Updating the TSP Quality Plan Using Monte Carlo Simulation

David R. Webb
309th Software Maintenance Group

The 309th Software Maintenance Group at Hill AFB has started implementing an updated version of the TSP quality plan utilizing Monte Carlo simulation. This article presents an overview of why an updated quality plan with variability is needed, what data the model requires to be useful, and how the new model works. Actual data from Hill AFB projects that have implemented this method are presented for review.

The TSP quality plan is composed during meeting 5 of the launch[1] by determining the defect injection rates and yields for each phase of the product development process. Using the team's historical averages for these rates and estimated hours per phase, the team can predict how many defects will likely be injected and removed as products move through this process. Unfortunately, these averages do not take into account normal variability in the process. However, by applying a Monte Carlo simulation to the standard TSP quality planning process, a team can determine the historical distribution of process variability and produce a plan with ranges for expected defects injected and removed, as well as a *measure of goodness* for the product and process.

The TSP Quality Plan

One of the hallmarks of projects using the TSP is the attention to quality or, more accurately, the ability to manage product defects. In fact, TSP creator Watts S. Humphrey says:

> ... defect management must be a top priority, because the defect content of the product will largely determine your ability to develop that product on a predictable schedule and for its planned costs. [1]

A chief component of this focus is the quality plan developed during meeting 5 of the TSP launch (for a project). This plan is composed by estimating defects injected and removed during the various phases of the software process. The team uses historical averages of defects injected per hour to determine defects injected and similar averages for yield (the percent of existing defects found and fixed during a phase) to determine those removed (see Table 1 for a sample quality plan). According to Humphrey, the true purpose of the quality plan "is to establish team yield goals for each process step" [2]. If the team does not have sufficient historical data, average injection and removal data collected by SEI can be employed. Using this approach, the team estimates final product quality and then determines whether or not that quality will meet their customer, management, and team goals. If those goals are not met, the team decides what process changes should be made to meet them.

Once the plan has been developed and the launch completed, it is the role of the team's quality manager (assigned during the launch) to monitor progress against the quality plan. Results of the monitoring activities are discussed during the team's weekly meeting. In addition to monitoring actual values for defects injected and removed, the quality manager can help focus the team on quality issues by examining other metrics, such as the defect removal profile (the defects per thousand lines of code removed from software components as they move through the development life cycle) and the product quality index. Exercises such as the capture-recapture method[2] can even predict how many defects may have escaped a personal review or inspection. When done properly, these measures, metrics, and activities can improve the team's quality focus, reducing rework and improving on-time and within-budget performance.

Many TSP teams that have no issues with most TSP concepts struggle with this progress monitoring. While teams are excited about producing the quality plan during launch, the quality manager no longer reports quality progress after a few weeks—other than announcing when the next quality event (inspection, test, etc.) will take place. Let's say, at the project post-mortem, that a team dutifully collects the quality data needed for the next launch, but notes in the lessons learned that they "need to do a better job on the quality plan in the future." In my experience, there are few key reasons for this fall-off of the quality focus:

- The team has not collected sufficient historical data for defect injection and removal; they utilize the by-the-book numbers provided by the SEI, but do not really believe them because they are not their numbers.
- Historical averages blend the results of high performers with average or low performers. Depending upon who is working on a module or series of modules, the predictions may or may not truly represent the work being done, so the team doesn't trust them—and certainly does not use the predictions to guide their work.
- Defect injection rates (DIRs) are based upon the effort estimate for each module; while TSP teams are great at using Earned Value techniques to balance workloads to meet their estimates, not every module is accurately estimated, making the defect injection numbers suspect.
- Team members are not consistently collecting defect data; either individuals are counting defects differently or they are not measuring them at all, making any defect prediction model inaccurate, and thus, unusable.
- When actual data begins to come in, the quality manager, team leader, and sometimes even the coach don't really know what to make of it (e.g., does a lower number of defects than expected mean the team is just very good, or that the quality activity was badly executed?).

These issues can be addressed by two basic practices: 1) consistently collecting data; and 2) properly using the concepts of variability in developing and tracking the quality plan. What follows is an examination of some simple ways to ensure quality data are consistently and properly collected, and a discussion of how to use Monte Carlo simulation to account for inherent process variability—in turn making the quality plan more accurate and usable.

Consistent Data Collection

From an examination of the data of 10 randomly selected PSP students from various classes over a five-year period, it becomes obvious that the rate of defects

injected per hour varies widely by person (averaging 0-60 per hour); even the plots of the averages of defect injection rates in design (averaging from 0-30 per hour) and code (averaging from 2-10 per hour) show that every person is different—sometimes vastly different.

While some of this variability has to do with individual capabilities, the programming environment used, the difficulty of the assignment, and personal coding styles, much of it also has to do with common operational definitions and recording practices. Anyone who has taught a PSP class has noticed that not everyone fills out their defect logs the same way: Some students record several missing semi-colons as a single defect then fix them all at once, while others count each semi-colon as an individual defect with distinct fix times. Most instructors allow this individual style of defect logging, as long as the student is consistent in the method used; however, when determining team defect injection rates, this kind of instability in definitions and recording methods can cause a prediction model to behave erratically. This leads the observer to doubt the validity of using personal defect logs, unless all engineers are somehow coerced into using identical logging techniques.

Another reason to suspect that personal defect data may not be the best fit for a quality prediction model can be seen in the actual project data. The distributions in personal defect logs were collected over an 18-month period from a TSP team at Hill AFB. During this project's execution, the variability in personal defect logging noted in the classroom data did not stabilize or become more consistent. The most disturbing trend in these data is the severe lack of personally recorded data, as evidenced by the number of engineers with data from only one module or no defects logged at all. It is important to note that these data come from a team with strong coaching and a heavy quality focus (they have never released a major defect).

For these reasons, it appears to be undesirable to use personal defect log data for defect injection analyses. That being the case, the question becomes: What kinds of data would make sense? Interviews with the engineers on the noted project (as well as other TSP projects at Hill) suggest that more consistency may be found in defect data from inspection and test databases. These public databases require more strict control to ensure that defects are properly identified, analyzed, addressed, and tracked. This typically requires users to enter data according to a defined procedure and to use common definitions for defects and defect types. This kind of control seems to drive more stable operational definitions and data recording practices than evidenced in the personal defect logs.

When looking at the design and code inspection data from our TSP project, it shows that the distributions are much tighter than those in the personal logs without the problem of a lack of recorded data. That being said, there is still some variability in the data—in this case, higher in the code inspections than the design inspections. For example, the average DIR on both the design and code review data is toward the lower end of the distribution, suggesting a skewed normal or lognormal distribution in defect injection rates.

Therefore, a possible conclusion of this analysis is that personal defect log data is not as useful in creating a quality model for the quality plan as is data from public databases, such as the inspection and test databases. However, even in these data, the defect injection rates display a certain amount of variability that should be accounted for in our quality model.

One very important note here is that this analysis should not be used to suggest or validate the idea that personal defect logs are not useful. Several engineers interviewed found them very useful for personal improvement—they simply are not consistent from person to person, making the data unusable for team modeling purposes. Strict coaching and quality manager oversight, focusing on common operational definitions and recording procedures, may make these data more usable.

TSP (v1) Rollup Plan Summary Quality Summary

Defect Density

Defects/KLOC	Plan	Actual
Detailed Design Review	164	
Detailed Design Inspection	49.1	
Code Review	395	
Compile	87.9	
Code Inspection	61.6	
Unit Test	31.1	
Build and Integration Test	2.76	
System Test	0.55	
Total Development	1038	
Total	1.04	

Inspection/Review Rates

	Plan	Actual
Code Review	28.5	
Code Inspection	5.51	

Cost of Quality (COQ)

	Plan	Actual
Percent Appraisal COQ	32.70%	
Percent Failure COQ	4.69%	
Appraisal/Failure Ratio	6.98	

Phase Yields

	Plan	Actual
Requirements Review	70%	
Requirements Inspection	70%	
High-Level Design (HLD) Review	70%	
HLD Inspection	70%	
Detailed Design Review	70%	
Code Review	70%	
Compile	50%	
Code Inspection	70%	
Unit Test	90%	
Build and Integration Test	80%	
System Test	80%	

Defect Injection Rates

Defects Injected per Hour	Plan	Actual
Requirements	0.25	
HLD	0.25	
Detailed Design	0.75	
Code	2	
Compile	0.3	
Unit Test	0.07	

Table 1: *Sample TSP Quality Plan Created During Meeting 5*

Monte Carlo Simulation

One method of taking into account the variability of the defect injection rates and yields in a quality model would be using a technique called Monte Carlo simulation. The Monte Carlo method is any technique using random numbers and probability distributions to solve problems [3, 4], using the brute force of computational power to overcome situations where solving a problem analytically would be difficult. Monte Carlo simulation iteratively uses the Monte Carlo method many hundreds or even thousands of times to determine an expected solution.

The basic steps of Monte Carlo are as follows:

1. Create a parametric model.
2. Generate random inputs.
3. Evaluate the model and store the results.
4. Repeat steps 2 and 3 many, many times.
5. Analyze the results of the runs.

This is useful in creating a form of prediction interval around an estimate. For example, assume the number of defects in a software product (in the design phase of development) can be predicted by multiplying the historical defects injected per hour by the number of hours estimated for the phase. We can improve that estimate by using the ratio of historically estimated hours to actual hours, known as the Cost Productivity Index (CPI). The CPI

Continued on Page 18

Continued from Page 15

represents how well tasks have been estimated in the past; a number near 1 means that estimates have been fairly accurate in the past; a number greater than 1 tells us that we tend to overestimate; a number less than 1 says we typically underestimate our tasks. Dividing the estimated hours by the CPI will compensate for any tendencies to over- or underestimate. Thus, our final prediction equation for design defects injected is the DIR for design multiplied by the number of estimated hours in the design phase, divided by the CPI for design. This is the parametric model needed for step 1 of the simulation:

$$d = DIR_{design} \times Hours_{design} \div CPI_{design}$$

In step 2, we need to generate random inputs to the DIR and CPI variables of the equation, since these are parameters that are subject to variability in our historical data[3]. The question is: Where do we get these random values from? The answer can be found by examining each of the variables. For example, the typical TSP approach to estimating design defects would be to use the average historical values for the DIR and CPI, as defined in Table 2. The only problem with that approach is that, while the average DIR in design is 2.1, it can vary from 1 to 5, in a lognormal fashion. Additionally, the historical data in Table 2 shows that the average CPI for design is 1, but it varies from 0.5 to 1.5 according to a normal curve. With this in mind, we would use these distributions to generate our random input data for step 2 of the Monte Carlo process. Having estimated that 8.3 hours will be spent in design, we randomly select values from each of these distributions, choosing 0.88 defects per hour for the DIR and a value of 1.12 for the CPI. Therefore:

$$d = 0.88_{defects/hour} \times 8.3_{hours} \div 1.12 = 6.52_{defects}$$

This gives us the value of 6.52 defects, which is how we evaluate the model and store the results for step 3 of the process.

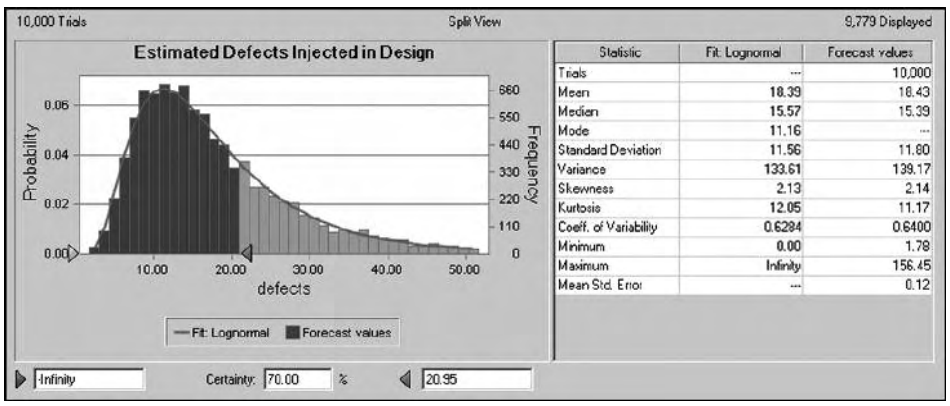
Step 4 of the Monte Carlo process simply requires repeating steps 2 and 3 many, many times—each time storing away the newly generated answers. Let's say we do 10,000 of these calculations and store them all away; when complete, we will have built up a new distribution for "d", the results of the equation.

Step 5 of this process is examining the distribution of the results to determine what we can learn. In Figure 1, we can see that the answers from our equation using the Monte Carlo process fall into a lognormal distribution, with a mean of 18.39 defects and a standard deviation of 11.56. Further analysis of the data suggest that 70 percent of the time, we should expect no more than about 21 defects will be injected in the design phase of our process. This provides us a bit more insight than we would see in a typical TSP quality plan. For instance, we now know that if there are fewer than 21 design defects found during our project, it's not necessarily a bad thing; however, if we find more than this, say 40 defects, something may be out of the ordinary (since that happens rarely). If we find many more than 21 defects—200, for example—then we can be pretty certain we have an issue that needs to be addressed. The wonderful thing about this is that we can determine these parameters at planning—a concept that fits well with TSP principles and philosophies.

Table 2: *DIR and CPI Notional Historical Data*

Project	DIR-Design	CPI-Design
P1	1.02	0.50
P2	1.33	1.15
P3	2.06	0.67
P4	1.13	0.88
P5	5.00	0.96
P6	2.50	1.35
P7	1.30	1.50
P8	4.10	0.62
P9	3.20	1.50
P10	1.08	1.38
P11	1.00	0.98
P12	1.62	0.89
P13	1.88	0.78
P14	3.10	0.88
P15	1.23	0.92
Average	2.10	1.00

Figure 1: *Sample Distribution of Results from Monte Carlo Simulation of Defects Injected in Design*



Using Monte Carlo Simulation for the TSP Quality Plan

There are essentially five steps in modifying a TSP quality plan to take advantage of the previously described Monte Carlo simulation techniques:

1. Gather historical data and determine distributions for the DIR, yield, and CPI.
2. Modify the equations that determine defect injection, defect removal, defects remaining, and any other metrics important to the team.
3. Run the Monte Carlo simulation using estimates for hours per process phase and the distributions for the DIR, yield, and CPI.
4. Examine the results, determine how well project goals are addressed, and come up with next steps for the project.
5. Use this plan to guide and track the project's quality progress.

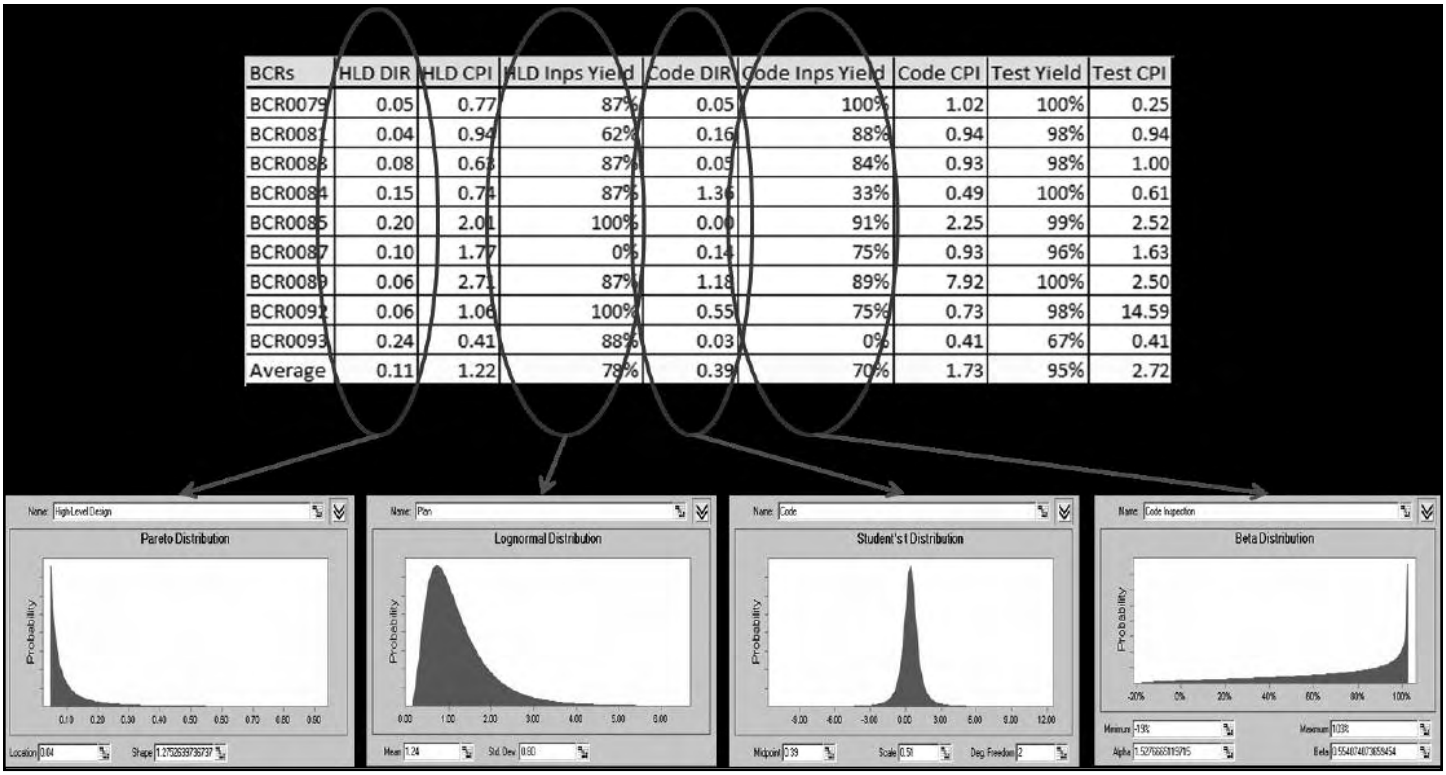
Gathering Historical Data and Determining Distributions

The first step is fairly straightforward for TSP projects that have been using the process for a while and have post-mortem data available. The team simply needs to gather data on the DIR, yield, and CPI for a number of past projects to determine the actual distributions of data. This can be done on a project-by-project basis, or by module, capability, or build (as desired). In Figure 2, the actual data from a Hill AFB project are listed as Baseline Change Requests (BCRs) and represent code changes made to an existing software baseline over 18 months. In this example, the team used Oracle Crystal Ball (a spreadsheet-based application suite for predictive modeling) to determine the distributions of each set of data.

Once the data gathering and analysis have been done, the team must determine the quality planning parameters[4], as shown in Table 3.

Modifying the Equations

Currently, the TSP quality plan predicts



BCRs	HLD DIR	HLD CPI	HLD Inps Yield	Code DIR	Code Inps Yield	Code CPI	Test Yield	Test CPI
BCR0079	0.05	0.77	87%	0.05	100%	1.02	100%	0.25
BCR0081	0.04	0.94	62%	0.16	88%	0.94	98%	0.94
BCR0083	0.08	0.63	87%	0.05	84%	0.93	98%	1.00
BCR0084	0.15	0.74	87%	1.36	33%	0.49	100%	0.61
BCR0085	0.20	2.01	100%	0.00	91%	2.25	99%	2.52
BCR0087	0.10	1.77	0%	0.14	75%	0.93	96%	1.63
BCR0089	0.06	2.71	87%	1.18	89%	7.92	100%	2.50
BCR0092	0.06	1.06	100%	0.55	75%	0.73	98%	14.59
BCR0093	0.24	0.41	88%	0.03	0%	0.41	67%	0.41
Average	0.11	1.22	78%	0.39	70%	1.73	95%	2.72

Figure 2: *Historical Data with Distributions*

measures that are useful in the planning stages of the project and can be used to guide the engineers during project execution. Some of these measures include defect densities per phase of review/inspection, review rates, and appraisal-to-failure ratios. In crafting the new quality plan, we can now be more specific and predict the expected number of found defects during each quality phase and how many defects are remaining in the product, with a prediction interval. The equations for doing this are a modification of the equation previously created, predicting how many defects will be injected in the design phase. Using this formula, we simply multiply by the planned yield of the inspection phase to estimate how many defects will be removed[5]:

$$d_{\text{design inspection}} = DIR_{\text{design}} \times HOURS_{\text{design}} \div CPI_{\text{design}} \times YIELD_{\text{design inspection}}$$

Similar equations can be generated for every phase, based upon the historical data from Figure 2. We can then use these equations, along with the distributions identified, to determine the results for our Monte Carlo simulations, as shown in the estimated defects portion of Table 3[6].

Running Monte Carlo Simulation and Examining the Results

At this point, during meeting 5 of the TSP launch, the Monte Carlo simulation is run with the variable inputs and the prediction equations. The simulation can create distributions of results for all 14 predictions highlighted in Table 3. The team can predict, for example, the minimum number of defects they would expect to find in each inspection phase, within a given prediction range (e.g., 70 percent of the time). In this case, the total number of defects found in detailed design inspection should be at least 456, and 633 in code inspection, 70 percent of the time, according to historical data.

To make this prediction even more useful, the team should run Monte Carlo simulation for each module following launch meeting 6. At this point in the TSP launch process, bottom-up plans have been made and hours have been estimated for each process phase of every module in the next-phase plan. Assuming every individual performs within the parameters established from the team data, the Monte Carlo simulation can now be run for each module. Table 4 (see next page), for example, shows a single BCR update to a software baseline, with its own design and code inspection predictions. Note how the numbers are much lower for this single update than for the combined numbers of the entire project update. When the Monte Carlo simulation is run for these planning numbers, the charts look similar. However, the key advantage is that we can now predict that 70 percent of the time the design inspection for this update

Table 3: *Sample Planning Parameters for a New Quality Plan*

TSP Quality Plan with Monte Carlo

Estimated Time	Plan	Actual
High-Level Design	434.93	
High-Level Design Inspection	147.28	
Code	901.87	
Code Inspection	175.82	
Unit Test	275.87	

Defects Injected per Hour	Plan	Actual
High-Level Design	0.11	
Code	0.39	

Project CPI	Plan	Actual
High-Level Design	1.22	
Code	1.73	
Unit Test	2.72	

Phase Yields	Plan	Actual
High-Level Design Inspection	78%	
Code Inspection	70%	
Unit Test	95%	

Estimated Defects Found	Plan	Actual
High-Level Design Inspection	30.61	
Code Inspection	143.83	
Unit Test	65.82	

Estimated Defects Remaining After	Plan	Actual
High-Level Design Inspection	8.85	
Code Inspection	154.21	
Unit Test	88.39	

Software Defense Application

The software defense community will benefit from utilizing the proposed TSP quality plan update, as this article shows how to determine and apply variability into the plan through Monte Carlo simulation. Users will be able to predict product and process quality at stages throughout the life cycle and at delivery. It will also help in meeting requirements for Quantitative Project Management and Organizational Process Performance at CMMI Level 4. These methods closely track product and process quality, providing tools for project managers in avoiding cost and schedule pitfalls—and in delivering near zero-defect products.

should find at least three defects (although it would not be unusual for the code review to find zero). This gives us some indication of the goodness of the inspections and a lower limit that we can look for during the execution of the project. Likewise, in the unit test for this change, we should find no more than 12 defects, 70 percent of the time (see Figure 3). In this case, we look for the upper limit, since our goals are to find more defects in inspections than in testing.

Guiding and Tracking Project Progress

Once the TSP launch is complete and the plans are approved by management, the team uses these plans to guide their work. The team also checks progress against the plans during their weekly meetings. The quality manager, for example, reports on the current defect injection rates and yields for modules complete to date. He or she also provides feedback on the current product quality index, defect removal profile, and so forth (as shown in Table 1).

Figure 3: *Estimated Maximum Defects Found in a Unit Test for a Single BCR*

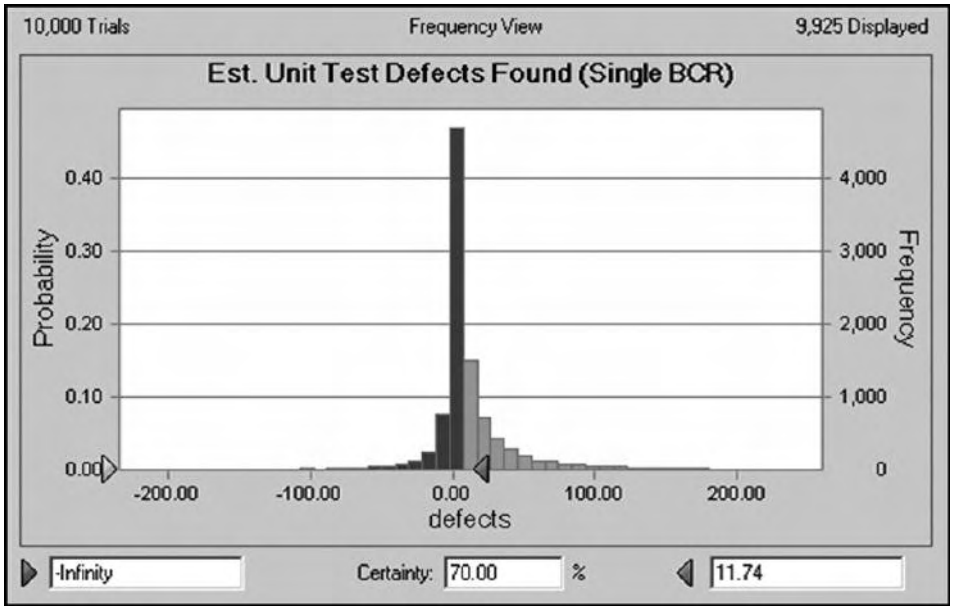


Table 4: *Sample TSP Quality Plan for a Single Update*

TSP Quality Plan with Monte Carlo (Single BCR)

Est. Time	Plan	Actual
High-Level Design	84	
High-Level Design Inspection	21	
Code	62	
Code Inspection	25	
Unit Test	14	

Phase Yields	Plan	Actual
High-Level Design Inspection	78%	
Code Inspection	70%	
Unit Test	95%	

Defects Injected per Hour	Plan	Actual
High-Level Design	0.11	
Code	0.39	

Est. Defects Found	Plan	Actual
High-Level Design Inspection	5.91	
Code Inspection	9.89	
Unit Test	5.57	

Project CPI	Plan	Actual
High-Level Design	1.22	
Code	1.73	
Unit Test	2.72	

Est. Defects Remaining After	Plan	Actual
High-Level Design Inspection	1.71	
Code Inspection	11.70	
Unit Test	6.13	

With the new Monte Carlo-generated quality plan, the quality manager has additional information to present at the weekly meetings. For example, he or she could present how many defects have actually been found in inspection or test activities—versus those predicted by the model. Another new metric is an updated estimate of the predicted defects remaining, easily calculated taking the estimates for defects injected and subtracting the estimates for defects removed. Once actual project quality data begins to come in, these models can be used again—this time replacing the estimated values with actual values and rerunning the simulation. This provides a new prediction for defects remaining that can be tracked throughout the project duration.

It is important to point out that this new way of examining and predicting the quality of the product in no way supplants those currently being used by TSP projects. This is simply one more weapon to add to the quality arsenal.

Summary

A TSP quality plan is a very effective way of focusing a team on the tracking and resolution of defects early in the project life cycle. However, the current version of the plan does not take into account variability. Applying Monte Carlo simulation to data already being collected by TSP teams provides a more robust insight into the quality processes TSP teams employ. It also gives further insight into what can be expected in terms of product and process quality. The TSP teams at Hill AFB recently started using this technique and are still gathering data on its usefulness.◆

References

1. Humphrey, Watts S. *TSP – Leading a Development Team*. Upper Saddle River, NJ: Addison-Wesley, 2006. Page 138.
2. Humphrey, Watts S. *TSP – Leading a Development Team*. Upper Saddle River, NJ: Addison-Wesley, 2006. Page 87.
3. Weisstein, Eric W. "Monte Carlo Method." *Wolfram MathWorld*. <http://mathworld.wolfram.com/MonteCarloMethod.html>.
4. Wittwer, J.W. "Monte Carlo Simulation Basics." *Vertex42*. 1 June 2004 <http://vertex42.com/ExcelArticles/mc/MonteCarloSimulation.html>.

Notes

1. The best resource to learn about TSP's numbered meetings and quality plans is Watts S. Humphrey's Nov. 2000 report "The Team Software Process." Section 7.1 discusses quality plans. See

<www.sei.cmu.edu/reports/00tr023.pdf>.

2. For more on this method, see <www.stsc.hill.af.mil/CrossTalk/2007/08/0708Schofield.html>.
3. Let us assume here that we determined hours earlier via Proxy-Based Estimation (PROBE) or other estimating model.
4. Don't be confused by the values you see in the shaded cells. Each of the highlighted cells for defects injected per hour, CPI, and yield in Table 2 initially contain an average value, similar to the current TSP quality plan; however, this value is replaced by the tool with random values from the distributions in Figure 2 when the Monte Carlo simulation is run.
5. In this situation, yield must be a decimal number between 0 and 1 instead of 0 and 100 percent.
6. The highlighted cells for "estimated defects found" and "estimated defects remaining after" in this table show the results of the parametric equations using the average values; these are replaced with the results of the calculations using random values from the distributions, during the Monte Carlo simulation.

About the Author

David R. Webb is a Technical Director for the 520th Software Maintenance Squadron of the 309th Software Maintenance Group at Hill AFB, Utah. Webb is a project management and process improvement specialist with 22 years of technical, program management, and process improvement experience in Air Force software. Webb is an SEI-authorized PSP instructor, a TSP launch coach, and has worked as an Air Force section chief, software engineering process group member, systems software engineer, and test engineer. He is a frequent contributor to technical journals and symposiums, and holds a bachelor's degree in electrical and computer engineering from Brigham Young University.

7278 4th ST
BLDG 100
Hill AFB, UT 84056
Phone: (801) 586-9330
E-mail: david.webb@hill.af.mil