

24<sup>th</sup> Annual Software Engineering Workshop  
Dec 1-2, Goddard Space Flight Center

Quantitative Methods *Do* Work

Edward F. Weller  
Fellow, Software Process  
Bull HN Information Systems  
13430 N. Black Canyon  
Phoenix, AZ 85029

Tele: (602) 862-4563  
Fax: (602) 862-4288  
e-Mail: e.weller@bull.com

## **Quantitative Methods Do Work**

Quantitative methods, including statistical process control, can be effective tools for predicting and evaluating product quality during development and test. The data analysis and conclusions from applications of quantitative methods, including statistical process control, to two projects that were major components of a software release to Bull HN Information System's GCOS 8 Operating System, will show how these techniques were effective and useful. During development and test, we used the release quality predictions as one of the project metrics. We found that analysis of inspection and test results using SPC techniques helped us predict (perhaps understand is a better word) the release quality and the development processes controlling the release quality. We were able to answer the question "Can we ship this product?" with data rather than guesswork.

Inspections have been used in GCOS 8 development since 1990<sup>1</sup>. The process is stable and provides data used by project management<sup>2</sup>. Our goal in the current release was to use defect density during development and test as input to predicting the post ship product quality with reasonable assurance. We are aware of the problems with using defects to predict failures (Adams<sup>3</sup>, Fenton and Pfleeger<sup>4</sup>), but in the absence of other data or usage based testing results, this was what we had to evaluate release quality.

### **Prediction: Stable versus Unstable Processes**

Predicting the future behavior of a process cannot be done unless the process itself is stable. This is a reason for using statistical methods. A variety of techniques can be used to evaluate the underlying process stability. Control charts can be used to calculate upper and lower control limits (UCL and LCL). Processes that stay within limits and do not exhibit other indications of lack of control can be assumed to be "controlled processes". This implies several things about the process:

- Past performance can be used to predict future performance within the control limits
- Process capability relative to a customer specification can be determined

### **Estimating Defect Injection**

Previous inspection process and product data were evaluated and estimates were made for:

- Defect injection rates
- Defect removal rates (inspection effectiveness<sup>1</sup>)
- Defects entering unit test

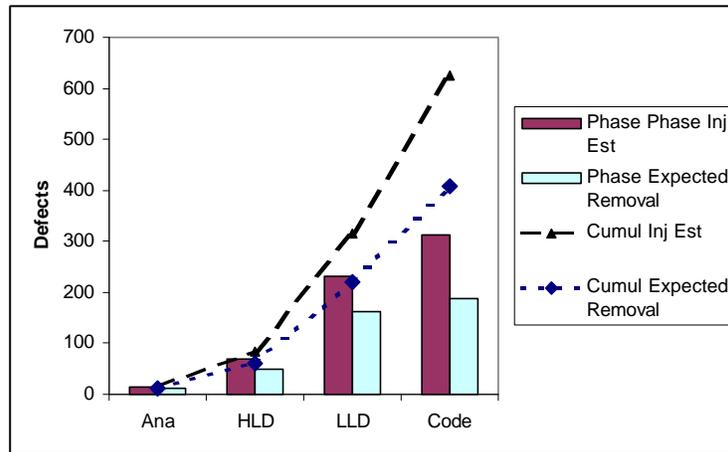
Prior inspection, test, and post ship defect history was used to estimate the defect injection rate. This is a potential area for applying SPC. With enough data, you can establish ranges for defect injection rates, accuracy of size estimates, and inspection

---

<sup>1</sup> Inspection effectiveness is the percentage of major defects removed in each inspection phase, or total defects removed in inspections, divided by the total number of defects in the product at the time of the inspection. Since the total number of defects discovered is never known until a product is retired from use, effectiveness is always an estimate, but one that changes very slightly after a product is shipped, assuming reasonable post ship quality levels.

removal effectiveness. For these projects we did not have sufficient data samples to do this, so we based our estimates on specific product and project history.

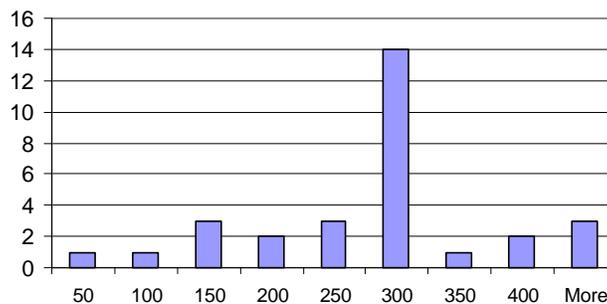
The size, defect injection rates, and prior inspection data were used to develop a defect injection and removal profile.



**Figure 1 – Initial Defect Injection and Removal Estimate**

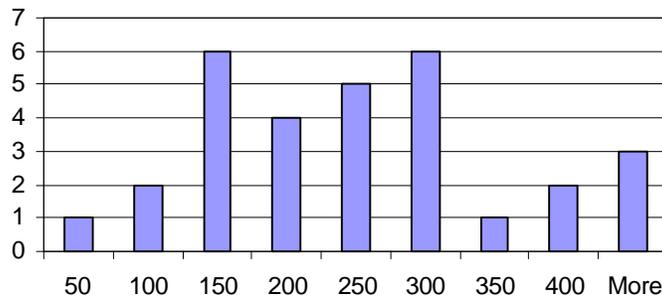
### Inspection Data Analysis

On these two projects the first opportunity to apply SPC was during code inspections. On one project, the work was divided into two parts; the creation of a product feature, and the revision of existing code. A histogram of preparation rates in lines of code per hour is shown in Figure 2.



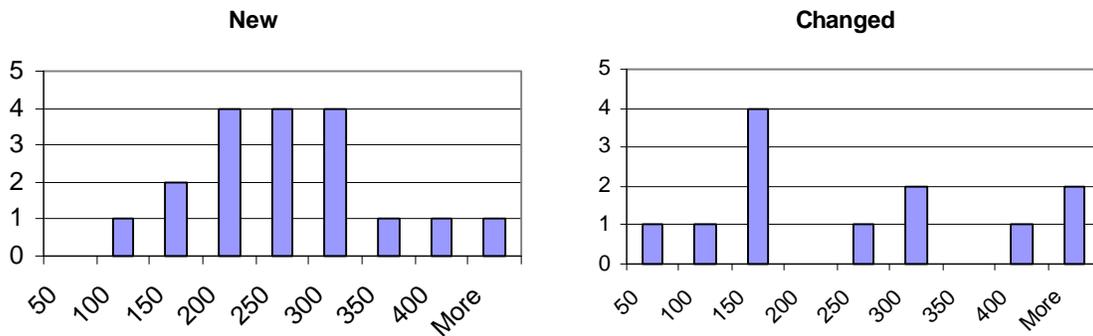
**Figure 2 – Preparation Rate Histogram**

Outliers were examined and eliminated when special causes of variation were discovered. I also compared the preparation rate distribution to the inspection rate, shown in Figure 3.



**Figure 3 – Inspection Rates for 30 inspections**

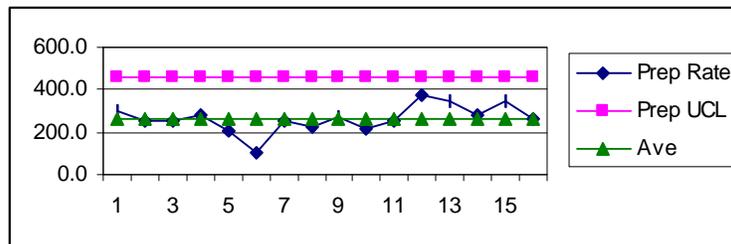
This bimodal distribution was caused by two types of code, “new”, and changed. Figure 4 shows these 2 classes separately.



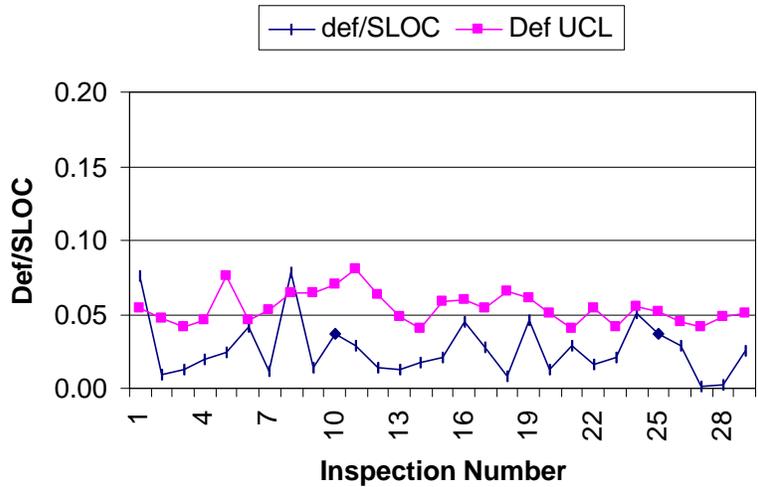
**Figure 4 – New vs Changed Inspection Rates**

I expect new code inspections to be “better behaved” than changes to existing code. (Many inspections of modified code are small in size, causing preparation and inspection rates to have a larger variance. Knowledge of the changed (old) code inspected may also have a wider variance than the new code). The separate views in Figure 4 are typical of much of the inspection data I investigate. The new code approximates a normal distribution as closely as you may see with real data.

A control chart for this data with special causes removed showed a well controlled process:



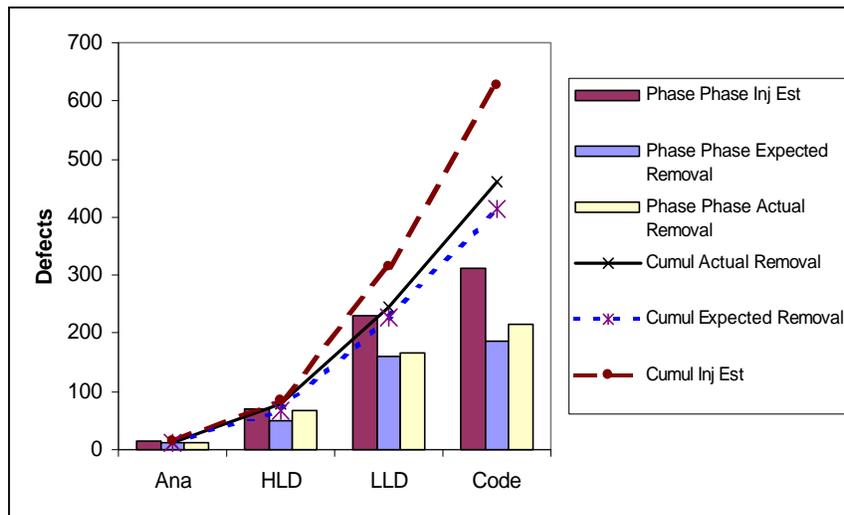
**Figure 5 – Preparation Rate with Outliers Removed**



**Figure 6 – Defect Density Control Chart<sup>II</sup>**

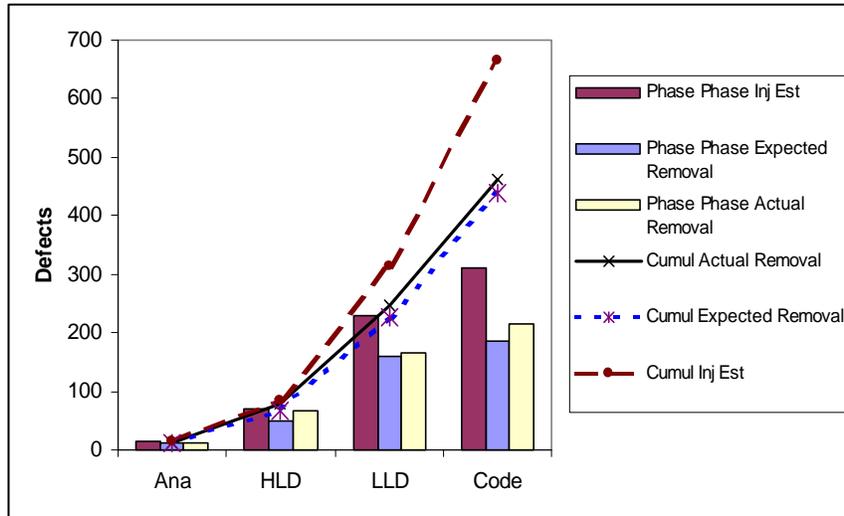
What have we learned about this product and its contribution to the system release? With two exceptions, the inspection process seems to be well controlled. The outliers were investigated (as were other inspection meetings) to understand how well the inspection process was performed. In this case, the outliers were for inspections of changed code, so these outliers were evaluated as caused an assignable cause, and the defect data was within control limits.

Once we were reasonably confident the inspection process was controlled, we developed defect depletion curves for the projects and the system release.



**Figure 7 – Defect Depletion at End of Code Inspection**

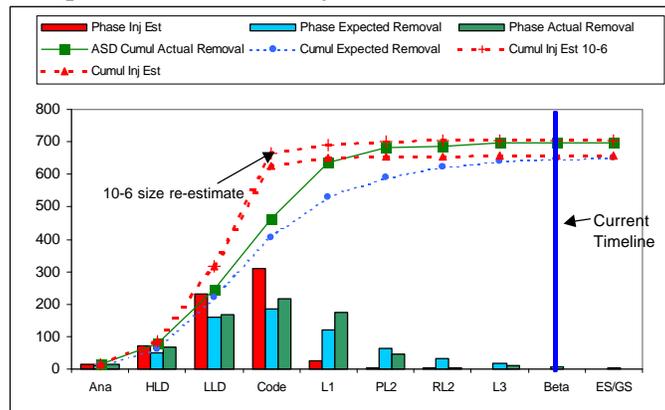
<sup>II</sup> The lower control limits cannot be less than zero, although for convenience the LCL was plotted on this chart as calculated. Once you verify the data is above the LCL, for possible values of LCL, it is probably better to delete this line from the chart.



**Figure 8 – Replotted Defect Depletion with New Size Estimate**

### Unit and Integration Test

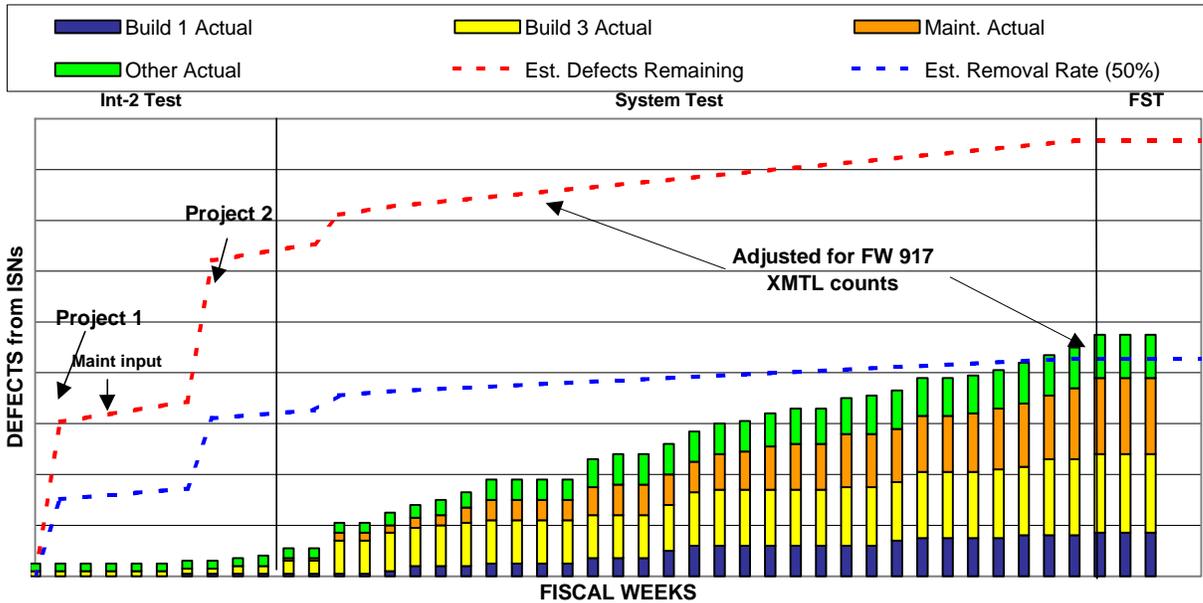
Both projects kept accurate records of defects found during Unit and Integration test. Both projects developed test objective matrices and developed test plans and specifications, so we had some expectation to remove defects more effectively than the 30-50% “norm” often quoted in the industry.



**Figure 9 – Project One Defect Depletion**

Figure 9 shows project one, as it was about to enter System Test (this chart is used in our monthly project review as well as the weekly team meetings). It shows the re-estimate for the number of defects injected. Note the defect removal in Unit Test was higher than estimated and that subsequently in the two phases of Integration Test a small number of defects were removed. Without accurate defect removal data from Unit Test these low numbers would be of more concern with respect to product quality. The Current Timeline is indicated to show the furthest stage where the project defect removal is happening.

This analysis continued through System Test as shown in Figure 10.



**Figure 10 – System Test**

### Conclusions

You should ask two questions about any metric or analysis technique:

- Is it *useful*? Does it provide information that helps make decisions?
- Is it *useable*? Can we reasonably collect the data and do the analysis?

We found that the knowledge we gained about product quality and the processes used to develop these products gave a definite “Yes” to both these questions.

<sup>1</sup> E.Weller, “Lessons Learned from 3 Years of Inspection Data”, *IEEE Software*, Sept 1993

<sup>2</sup> E.Weller, “Using Metrics to Manage Software Projects”, *IEEE Computer*, Sept 1994

<sup>3</sup> E. Adams, “Optimizing Preventive Service of Software Products”, *IBM Journal of Research & Development*, Jan 1984

<sup>4</sup> N. Fenton and S. Pfleeger, *Software Metrics*, PWS Publishing Company, 1997, pp 344-348