Blog: Meaningful Metrics

From Michael Bolton at Developsense.
Original post at http://www.developsense.com/blog/2009/01/meaningful-metrics/

Over the years, I can remember working with *exactly one* organization that used my idea of an *excellent* approach to software engineering metrics. Their approach was based on several points:

- They treated metrics as first-order approximations, and recognized that they were *fundamentally* limited and fallible.
- They used the metrics for *estimating*, rather than for *predicting*. When their estimations didn't work out, they didn't use the discrepancy to punish people. They used it to try to understand what they hadn't understood about the task in the first place.
- They used *inquiry metrics*, rather than *control metrics*. That is, they used the metrics to *prompt questions about their assumptions*, rather than to *provide answers or drive their work*.
- They used a large number of observational modes to manage their business and to evaluate (and solve) their problems. Most importantly, the managers observed *people* and what they did, rather than watching printed reports. They used close personal supervision, collaboration, and conversation as their primary approach to learning about what was happening on the project. They watched the game, rather than the box scores.
- They didn't collect any metrics on things that weren't interesting and important to them.
- They didn't waste time collecting or managing the metrics.
- They had no interest in making the metrics *look good*. They were interested in optimizing the quality of the work, not in the appearance afforded by the metrics.

- They took a social sciences approach to measurement, as Cem Kaner describes the social sciences here (in particular on page 3 of the slides). Rather than assuming that metrics gave them complete and accurate answers, they assumed that the metrics were giving them *partial answers that might be useful*.

In summary, they viewed metrics in the same kind of way as excellent testers view testing: with skepticism (that is, not rejecting belief but rejecting certainty), with open-mindedness, and with awareness of the capacity to be fooled. Their metrics were (are) heuristics, which they used in combination with dozens of other heuristics to help in observing and managing their projects.

The software development and testing business seems to have a very poor understanding of measurement theory and metrics-related pitfalls, so conversations about metrics are often frustrating for me. People assume that I don't like measurement of any kind. Not true; the issue is that I don't like *bogus* measurement, and there's an overwhelming amount of it out there.

So, to move the conversation along, I'll suggest that anyone who wants to have a reasonable discussion with me on metrics should read and reflect deeply upon

Software Engineering Metrics: What Do They Measure and How Do We Know (Kaner and Bond)

and then explain how their metrics *don't* run afoul of the problems very clearly identified in the paper. It's not a long paper. It's written by academics but, *mirabile dictu,* it's as clear and readable as a newspaper article (for example, it doesn't use pompous Latin expressions like *mirabile dictu*).

Here are some more important references:

- The Dark Side of Software Metrics (.pdf, Hoffman)
- Meaningful Metrics (.pdf, Allison)
- How to Lie With Statistics (book, Huff)
- Measuring and Managing Performance in Organizations (book, Austin)
- Quality Software Management, Vol. 2: First Order Metrics (book, Weinberg)
- Why Does Software Cost So Much? (book, deMarco)

Show me metrics that have been thoughtfully conceived, reliably obtained, carefully and critically reviewed, *and* that avoid the problems identified in these works, and I'll buy into the metrics. Otherwise I'll point out the risks, or recommend that they be trashed. As James Bach says, "Helping to mislead our clients is not a service that we offer."

Update: I've just noticed that this blog post doesn't refer to my own Better Software columns on metrics, which were published later in 2009.

***Three Kinds of Measurement (And Two Ways to Use Them)* Better Software, Vol. 11, No. 5, July 2009**

How do we know what's going on? We measure. Are software development and testing sciences, subject to the same kind of quantitative measurement that we use in physics? If not, what kinds of measurements should we use? How could we think more usefully about measurement to get maximum value with a minimum of fuss? One thing is for sure: we waste time and effort when we try to obtain six-decimal-place answers to whole-number questions. Unquantifiable doesn't mean unmeasurable. We measure constantly WITHOUT resorting to numbers. Goldilocks did it.

***Issues About Metrics About Bugs* Better Software, Vol. 11, No. 4,**

May 2009

Managers often use metrics to help make decisions about the state of the product or the quality of the work done by the test group. Yet measurements derived from bug counts can be highly misleading because a "bug" isn't a tangible, countable thing; it's a label for some aspect of some relationship between some person and some product, and it's influenced by when and how we count… and by who is doing the counting.

This entry was posted on Monday, January 19th, 2009 at 5:11 pm and is filed under Measurement. You can follow any responses to this entry through the RSS 2.0 feed. You can leave a response, or trackback from your own site.