PS&J SOFTWARE SIX SIGMA

Measured Managed and Controlled Project Performance

Contact us at: 201-947-0150; 201-358-8828; Quality@SoftwareSixSigma.com

MEASUREMENTS AND MANAGEMENT

Implementing a measurement framework is fundamental to software process improvement. After all: "You can't manage what you don't measure." Without reliable measurements, it is difficult to prioritize targets for improvement. Without a performance baseline, it is impossible to assess if improvements have been effective. And without measurements it is impossible to keep a new process at expected performance levels.

The software development measurement framework is extraordinarily simple. All metrics derive from a few basic measurements: development effort, product size, and defects. With a few operational definitions, some simple automation, and a little hands-on training, it's easy to collect these metrics with minimal overhead and negligible measurement errors. Although these implementation requirements are simple, each one is essential and skipping one will lead to a lot of wasted effort:

- Operational definitions contain a procedure for making each measurement. They need to be written down carefully. They need to specify how to use the interface to the automated system for collecting the metrics and they need to be regularly updated to address dealing with unforeseen cases. Without good operational definitions, its very unlikely that data will be taken consistently.
- Good automation is essential to minimizing the cost of measurements. With good automation, the actual cost of data collection can be reduced to a few minutes a day and simple measurement errors virtually eliminated. However, automation is even more important for data analysis than for data recording. Manual systems tend to produce "write-only" data repositories. Information must be accessible in real time. Basic analysis must be available at the click of a button. Any system that requires manual data aggregation and analysis is going to be defect-prone, costly, and underutilized.
- Without formal training, the operational definitions and the automation are useless. The only people that can collect the data are the software engineers. They need to be trained to do it in a consistent way. Its been our experience that "on-the-job" training does not produce the level of consistency required.

There is one more issue that has to be addressed in order to get an effective measurement program in place – motivation. If the developers are not motivated to collect the data they will find measurement burdensome no matter how little time it takes, there will be content problems, and without constant management pressure, they will gradually stop taking data.

In order to deal with the motivation issue, it is necessary to apply a simple principle to the whole measurement program. No one should take any data that they don't directly use. For example, if defect data is being taken to improve inspection checklists, the developers themselves, not a process group or a quality group, should be responsible for revising the checklists and lowering the number of defects found in test. The idea is simple: if you are the one using the data, you will act responsibly about recording it completely and consistently.

Contact us at: 201-947-0150; 201-358-8828; Quality@SoftwareSixSigma.com

We experienced this first hand once with data from an inspection process. We had been collecting defect data for quite a while. Most fields on the data collection form involved making a choice in a combo box, but one field did not. It required a textual description of the defect. After a while we decided to implement a formal defect prevention process and we used the inspection data for training. During the training we discovered that no one could understand the defect descriptions just weeks before. The defect prevention training address an issue we had neglected the motivation for recording the defect description. The defect prevention process required a review of the defect data once a week by the developers. This forced them to keep the quality of the descriptions up when recording the data. Within in a few weeks we had useful high quality defect data.

Conversely, if data is collected by development for a "quality group", you will probably find that developers put just about anything in a field in order to complete the data collection form. The quality and consistency of the data will be low. The "quality group" will notice this, complain, and propose more audits or automated checking to ensure data integrity. Frequently, by the time the data has been "analyzed" and recommendations sent back to development, a month will have gone by, the recommendations will not be read, and their existence irrelevant. Unfortunately, this particular organizational pathology seems to be more the rule than the exception.

So, whenever measurements are introduced into a process, it is essential to ensure that the process also requires that the people that take those measurements personally do something with the data that has an obvious benefit to everyone. For example, if you require the collection of defect data during inspections or test, you need to introduce a defect prevention process that makes the benefit of taking the data obvious to the developer. Telling the developer that the data is very useful for management or process improvement and that is must be collected and submitted to someone else for their use is not an effective substitute!

To be useful for management, measurements need to be complete and consistent. Measurement Systems Evaluation (MSE) is an element of the Six Sigma toolkit that can characterize the quality of a set of measurements. It is a collection of statistical techniques for characterizing measurement error (bias and repeatability). A measuring device will generally introduce an error into the value of a measurement of any physical quantity. MSE is used to characterize these errors.

With the software measurement framework we have just described, the measurement errors in software metrics can be made truly negligible. Inconsistencies and variability are usually just the consequence of failure to meet one of the conditions for implementing an effective measurement framework outlined above. The solution is not extensive MSE, it is fixing the measurement framework.