THE IMPORTANCE OF UNDERSTANDING VARIATION IN SOFTWARE METRICS

David C. Trindade, Ph.D.
Sun Microsystems, Inc. (Java Software)
Cupertino, CA, USA

ABSTRACT

To improve software quality, companies often employ metrics to measure progress and highlight areas needing attention. Primarily, however, the concern has been with the actual values in the metrics as opposed to the variation in time of the metrics. A Pareto analysis of the components of an index is often employed to determine which elements contribute the most to the total index. We show in this paper why also evaluating the individual component variations compared to the total variation can provide valuable insights into which areas contribute most to variation. We explain also how common practices of measuring various cycle times can result in excessive swings in periodic indices. By controlling the variation in software metrics, the measures become more stable and predictive. In addition, the identification of the high variation components provides opportunities for meaningful process improvement.

INTRODUCTION

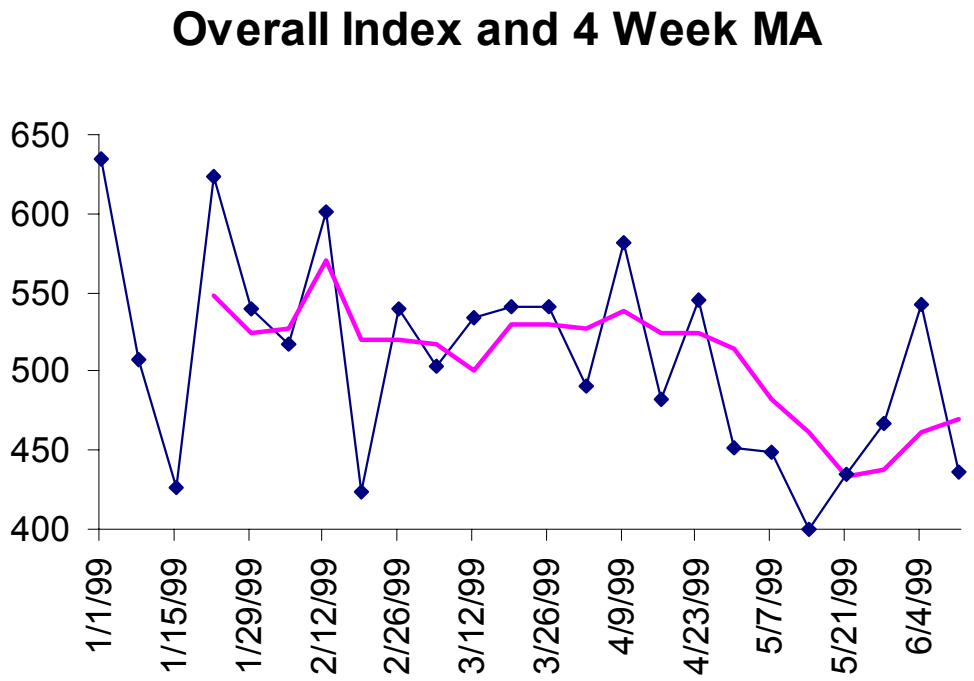
Many software development organizations employ various types of metrics to measure progress and identify areas needing attention. For example, there may be measures of bug counts by severity and priority; cycle times for the various stages of reporting, evaluating, fixing, testing, and closing bugs; frequency of customer calls; cost of resolving customer complaints; survey responses; and so on. Some companies may even combine the individual measures into an overall index based on a weighted combination of the components. This summary index is viewed and tracked by management as a key indicator of product or process quality.

In assessing the opportunities for improving quality, management will usually focus efforts on those components that contribute the most to the index. Often the results are presented graphically to show time trends using a simple line chart. However, graphical comparative techniques can be misleading because of the self-adjusting scaling properties of software.

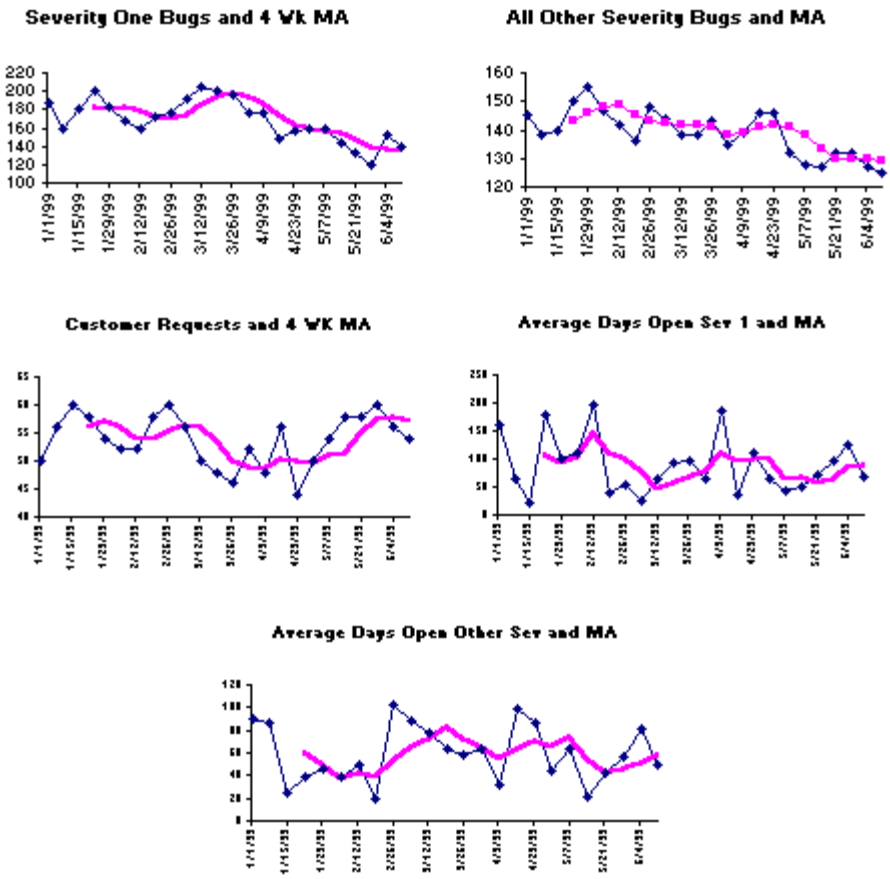
For example, the graph below shows a typical index used as an overall measure of quality. Five components are involved:

1. the total number of open severity one bugs,
2. the total number of open bugs of severity two through four,
3. the total number of open customer product enhancement requests,
4. the average number of days for severity one bugs in the fix process,
5. the average number of days for severity two through four in the fix process.

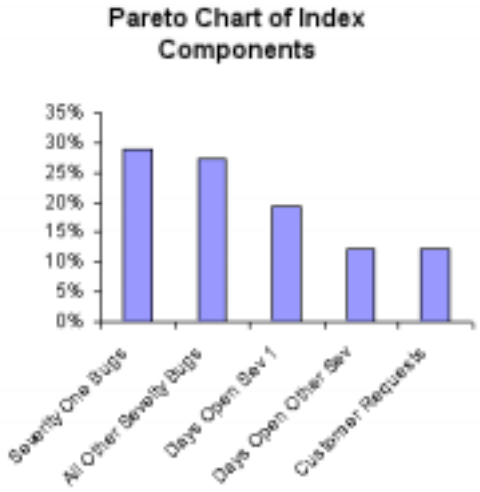
We assigned the weights 2,1,2,2,1, respectively, to these five components. A four-week moving average has been used to smooth out the data and make trends more visible. Here, the choice of four weeks for the averaging period is purely arbitrary.



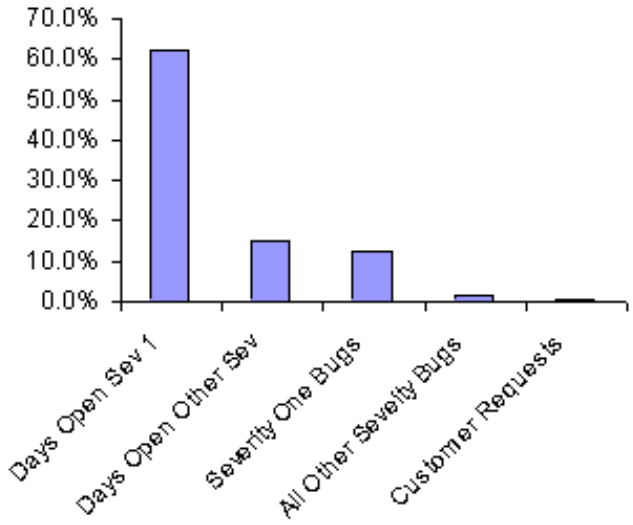
We see that there appears to be a trend downward in the index, but what is the cause of the improvement? To answer this question, management may choose to look at the individual graphs. We present the component graphs below. Apparently, the bug counts are declining for all severities.



It is not easily evident, however, by comparing graphs across categories to see which components contribute the most to the total index. Instead, we try to present the data in a manner that is interpretable quickly. The Pareto chart of contributors is shown below.



It is now apparent that severity one and all other severity bugs contribute the most to the index, but we also need to explain the cause of the variation in the index in order to improve the stability of the process and forecast more accurately. To understand the variation sources, we calculate the individual component variances of the weighted data, and then do a Pareto analysis of the contributors as a percentage of the total variance. The Pareto chart is shown below.



We see a different and important aspect emerge. It appears that the days open for severity one bugs contributes the bulk of variation to the process. Looking at the graphed values, we see that the weighted days open varies from 22 to 198.

However, the perplexing situation occurs when we see the days open in one week changing dramatically to the next week. For example, in the third week the average unweighted days open goes from 11 days to 98 days in the fourth week. This behavior is puzzling given the fact that there are only seven days in a week.

Further investigation reveals a very common practice used in the index tracking systems, and that is to measure only those bugs that are closed in a given week and not the time on the bugs that remain open. By changing to a system that records the running time on all open bugs (since a bug can not be fixed in a time less than the time it has been open), the variation in the index can be dramatically reduced. The result will be better stability and improved predictability. This analysis of variance approach can be used effectively to improve the quality of metrics reporting.

(Presented October 1999 at ASQ Software Division's 9th International Conference on Software Quality, Boston, MA)