# Metrics You Can Bet On

by Mike Cohn • 5 Comments • originally published in Better Software on 2006-08-01

I'm a strong proponent of using metrics. They can help us understand how our projects are progressing and how well our software development process is working. I am not a believer, however, in the old adage that we cannot manage what we cannot measure. As Robert Glass pointed out in his book Facts and Fallacies of Software Engineering, we manage what we cannot measure all the time. He gives the example of cancer research. I am sure that cancer researchers manage their work and that they themselves are managed. Yet no one can draw a Scrum-style burndown chart or calculate earned value toward discovering a cure.

I was recently in Las Vegas for the annual Better Software Conference & EXPO. One of the things I learned while there was that casinos have installed monitors at the roulette tables that show the results of the last twenty or so spins of the wheel. The casino's idea was that if they could show that the last four spins of wheel had come up red then gamblers would decide that black was due. Or gamblers would decide that red would continue its streak. The casino would win either way as more money was bet.

A metric such as the results of the last twenty spins of a roulette wheel is enticing. It seems like it should be useful, and it can be presented in a visually appealing manner. Indeed, the casinos display each of the last results in either red or black with results of each number offset to different sides of a display. This makes it easy to see at a glance the relative number of red and black results, as well as any current "trend."

Ultimately, however, this is a useless metric. Each spin of a roulette wheel is independent. Red's having shown up four (or four hundred) consecutive times has no influence on the next spin of the wheel. Most gamblers are smart enough to know this, yet roulette betting is up 30 percent since use of this metric began.

If we can be misled by a metric as obviously flawed as this one, think how far astray some more complicated project metrics may lead us. As an example, consider a team that wishes to measure the rate at which testers are discovering defects in a product that is nearing its release date. Presumably this team will experience a decrease in the rate of defect discovery as the application improves. If the team decides to report hourly on defect discovery data, managers will see a definite drop during the lunch hours when fewer testers are working. They could then misinterpret this to represent an improvement in the application. More realistically, the team could measure at the daily level, in which case they'll see misleading improvements whenever testers take a day off.

In devising or considering metrics for your project, keep in mind these guidelines:

- Measure only what can be measured. I've come across a number of metrics lately that attempt to measure the business value delivered by each small feature added to a project. If the features are too small, measuring the value contributed by each doesn't work. For example, what is the value of adding delete key support to a word processor?
- Measure at the correct level and in the correct units. Measuring the number of defects discovered per hour can be misleading because vastly different amounts of testing occur during different time periods. A better metric would be the number of defects discovered per hour of actual testing time.
- Measure something only if you plan to act on the results. There is a cost to collecting metrics. Don't incur these costs unless you plan to act on the data.

Metrics can be a very useful tool on a project. Choose yours wisely—and remember that it is possible to manage some aspects of a project without metrics.

---

Posted: August 1, 2006

**Tagged:** management, metrics, testing