

# Software Quality Data

## Part 3: Quality Control Using In-Process Data

### Abstract

We measure, quantify and report on software quality. But can we control it? Can we actually **assure** quality (as opposed to just measuring it)? This is the third of three papers in which we will learn how we can go beyond just quantifying the quality of the software after it has been built.

In “Part 1 Basic and Derived Metrics”, we discussed the three basic metrics of software quality and four important metrics that can be derived from those basic three.

In “Part 2: Quantitative Quality Planning”, we looked at how to use this information to produce a Quality Plan that we can use to understand our quality performance *before* the project is complete.

Now, in “Part 3: Quality Control Using In-Process Data” we will discuss:

- Using in-process metrics to track against our quality plan, and
- Taking corrective action when the in-process data suggest it.

### Introduction

Those of us who are charged with assuring the quality of the software that our organizations produce have quite a challenge. We are being asked to manage something that is difficult to quantify, and even harder to control. We are familiar with a variety of activities we could engage in that would be likely to assure the quality of our software, and we do what we can to be sure that those activities are done.

But activity is not sufficient. Indeed, there are far more activities that we could engage in than there is time to do so. So we must choose how we will spend our available time and effort; what we will do, and to what extent. Making these choices requires that we have a clear picture of what we are trying to achieve. Not the nebulous objective of “good quality”, or even “better” than last time. We need to set specific quality goals for each project: goals that we can use to direct our activities. And finally, we need ways of understanding how we are performing against those goals early enough in the project to allow us to take corrective action.

The key to all of these things; the key to our ability to actually *manage* quality as we are working, is data. By collecting a few carefully focused measures, we can build the understanding we need to be able to set quantitative quality goals, plan to achieve those goals, and take corrective action during the project to make our success more likely. We need to use data to understand:

- How we are spending our time,
- How the activities we engage in are affecting the quality of our software,
- What our past performance has been, and
- What performance we can reasonably expect to achieve on the next project.

# Software Quality Data

## Part 3: Quality Control Using In-Process Data

Although the methods we will explore are rarely used in software projects, they are by no means new. These methods have been used with great success for many decades by many engineering disciplines, as well as manufacturers. Luminaries such as Walter Shewhart and W. Edwards Demming made these topics the subject of their lives' work, and taught the world how to apply them. The Software Engineering Institute (SEI) at Carnegie Mellon University (CMU) has been building our understanding of how these principles apply to software with their work on the Capability Maturity Model for Software (CMM), the Personal Software Process (PSP), and the Team Software Process (TSP). Much of the content of this paper is application of these principles from the SEI.

### Analyzing In-Process Quality Data

Making a good quality plan is only the first step. The real benefits come when we use that plan to actually manage the quality of the software that is being produced while the project is going on. Because of the data we are collecting, we can tell at each step how the project is going, and we can take corrective action if the data show that there may be a problem.

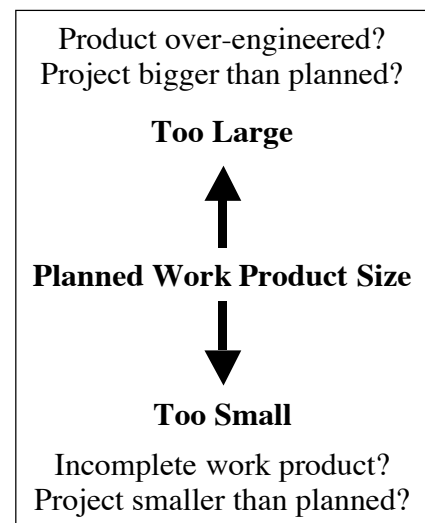
#### *Size of Artifacts*

As each artifact is completed, you will want to compare its actual size to the estimate on which your plan was based. We do this for two reasons.

First, if the size of an item is significantly different from what was estimated, that could indicate a quality problem. For example, if your High-Level Design Specification is only half the size you expected, then it might indicate that some requirements have been overlooked. If it is much bigger than expected, then your entire product may be bigger than you expected – which could cause schedule problems. Any large size variance could be a leading indicator of trouble ahead, so it should be investigated.

The second reason to check actual sizes against your estimates is to adjust your plans, if needed. If the project is legitimately larger or smaller than you planned, it is better to re-plan early than to wait until the project is in trouble later.

Finally, you will need accurate actual sizes for your artifacts to compute some of the other quality metrics you will use to understand your project status.



# Software Quality Data

## Part 3: Quality Control Using In-Process Data

### ***Planned vs. Actual Time in Reviews***

Time spent in a review is a critical measure of the effectiveness of the review. When a review is completed in too little time, that is a sure indication that it was superficial, and not likely to have found the defects it should have found. This is the most common problem with reviews: that people rush through them thinking they are saving time, not realizing that the defects they miss will cost more time to remove when they are found later.

With that understanding, one would be tempted to welcome over-runs in review time. But these could be indicators of trouble as well. The reviewers could be having trouble with the review process, or they may misunderstand what they are expected to do. At best, this would mean wasted time, but it could also result in ineffective reviews.

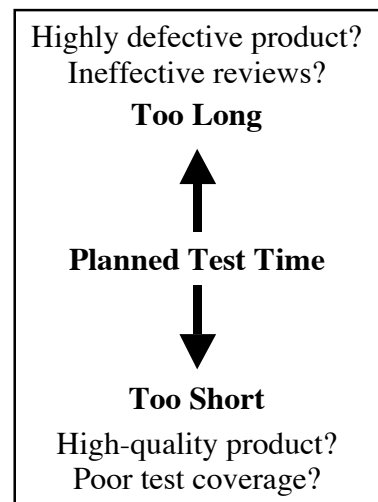
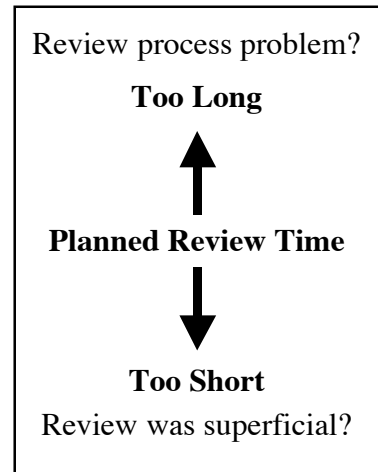
Whether review time is too short or too long, and large variance should be investigated by looking at the number of defects found, as we will discuss below.

### ***Planned vs. Actual Time in Testing***

Like review time, testing time variance can also indicate problems, but the nature of the problems is likely to be different for testing than for reviews.

Too little time in testing is almost always associated with fewer defects than expected being found. Although this *might* be an indicator of superior quality, it more often indicates poor test coverage. You should check the tests that have been run to assure that they are adequate.

Too much time in testing is almost always associated with finding *more* defects than expected. This is a serious problem in almost all cases, as it indicates that the quality of the software that is entering test is lower than planned for. And because test yields tend to be below 50%, finding many defects in test generally means that many defects are escaping the testing phase, and future testing phases will also go on longer than planned. If this is the case, then not only is the project schedule in jeopardy, but the defect content that will be delivered is likely to be much higher than planned for.



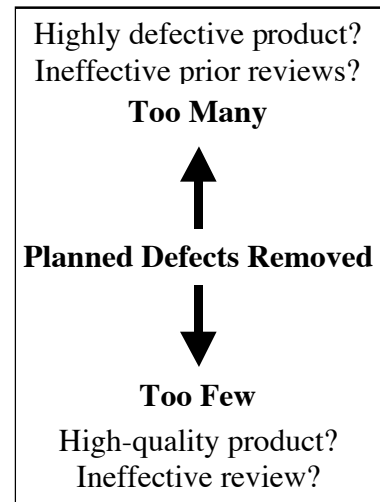
# Software Quality Data

## Part 3: Quality Control Using In-Process Data

### ***Planned vs. Actual Defects Removed***

When a defect removal activity finds fewer defects than expected, the first thing to suspect is the yield (that the defects are in the item but were not found). Since your quality plan is based on your expectations about defects being removed in each phase, this means that there could be more defects remaining in the product than you anticipated. More defects entering later phases (especially the testing phases) can cause serious schedule problems, besides resulting in more defects being delivered with the product.

Finding more defects than you expect to find is rarely a good thing. This indicates that the product has more defects in it, possibly indicating that a prior defect removal phase was ineffective. If this is the case, there could be serious schedule and quality consequences.



### **Taking Corrective Action**

When you find that your in-process data is significantly different from plan, you should immediately investigate as suggested in the preceding sections. Often, you will determine that things are OK despite the variance, or that you need to watch the numbers from future quality activities to see if there really is a problem. But there will be times when the data clearly indicate a problem. In those cases, you have the opportunity to take corrective action to salvage your project goals.

If the project is clearly larger than planned for, you must rework your plans and probably re-negotiate commitments for cost or delivery date. Never hope to make up the difference. A larger project will take more time and effort to complete than the smaller one you planned on. Waiting until later to re-negotiate will make the situation worse by leading the project sponsors to believe things are on plan when they are not.

If you find that a review or a test was ineffective, you have two choices. You can re-do the review or test, or you can plan to catch the remaining defects in later phases of the project. Either of these options will cost the project time and effort. The only question will be which option will be the less painful. Since the earlier activities tend to be more efficient at removing defects than later ones, re-doing the review or test will likely be more efficient. Of course, it may cause political problems, so you will have to weigh the options and choose accordingly.

A common problem that requires corrective action is when an artifact is found to have a much higher defect density than expected. When a review shows that a document or a program has many more defects than planned for, that indicates that the artifact will cause quality problems throughout the project. Because each defect removal activity removes only a percentage of the existing defects, more defects found usually means more defects escaping to future phases.

# **Software Quality Data**

## **Part 3: Quality Control Using In-Process Data**

In some cases, you may find that it is expedient to re-inspect something to try to get the defect rate down to an acceptable level. But there are times when the best use of project resources will involve throwing the artifact away and re-writing it. This is always a hard decision to make, but when you consider all of the time a buggy program might cost in testing, the time to re-write may be justified.

### **Conclusion**

Data is the key to our ability to make actionable quality plans and use them to actually manage quality during a development project. Your organization's own historical data is best, but if it is not available, industry benchmark data will do. Then, after your next project, you will have actual data from your organization to use in planning.

With the necessary data, you can make quantitative plans that include targets for product size, defect content, and effort for defect-removal activities. The data you collect as you are working through the project can then be compared with the quantitative plans so you can see when things are going according to plan, and when corrective action is needed.

The hardest part often comes when corrective action must be taken. This often requires that you take bold counter-intuitive action based on the available data and the principles we have discussed. But with experience, your historical data will become a more and more compelling guide for the difficult situations. And ultimately, you will succeed in managing the quality of your organization's software products.