Can SPC be modified to produce control charts for other situations? Yes. Methods have been developed for:

- Binomial, Hypergeometric, Poisson, and other probability distributions.
- Serially correlated variables.
- Cumulative measurements.
- Measurements that do not obey any particular probability law.

2.1 The Traditional Control Chart Approach

In our organization, traditional control charts for inspections were very difficult for the end users to understand. This traditional approach relied on a modified "u-chart" approach, based on the Poisson distribution for software faults. U-charts use control limits based on a center-line ± 3 standard deviations. The standard deviation of a Poisson random variable is the square root of the expected value. The expected value, or number of faults found in inspection, depends on the size of the module being inspected. The u-charts are obtained by multiplying module size by historic fault density to obtain the expected number of faults. Therefore, the traditional approach yields control charts with variable limits. While there is nothing wrong with variable limits in principle, it makes for a very messy control chart. Such a chart is shown in Figure 1A.

Figure 1A. Variable Limits Control Chart for Total Faults

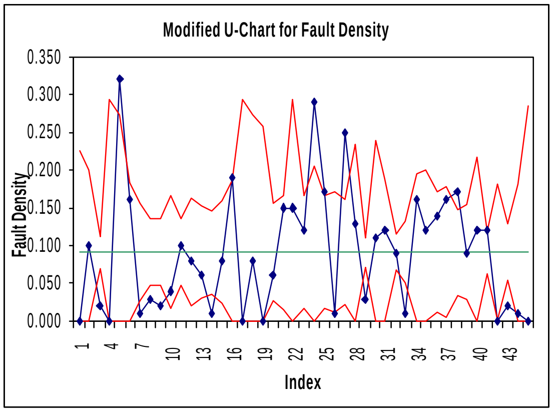


[1] "Software Engineering Economics", Barry W. Boehm, Prentice Hall, October 1981.

In the chart the red lines are the control limits, the green line is a target value, and the black line and dots are the data values. The wider control limits apply to inspections of more lines of code, hence a greater expected value and standard deviation. End users of the charts found the logic of variable limits charts difficult to grasp. Also, these charts made it very difficult for them to assess progress over time. Does the chart in figure 1A show that things are getting better or worse?

One attempt to improve the readability of the u-chart was to divide faults by module size to get fault density. The standard deviation of this variable is obtained by dividing the standard deviation of faults by module size as well. This modified u-chart helped somewhat. At least the center-line was now constant. The control limits were still variable, however. Unlike chart 1A, the modified u-chart in figure 1B has wider limits for smaller sized modules, reflecting the fact that smaller modules could result in much higher fault densities. For example, five errors in a module with 20 lines of code would result in a fault density of .25. But given an historical fault density of about .1, it would be extremely unlikely to find 50 errors in a module of size 250.

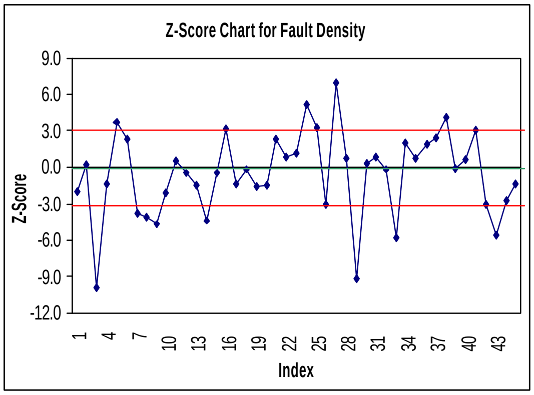
Figure 1B. Modified U-Chart for Fault Density



The chart in figure 1B is easier to read, but the variable control limits are still difficult for the end user to relate to. One inspection could have a fault density that was outside the upper limit, yet be lower than other fault densities. Chart 1B is still not intuitive.

Another attempt to ameliorate the problem was to use a "z-chart". This type of chart standardizes the values by subtracting the mean and dividing by the standard deviation. The result is shown in Figure 1C.

Figure 1C. A Z-Chart for Fault Density



The control limits are now stabilized, but problems remain. First, the user has little intuitive feel for data values in the range -3 to +3. Second, z-charts are designed for variables that have normal (bell-shaped) distributions. For such distributions over 99% of the data should lie between -3 and +3. The fault densities in figure 1C obviously have a highly skewed nature. Therefore, neither this z-chart nor any other chart that assumes a normal distribution will be adequate for the task of statistically controlling faults or fault density. What the end users needed were accurate control charts in the original data units with control limits that they could relate to.

2.2 New Inspection Control Chart Approach

Clearly, a new approach was needed. No matter what the distribution of the measurement, constructing and applying control charts involves the same concepts:

1. What is the distribution of in-control values?
2. What is the central or target value?
3. How tight should the control limits be? What is the cost trade-off between risking intervention when the process is in control versus not intervening when it is out of control?
4. How can accumulated historical data be used to improve the process and perhaps recalculate tighter control limits?

Three new control variables were chosen, each one normalized by the amount inspected. They are:

1. Preparation Effort = (Total Preparation Time) / (Amount Inspected)
2. Inspection Effort = [(Total Meeting Duration)*(Total Number of Inspectors)] / (Amount Inspected)
3. Fault Density = (Total Number of Major Faults + Total Number of Minor Faults) / (Amount Inspected)

Time is measured in minutes, amount inspected in pages for document inspections and non-comment lines of code for code inspections. Major and minor faults are included in the fault density calculation. There are two reasons for this:

- All faults impact the customers.
- The distinction between major and minor is often obscure, and is at times abused for convenience.

Traditionally, preparation and inspection rates have been expressed as amount / time, the inverse of the "efficiencies" presented here. The problems with the traditional representation are:

1. A time variable may be zero, yielding an undefined control variable.
2. Time is the random variable, while amount inspected is a fixed, normalizing variable. The inverse of a random variable is inevitably more difficult to analyze, especially if it's range includes zero. Indeed, the time rates above will be shown to obey well-known probability distributions. This is not the case with their inverses.

The first two variables are measures of the inspection process itself. The third control variable may measure either the development or the inspection process. That is, a low fault density may be the result of very good development. But it could indicate an inspection so cursory that it fails to detect faults present in the code or document. As an inspection process becomes more repeatable the variability of the effort variables will be reduced, which would be reflected in tighter control limits.

The control variables are all normalized by amount inspected. Thus, control charts have limits based on time (or faults) per-unit inspected. Large modules require more time, and have more faults, than small ones. If control charts used non-normalized values, the central values and limits would depend on module size and vary widely.

Since all three variables are similarly normalized, overlay plots can provide a "poor-man's" multivariate chart, as will be seen below.

3. Example of SPC Applied to Software Inspections

The data consists of 43 observations on code inspections for a recent software release, recorded between April 1997 and August 1998. Only code inspections of 20 lines or more were included, since small code inspections can lead to absurdly high values of the three control variables defined above. Output is displayed to guide the selection of distributions. The control charts themselves appear in figures 4-6 below.

Table 1 provides information on the three control variables. This table gives the descriptive statistics of mean, median, standard deviation and skewness, as well as some selected percentile points. A symmetric distribution has a skewness coefficient of zero. Positive skewness indicates a distribution with a longer right tail than left. It is important to recognize non-symmetric distributions, since control charts based on assumed symmetry would then produce too many false out of control signals. Table 1 shows that the inspection control variables do not possess the bell-shaped, symmetric distributions used in classical control charts.

Table 1. Control Variable Statistics

Control Variable	Mean	Standard Deviation	Skewness	Percentiles 1st	5th	50th (Median)	95th	99th
Preparation Effort	2.94	1.88	1.51	0.39	0.83	2.40	6.75	9.00
Inspection Effort	2.78	1.58	0.72	0.64	0.76	2.40	6.00	6.52
Fault Density	0.094	0.080	0.82	0.000	0.000	0.088	0.251	0.320

Each of the control chart variables exhibits a substantial right skew. Note that the standard deviation of each is much too large to allow a lower control limit more than 1.5 standard deviations below the mean. When dealing with such non-normal data, there are two options:

- Transform the variable into a normal shape and present the transformed variable in the control chart.
- Base control chart limits on the percentiles of the original variables, and present in the original metric.

The latter approach benefits those interested in the chart variables. They have an intimate knowledge of the data, and can relate more intuitively to the second type of chart.

3.1 Determining the Distributions of Control Variables

This report was prepared using a statistical package that fits several probability distributions. The useful distributions for these control variables are the Lognormal, Exponential and Gamma. Chi-Square tests are used for statistical comparisons.

3.1.1 Preparation Effort

Table 2A gives the results of the distribution fitting for preparation effort. The first column provides the formal statistical tests, including distribution name, Chi-Square test statistic, and p-values. A small p-value indicates that the data does not fit the distribution well. Actual and fitted percentile points fill out the rest of the table. For example, the first row shows that 1% of preparation efficiencies are less than or equal to .399 minutes per line of code. The next three rows give the fitted percentiles for each of three distributions.

According to Table 2A, the Lognormal distribution gives the best statistical fit, since it has the smallest Chi-Square test statistic and largest p-value. But it is also important to check the extreme points, since these are where the control limits will lie. Here the results are mixed. The Gamma distribution comes closest to the actual 1st and 2.5th percentile points, but the Lognormal comes closest to the 97.5th and 99th percentiles. Low values will be flagged as "out of control" more often with the Lognormal limits than the Gamma, but the reverse is true for large preparation efficiencies. Figures 2A and 2B show histograms of preparation efficiencies with superimposed distributions.

Table 2A. Distribution Fit Statistics for Preparation Effort

Test Statistics	Percentile	Actual	Exponential	Gamma	Log Normal
Exponential	1	0.399	0.030	0.373	0.517
X^2 = 15.50	2.5	0.444	0.074	0.544	0.660
p = .0084	5	0.825	0.151	0.735	0.813
Gamma	25	1.656	0.846	1.632	1.548
X^2 = 8.79	50	2.400	2.039	2.591	2.422
p = .0665	75	3.620	4.078	3.873	3.790
Log Normal	95	6.750	8.813	6.348	7.215
X^2 = 4.91	97.5	8.400	10.85	7.322	8.893
p = .2964	99	9.000	13.55	8.565	11.34

Figure 2A. Gamma Fit to Preparation Effort

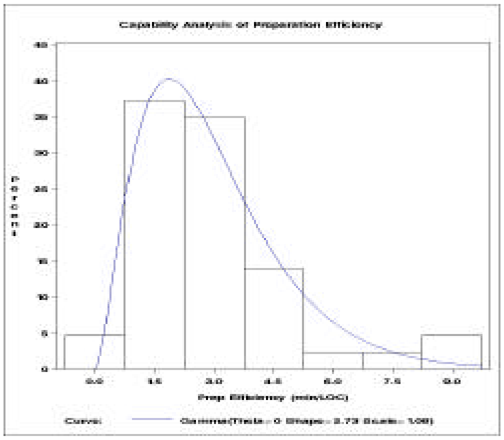
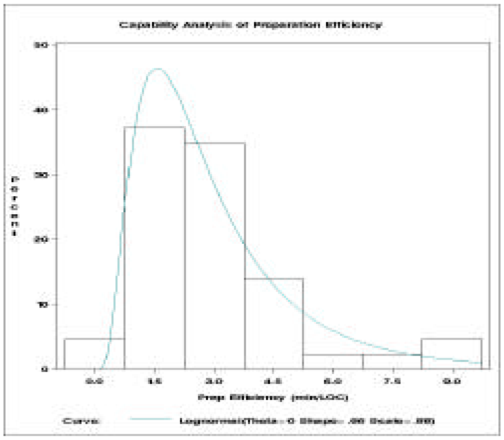


Figure 2B. Lognormal Fit to Preparation Effort



Figures 2A and 2B show that either the gamma or lognormal distributions give a good fit to the data. In fact, Lognormal and Gamma distributions tend to be very similar in shape.

3.1.2 Inspection Effort

Table 2B gives the results of the distribution fitting exercise for inspection effort. The Gamma distribution gives the best fit, with the Lognormal providing a statistically sound fit as well. However, when the endpoints are compared, the Gamma provides better control limits. The exponential distribution is clearly poor. Figures 3A and 3B compare the Log Normal and Gamma curves to the inspection effort histogram.

Given the toss-up between the Gamma and Lognormal distributions, two points are worth considering:

1. The Gamma distribution can be expressed as a sum of exponential random variables. The exponential probability distribution models the time or distance between faults, so total preparation or inspection time should be well approximated by a Gamma distribution.
2. On the other hand, the lognormal distribution is easier to work with and easier to communicate.

Given these considerations, the author feels that the Lognormal would be a good choice here.

Table 2B. Distribution Fit Statistics for Inspection Effort

Test Statistics	Percentile	Actual	Exponential	Gamma	Log Normal
Exponential	1	0.638	0.028	0.405	0.537
X^2 = 16.96	2.5	0.675	0.071	0.574	0.677
p = .0046	5	0.761	0.143	0.759	0.826
Gamma	25	1.748	0.801	1.603	1.525
X^2 = 14.4	50	2.400	1.930	2.482	2.336
p = .0061	75	3.600	3.860	3.639	3.577
Log Normal	95	6.000	8.342	5.844	6.604
X^2 = 16.06	97.5	6.000	10.27	6.707	8.060
p = .0029	99	6.522	12.82	7.803	10.90

Figure 3A. Gamma Fit to Inspection Effort

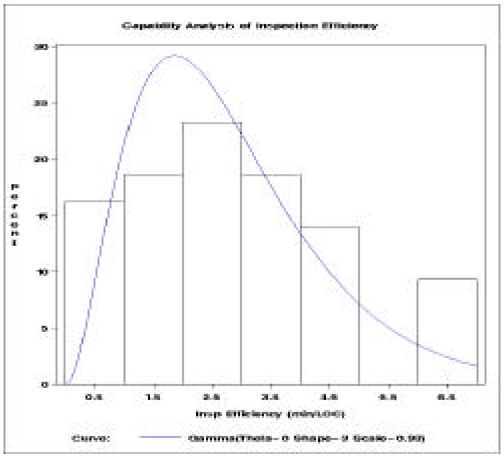
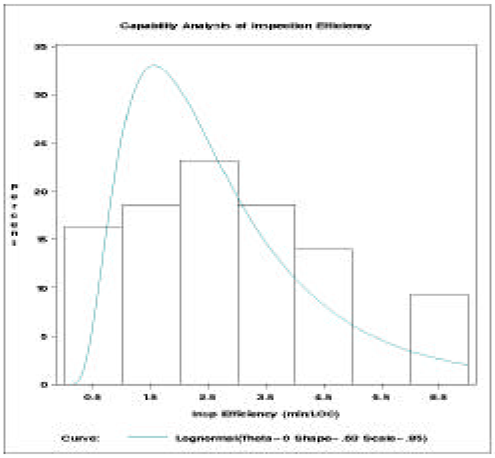


Figure 3B. Lognormal Fit to Inspection Effort



3.1.3 Fault Density

Table 2C provides the statistics and percentile points for the fault density, and Figures 4A and 4B provide a graphical view of the fits.

The table shows that the Lognormal distribution is clearly out of the running for fault density. The exponential gives the best fit statistically, followed by the Gamma. When the percentile points of these two distributions are compared, the Exponential is a clear winner, especially in the endpoint regions.

Table 2C. Distribution Fit Statistics for Fault Density

Test Statistics	Percentile	Actual	Exponential	Gamma	Log Normal
Exponential	1	.00001	.00094	.00001	.00001
$X^2 = 9.13$	2.5	.00001	.00237	.00004	.00004
p = .0580	5	.00001	.00481	.00019	.00010
Gamma	25	0.0127	0.0270	0.0073	0.0024
$X^2 = 17.3$	50	0.0879	0.0650	0.0381	0.0218
p = .0006	75	0.1500	0.1299	0.1213	0.1984
Log Normal	95	0.2514	0.2807	0.3753	4.7412
$X^2 = 61.6$	97.5	0.2857	0.3457	0.4968	13.289
p = .0001	99	0.3200	0.4315	0.6632	44.051

Figure 4A. Exponential Fit to Fault Density

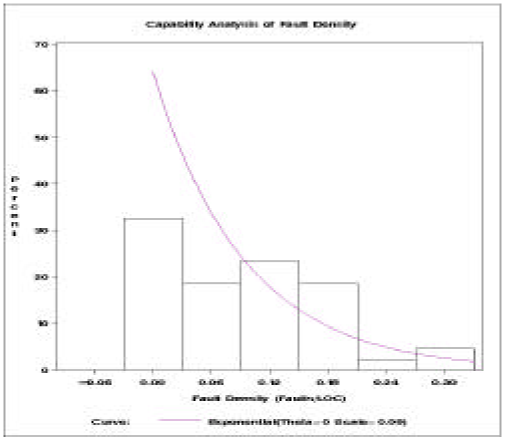
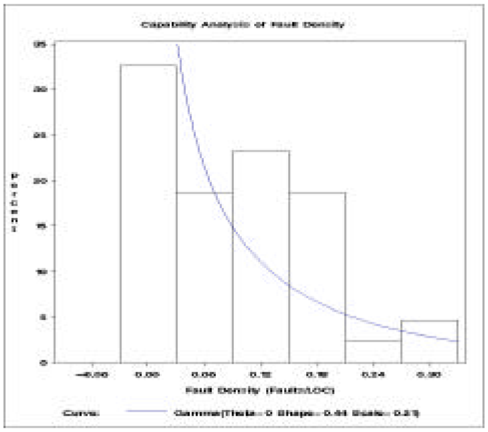


Figure 4B. Gamma Fit to Fault Density



The figures seem to indicate that the gamma gives a better fit, but this is an artifice of the manner in which the statistical software package produces histogram intervals. There is also a compelling theoretical reason to accept the exponential model. Total faults in a code module possess a Poisson probability distribution, characterized by a constant rate of fault introduction. By definition, the fault density is that rate of fault introduction. Since the waiting time between Poisson events has an exponential distribution, it is not surprising that the fault density is well modeled by this distribution.

3.2 Control Charts

The best fitting distributions are used to determine the center-line and limits for the control charts. Exactly where to set the limits is a decision made after careful consideration of costs. For software development, there are two possible errors one can make:

1. Deciding to re-inspect when the code was well written and properly inspected.
2. Concluding the code is good and the inspection properly performed when either rewriting the code or re-inspection is in order.

If the first mistake is more costly than the second control limits should be wide, making a re-write or re-inspection less likely. In software development, the opposite is generally the case. As the introduction made clear, the cost of allowing a fault to escape detection is high compared to the cost of re-inspection. So control limits for preparation and inspection efficiencies were set at the estimated 2.5th and 97.5th percentile points. This means that 5% of the good code will be re-inspected.

In the following control charts, green lines represent the 50th percentile, and red lines the control limits. Notice that the vertical axes for the effort measures are scaled logarithmically. Tables 2A, 2B and 2C provided the control limits for the charts in figures 5, 6 and 7, respectively.

<u>Preparation Effort</u>

Figure 5 shows the chart for preparation effort. There are four unusually low values, indicating inadequate preparation time. The last section of this paper will provide actions that the decision-maker might wish to consider.

<u>Inspection Effort</u>

The chart for inspection effort also shows some lower out of bound signals. Some of these correspond to the low points in the preparation effort chart.

<u>Fault Density</u>

The chart for fault density only has a center-line and an upper control limit. It is not at all uncommon for code modules to be error-free, so we do not need a lower limit. Figure 6 shows two code modules with exceptionally high fault densities.

Figure 5: Control Chart for Preparation Effort

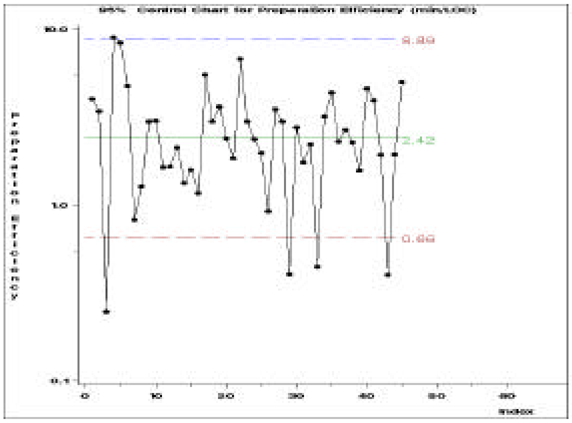


Figure 6: Control Chart for Inspection Effort

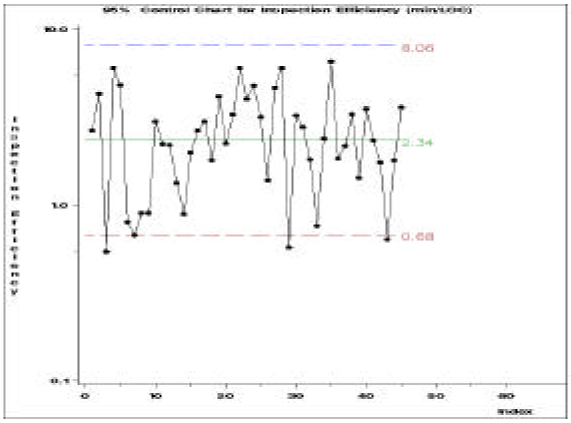
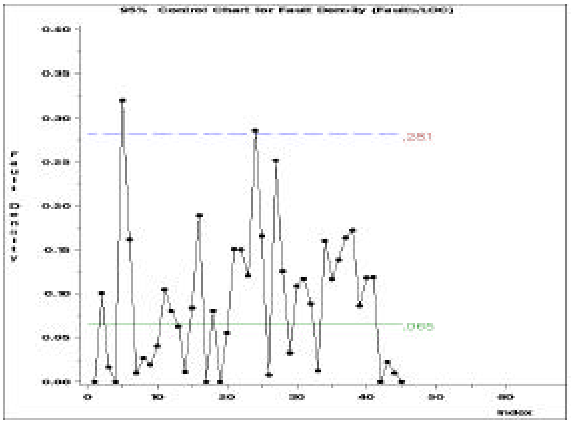


Figure 7: Control Chart for Fault Density



3.3 Responding to an Out-of-Control Signal

When the preparation or inspection effort variables exceed the upper limit, the inspection process has taken an abnormal length of time. This may indicate that:

1. The document or code module is unusually difficult to inspect.
2. The inspection process is not being performed in accordance with a standard, repeatable process.

If #1, attention should be paid to the defect density. If that density is quite high, (even if it is within its control limit), the product may require rework. This is a judgment call that requires some experience on the part of the decision-maker.

If #2, then an investigation should be undertaken. Do inspections facilitated by the same moderator tend to have very high prep or inspection efficiencies? Is this document inherently difficult to understand? Maybe the document author tends to write in a very difficult manner and needs to take a business writing class. The inspections database contains a rich set of administrative variables to facilitate this type of root cause analysis. Another option is to perform analysis of variance to see if the control variables differ significantly for some categories, such as coding language.

If the fault density exceeds the upper control limit there is a clear indication of a poorly written document or code module. This is especially true if the preparation and/or inspection efficiencies are also low, for the inspectors found a large number of faults without expending a lot of time. The inspected product should be returned to its author for rework.

If the preparation or inspection effort is too low, there is a possibility that the inspection has not been thorough. The decision-maker should check the fault density. If this is very low, it is likely that the product needs to be re-inspected. On the other hand, the document or code may be very straightforward, in which case it is understandable that the control variables would take low values.

Unusually low fault densities could indicate that a segment of code was exceptionally well written. One would in fact be very confident of this conclusion if the inspection was very thorough. On the other hand, an extremely low fault density will more likely indicate a poor inspection performance when accompanied by low preparation or inspection efficiencies. Therefore, one must consider all three variables before making a decision to re-inspect

The decisions considered in this section require simultaneous review of all control variables. Overlay plots are especially useful in this regard. Two such plots are given in figures 8A and 8B. Three points stand out in the overlay charts. These are the 3rd, 33rd, and 43rd points. They are characterized by low fault densities, and low preparation and inspection rates. This is a warning that the low number of faults is perhaps due more to a poor effort than it is to a good module. The manager of the inspection team should investigate to determine the root cause.

Figure 8A: Overlay Charts for Fault Density and Preparation Effort

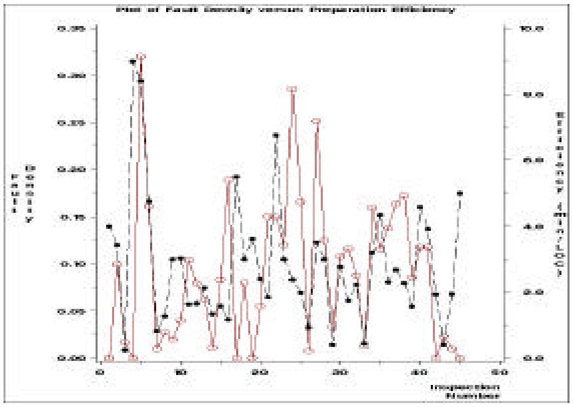
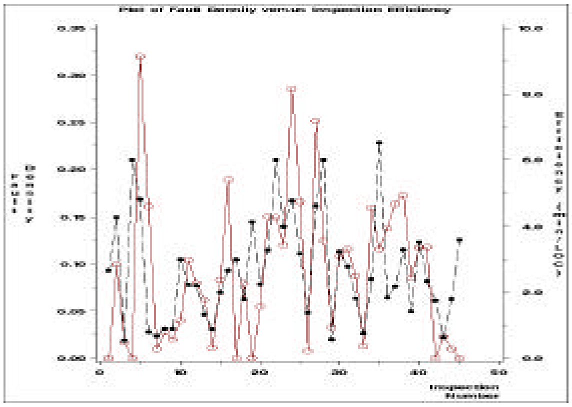


Figure 8B: Overlay Charts for Fault Density and



Inspection Effort

In contrast, a point such as 19 shows zero faults, but adequate preparation and inspection efficiencies. Therefore, the low faults can be attributed to a superior product rather than insufficient inspection effort.

4. Bibliography

[1] Applied Software Measurement: Assuring Productivity and Quality, Capers Jones, McGraw Hill, August 1996

[2] Controlling Software Projects: Management, Measurement and Estimation, Tom Demarco

[3] Introduction to Statistical Quality Control, Douglas C. Montgomery, John Wiley & Sons, 1985

[4] Software Engineering Economics, Barry W. Boehm, Prentice Hall, October 1981,

[5] Understanding Variation: The Key to Managing Chaos, Donald J. Wheeler, 1993, SPC Press Incorporated. ISBN 0-945320-35-3

5. Glossary

Control Chart Terminology

Assignable Cause A shift of the process level or an increase in process variability assignable to a specific reason, as opposed to the random variation commonly associated with the process.

Center Line A horizontal control chart line representing the average value of the process.

Control Chart A graphical characterization of measured process outputs (see figure 2 on page 4 below). Control charts present limits that are based on the current distribution of output values. When observed process values fall inside these limits, the process is declared "in control".

Control Limits A set of one or two horizontal lines on a control chart. When an observed process value is either greater than the upper control limit or is less than the lower control limit, the process is declared "out of control".

In control A state in which a process is producing output consistent with its expected distribution.

Out of Control A state in which a process is producing output that is inconsistent with its expected distribution. This may result from a shift of the distribution of output values or from an increase in its variability.

Outlier A number that is inconsistent with the other values in a sample.

Random Cause A process will not produce identical values every time. Outputs vary randomly, in accordance with an underlying probability distribution. Such variability cannot be assigned to a specific cause, and is accepted as uncontrollable within the scope of the current process.

Target Can refer to the desired level of process output ("voice of the customer"). For control charts, it commonly refers to the expected level of the process, and provides the centerline of the chart.

Statistical Terms

Mean A measure of central tendency, given by the arithmetic average of a set of values. For a random variable, the mean is also called the expected value.

Percentile When a sample is arranged in ascending order (order statistics) percentiles can be determined. For a given percentage P, the associated percentile is a value such that P% of all values are less than or equal to that percentile. For example, consider the set of eight numbers {2, 5, 5, 9, 11, 15, 16, 22}. The 25th percentile (1st Quartile) is 5. The 50th percentile (the 2nd Quartile or Median) is 10, and the 75th percentile is 15.5. Note that 15.8 could also serve as the 75th percentile. There are a variety of methods for interpolating between values in a case like this, but they differ little.

Type I Error When testing an hypothesis of the form H0: Population Mean = k, a type I error occurs when one rejects H0 when, in fact, the mean is equal to k. In terms of control charts, a type I error occurs when we conclude the process mean has moved from centerline when it has not.

Type II Error When testing an hypothesis of the form H0: Population Mean = k, a type II error occurs when one accepts H0 when, in fact, the mean is not equal to k. In terms of control charts, a type II error occurs when we conclude the process mean has remained at the centerline when it has actually drifted.

Don Porter

Don Porter graduated from Northern Illinois University, where he received the Master of Arts Degree in Economics in 1978 and the Master of Science Degree in Statistics in 1980.

Don taught courses in Economics, Mathematics, and Statistics until 1985, when he entered the business world as a statistical consultant. He has worked for several Fortune 500 companies, including Continental Telephone, Ameritech, Abbott Labs, Amoco, and Motorola. His clients have also included several smaller companies in the Chicago area.

Don has experience in statistical modeling, experimental design, sampling design, statistical quality control and process improvement. He has developed and automated all major categories of control charts, including CUSUM and time series residuals charts.

Don is a member of the American Statistical Association and the American Society for Quality.