July/August 2009 $9.95 www.StickyMinds.com

BETTER SOFTWARE

The Print Companion to StickyMinds.com

WHY AUTOMATE?
6 good (and not-so-good) objectives

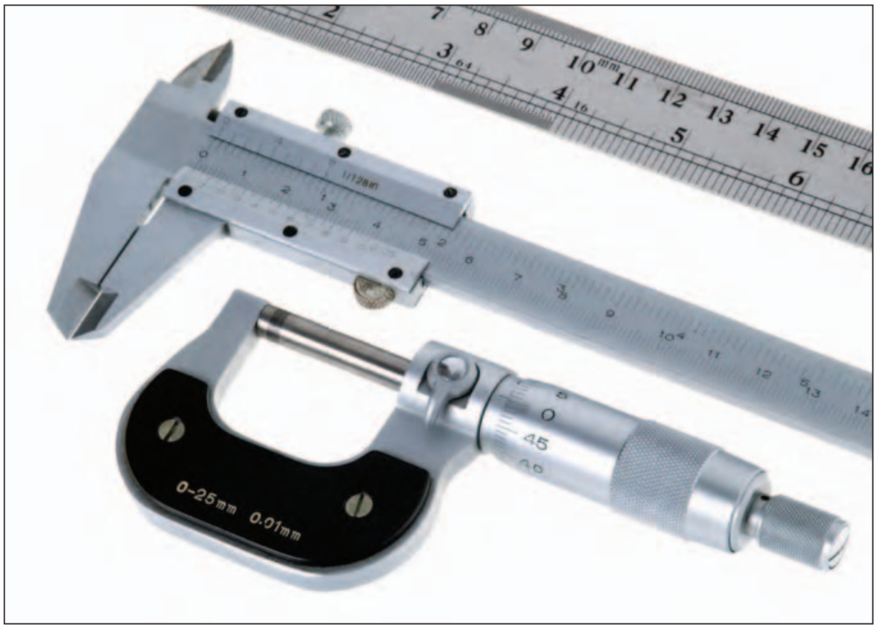
SOUND THE ALARM!
Burn charts for tracking project progress

Developing Applications for a Wireless World

Three Kinds of Measurement and Two Ways to Use Them

by Michael Bolton

People often quote Lord Kelvin: "I often say that when you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind; it may be the beginning of knowledge, but you have scarcely, in your thoughts, advanced to the stage of *science*, whatever the matter may be."[1] But, few note the sentence that precedes the passage: "In *physical* science the first essential step in the direction of learning any subject is to find principles of numerical reckoning and practicable methods for measuring some quality connected with it." The missing sentence prompts some questions: *Are software development and testing sciences subject to the same kind of numerical measurement that we use in physics? If not, what kinds of measurements should we use? How could we think more usefully about measurement?*



Gerald M. (Jerry) Weinberg suggests thinking in terms of three broad categories. [2] *First-order measurement*, he says, is what we need to get started—"just adequate to the task of getting something built." First-order measurement tends to be qualitative, fast, and inexpensive; it generally doesn't require mechanisms or devices to enhance or extend the observation. In a recent conversation, Jerry told me that first-order measurements "are unobtrusive, or minimally obtrusive, and can be used without a whole lot of fuss. They help give you a lot of important information that can lead to other information or, in the best case, to immediate action if needed." [3]

First-order measurement is what we're doing most of the time as we're driving a car. We look through the windows, listen to the engine, and feel the acceleration and deceleration. We make observations and comparisons without getting hung up on quantification. "The road is dry. It's cloudy. There's traffic on the right and a car up ahead with its brake lights on." First-order measurement suggests answers to the questions *What seems to be happening?* and *What should I do now?* In this situation, if you feel like you're driving too fast, you probably are driving too fast. If so, first-order measurement is enough to inform an immediate and appropriate action: slow down.

Because it's based on ongoing experience and feelings, rather than on careful experiments and controlled data intake, wise use of first-order measurement requires us to consider a number of possible interpretations of the meaning and significance of what we see. Suppose you feel like you're driving fast, but not *too* fast. Now you observe a set of red and blue lights on the top of the car ahead. The extra data suddenly prompts you to realize that you're uncertain about your relationship to the speed limit. The situation and first-order measurement prompt a different response in the form of questions: *What else do I need to know?* and *Where should I look?* At this point, you move into *second-order measurement* and refer to the speedometer.

Second-order measurement, says Jerry, is the kind of measurement that engineers use to tune relatively stable systems, making them cheaper, stronger, lighter, more reliable, faster—or slower, if that's what's desired. Second-order measurement focuses on questions like *What's* really *happening?* and *How is it changing?* tending to be more quantitative, subject to more refined models, and generally busier than first-order measurement. It is often assisted by external instruments to supplement or refine direct sensory intake. In particular, metrics—mathematical functions that relate objects or events to numbers via a model—are second-order measurements.

Back in the car, second-order measurement is the kind of information that you obtain from looking at the dashboard. You note that your speed is forty-three miles per hour and that the posted limit is thirty-five miles per hour. Your quantitative, second-order measurement tells you that you're above the legal limit. The apprehensive feeling in your gut, triggered by the combination of police car and the second-order measurement, informs a decision to slow down.

What of *third-order measurement*? That, says Jerry, is the kind of precise, highly quantitative measurement that supports the physicist's search for new natural laws. It helps us answer the question *What happens*? in a universal and general sense. But third-order measurement can be precise only because it tends to be about very simple systems (such as two interacting masses) or very simple models of complex systems (in which we choose to ignore many dimensions of the system, but analyze a very small number of dimensions very thoroughly). Perhaps most significantly, third-order measurement emphasizes and depends upon keeping messy human traits—variability, subjectivity, and values—out of the way. As noted in an important paper by Cem Kaner and Walter P. Bond, [4] using metrics and higher-order measurement wisely depends on *construct validity*—critical rigor in evaluating the models and the functions that form the basis for the measurement.

In Rapid Testing, we define a *control metric* as any metric that drives a decision. Some development groups standardize the decision to ship the product when it contains a low-enough threshold number of high-severity bugs. Others consider a program adequately tested if there's one positive and one negative test per "requirement" (meaning per line in a requirements document). Still others deem a test group "successful" if there is a low-enough percentage of rejected bug reports. By contrast, an *inquiry metric* is one that prompts a question: We have three open high-severity bugs—*What's the story there?* Jim and Mark are two days behind where we thought they'd be—*Do they need help?* The program managers are deferring a lot of problem reports—*Are the problems insignificant, or do we need more training because we don't understand the product?*

One of my recent clients rated the quality of its products and customer satisfaction with a basket of five second-order metrics. Each measurement collapsed months of work and tons of data into a single number. "Better" numbers earned praise; "worse" numbers earned a reprimand, so management meetings dragged on while people tried to explain changes from last month's numbers—especially when things had gotten worse. At this company, schedules frequently slipped and shipments were often delayed. Yet when I asked testers the simple question: *What slows you down?* I got a wealth of information. They told me about broken and buggy builds, inadequate test environments, excessive emphasis on scripts that were out of date by the time the product arrived, and a lack of information about real customer needs. They also said they were wasting time collecting data that wasn't being used to help speed up development or testing, and they offered dozens of ways in which the numbers could be gamed.

A different client, also working on one-year project cycles, focused on questions like: *What happened this week? What did we get done? What problems did we run into?* Managers used personal contact—direct observation of and conversation with people—as their primary approach to assessing the project's status. They took a good number of quantitative measurements, but used them only as indicators to refine their initial assessments and to inform new first-order questions. The team made rough long-term estimates and more precise short-term estimates, dividing two-week cycles into tasks of two days or less, with clear deliverables that signaled completion. When tasks weren't finished in the estimated time, no one was punished; instead, everyone considered what he hadn't understood earlier, what he had learned, and what might inform a better estimate next time. Team members didn't collect metrics on things that weren't immediately interesting and important to them. They were interested in understanding the situation and optimizing the quality of the work, not in the appearances afforded by the metrics. They emphasized the game and the season over the box scores. And they consistently shipped high-quality products on time.

They did use one—and only one—control metric. When the amount of open problems exceeded a certain number, they stopped working on new features and fixed problems until the list was comprehensible and manageable again.

Jerry observes that in software engineering we seem obsessed with higher-order measurements. Why? He suggests that decisions about quality are political and emotional, based on discussions and decisions about whose values count and how much they count relative to one another. [5] Such issues are often distasteful to people who want to appear rational and "scientific," so we try to avoid those issues with appeals to higher-order measurement.

Each new software project involves a human context—interaction between different sets of clients, developers, tasks, and problems to solve, with high variability, contending values, and small sample sizes. In those environments, third-order measurement isn't achievable; it's an expensive distraction. That leaves us with cycles of first- and second-order inquiry measurement—not physics, but easily good enough to build and tune our systems. **{end}**

References

[1] Thomson, William (Lord Kelvin). "Electrical Units of Measurement." Popular Lectures and Addresses I (London, 1981-94).

[2] Weinberg, Gerald M. *Quality Software Management, Vol. 2: First-Order Measurement.* Dorset House Publishing, New York, 1993.

[3] Weinberg, Gerald M. Personal correspondence with the author, May 18, 2009.

[4] Kaner, Cem and Walter P. Bond. "Software Engineering Metrics: What Do They Measure and How Do We Know." 10th International Software Metrics Symposium. Chicago, IL, 2004. **www.kaner.com/pdfs/metrics2004.pdf**

[5] Weinberg, Gerald M. *Quality Software Management, Vol. 1: Systems Thinking.* Dorset House Publishing, New York, 1991.

What's your experience with observation and measurement in your organization?

▼

Follow the link on the **StickyMinds.com** homepage to join the conversation.