Familiar Metric Management: Come to Dinner, Henry!

A hundred years ago a dedicated inventor worked alone in a dusty attic, his physical maintenance (Come to dinner, Henry!) attended to by a loving wife, or sometimes a mother or sister. His drive originated within his own spirit. Who else would back a “perpetual motion” machine?

Today we still have dedicated loners, working in a garage, attics having become passé. But most of us, in the software field, work in large organizations. We don’t come, ready mixed, with the overwhelming drive of the lone inventor. Yet we are doing very difficult work that demands high creativity. Moreover, we have skills, based on an extensive knowledge base, that are in great demand elsewhere. That large organization has to pay attention to our psychic needs, sometimes termed “morale,” or we might walk. That is the subject for today!

Feasibility. Let’s look first at the very start of a big project. Is it feasible? This issue is not unrecognized. An opening stage for feasibility study appears in Department of Defense directives. Our observations indicate that the minimum length of this stage is about one quarter of the mainbuild development time the project itself eventually takes. The number of staff is usually small, but they ought to be highly experienced. Feasibility means that your organization has the capacity, or can readily add it, to carry out this project. It is within your state of the art technically. You can likely accomplish it within the general limits of development time, cost, and quality criteria that the client has in mind or the marketplace sets.

Later, lacking a determination of feasibility, when the people in the main build begin to sense the reality—this project is not feasible—the best ones edge away. The time servers stay to collect their paychecks. Eventually the project collapses and we read about it in a horrified media. By implication the media place the blame on software people, but it was not they who originally decided to skip the feasibility stage.

Functional design. The next stage of software development is called by various names:

requirements determination, functional design, architecture, or risk reduction. Whatever we call it, its purpose is to come up with a fairly concrete architectural-level design of what is to be done. This stage tends to be at least one third the length of the main build. The software people need this high-level design stage in order to estimate within business-set limits, now much closer than in the feasibility phase, the development time and amount of effort the main build will take. They need it to project a defect rate. They need it to plan the work in some detail. The people in this stage have to investigate the serious risks in order to reduce them to a level at which they can plan and estimate the remaining work.

When the main build gets under way and the working people find that the powers-that-be skipped these preliminary steps—the schedule is unrealistic, the risks are unresolved—morale plummets some more. People feel more comfortable when they can see the path ahead. Architecture and a plan lay out this path.

Knowing what to do (architecture), having enough time to do it (schedule), knowing it can be done (risk reduction), and actually doing it (milestones) enable people to feel closure every time they pass a milestone. That kind of pattern makes light work of heavy software lifting. As we were writing this column, Tony Danza, now appearing in a Broadway musical, told an interviewer that the immediate applause he gets on stage is a tremendous boost, compared to the much-delayed report that Nielsen ratings bring to a television star.

Milestones. “Chronic schedule slippage is a morale killer,” Fred Brooks found 30 years ago and reported in *The Mythical Man-Month.*

His solution: make the milestone a sharp-edged point, so sharp that developers and first-line supervisors can't fool themselves into thinking they are 90 percent complete. Wishful thinking comes easy to our species. It probably came as we developed language, pulling away from the great apes. At first, words referred to things like man, woman, and child—fairly sharp items. As we moved on to more abstract concerns, words became more fuzzy. What is 90 percent complete, for example? Fuzzy words lead to wishful thinking. The consumers of status reports at the higher levels are still human beings, still subject to letting their wishes overwhelm a fuzzy metric. *Moral*: get the metrics sharp. It's good for *morale*.

If we have milestones, even if they are sharp, they can be *missed.* A sharp status report makes the *miss* painfully evident. Then what? Brooks has further guidance for us. "Every boss needs two kinds of information, exceptions to plan that require action and a status picture for education." The boss who thrashes out in anger as soon as he hears the bad news is not good for the morale of his subordinates. Rather, he should keep in mind three rules for hearing metric reports.

- Stay calm. Otherwise the report preparers may twist the metrics to keep you calm. Actually, you need the straight facts.
- Defer your reaction. Your subordinates deserve the opportunity to act or, at least, to advise you.
- Call a later, action-oriented meeting. It may be only a coffee break later if action is imperative, but preferably it should be a day off. All involved need the time to think through the problem.

Measure work, not people. On the one hand, there are merit reviews, performance appraisals, periodic pay increases, and terminations. These operations, if you like, "measure" people. On the other hand, there is the measurement of work for the purpose of making estimates, projecting work plans, and checking progress against the plans. If you must "measure people," do it as an entirely separate operation from the measurement of work. Keep these two activities in separate compartments.

The reason: people don't like to be measured. If they believe the "measurement of work" is being surreptitiously converted to the "measurement of people," they tend to twist the work measurements to make themselves look better. Estimates, plans, and control are too critical to the business to be based on twisted data.

Don't focus on one metric. Within the scope of "measuring work," don't put special emphasis on one metric to the degree that people begin to think that it is all management is interested in. If that belief becomes prevalent, people begin to twist what is going on to make that one metric look good. Then that twisting twists the other metrics in the other direction. The result is that all the metrics become somewhat inaccurate, and estimates and controls based on them become less effective.

Metrics are not fun. Acquiring, compiling, and keeping metrics handy takes work and time, and time is often scarce in software circles. Because metrics take work and time, people are tempted just to go through the motions. In time the measures become less accurate and less accessible.

Why does this gradual descent into metric darkness happen? Basically, because the metrics are not used. Of course, management can periodically mention the importance of metrics. But talk is cheap. More important, look at them every week or month when they are compiled, and make sure some one notices that you are looking.

The real payoff comes in using the metrics. Use them as a base for estimates, for assigning people and resources in the proper numbers for the time needed, for controlling progress against the plans, and most important of all, for taking action, obvious action, when the comparisons show that progress is off plan. Words have become pretty cheap in the Internet society, but a few still resonate. "Come to dinner, Henry!" People see and feel action.