# Familiar Metric Management: The Future of Metrics in Software Development

**State of the Art.** A few of the best companies have decent metrics programs. By that we mean that they have discovered that there are, fundamentally, four key metrics. They are: function (measured by size, usually in source lines of code, sometimes in function points, modules, or subsystems), schedule, effort, and reliability (defect count). Other metrics are useful for particular purposes, but these four are the inescapable minimum if an organization is to estimate future projects with reasonable accuracy, control those in progress, and have some measurement basis for improving its software process.

Immature organizations, such as those on the Software Engineering Institute's Capability Maturity Model Level 1, seldom have much in the way of useful metrics. They sometimes try to start with an edict from on high containing an extensive list of things (the . . . "ilities") to measure. They put out ponderous pamphlets, but most of these metrics have little value, at least when a company is just beginning to think about quantitative management. Moreover, doing it all would require lots of servicing by busy system development people. The natural result is that metrics get short-changed. Lower level managers don't press the effort. Developers guess at counts. Everyone ignores whatever is collected. It is a lip service game. Essentially nothing happens.

At any rate metrics penetration is fairly modest—on the order of 20 to 30 percent of software organizations.

**Benchmarking.** Software benchmarking has been slow to catch on and be used in any widespread way. Those organizations who have done it are those most advanced in applications development and that seem to have a good appreciation of the advantages accruing to companies that know where they are, what their weaknesses are, and how they stack up with their competition. In fact, competitive comparison seems to be the most important reason for doing benchmarking.

Benchmarking for process improvement usually follows after senior management sees the benefits accruing to them and then devises strategies to improve more rapidly than their competition and manage and accelerate the benefit stream. Benchmarking soon tells senior management that knowledge is competitive power.

Although the take-up of quantitative benchmarking has been slow, we expect it to increase gradually and eventually become widespread as focus on the SEI CMM process improvement concepts become more widespread and as organizations doing development outsourcing insist on measurable standards before hiring vendors to build software for them. Moreover, when organizations realize that coupling the SEI survey approach to determining their maturity level with quantitative benchmarking using the key management metrics (Function, Schedule, Effort/Cost and Defects) they then have a powerful capability to manage their software development progress in a much more focused, productive way.

**Practitioner focus.** Metrics is not an end in itself. The number on a speedometer in a car is not a mystic talisman. It is there to keep you from wrecking the car on the next curve. Similarly, software metrics are there to help you get projects done on time, within budget, at an adequate reliability level and to enable you to gauge process improvement from project to project. Therefore, practitioners should focus their energy on the use of the management-oriented metrics: function, effort, schedule, and defect numbers.

We believe that the metrics we derive from these basic numbers, process productivity and manpower buildup—that we have explained in past newsletters, are an additional aid to software people. Those not familiar

with them can measure benefits in terms of schedule reduction, cost reduction, and defect reduction. They can understand that "small teams are better than large teams," that "enough time is better than too little time." Issues such as those are what they should focus on.

**Research focus.** Large-scale software projects are accomplished in economic settings, whether in business, industry, or government. In these settings, a researcher cannot hold constant all factors except one, and observe and measure that one. They are difficult research settings. Yet that is where the experience occurs from which industry needs to learn. For example, Michael Cusumano studied the Japanese software factories almost as an anthropologist would. Recently he and his associate, Richard Selby, observed Microsoft in much the same way—scores of interviews, much reading of internal documents and statistics. So, our suggestion to researchers is: go where the action is.

**Metrics must evolve.** That modest level of metrics penetration applies to today's software organizations, where most are organized on a project basis and need mostly project metrics. A few leading companies—sometimes only a few divisions within leading companies—have organized software production on a basis broader than the project, that is, they have some kind of reuse organization creating software components for their project teams. They have a support group moving these components out into the projects. They have people in their projects pulling the components into their analysis models, design representations, and code.

The point of metric significance is that these reusable components are being applied over a wider organization area than a single project. They are being applied over a longer time than that of a single project. The significance adds up: more organizations involved; more projects covered; longer time period. Project-level metrics will no longer be enough. We need metrics to keep track of what this more extensive software organization is accomplishing. Is it paying off? Can this scale of operations show a return on investment?

Moreover, there is another cloud on the horizon: business process reengineering. Again a few leading companies are applying BPR successfully. Others have tried, but couldn't hack through the complexities. It really means: Focus on the business process. In stark grandeur, that process is nothing more than getting a good product to a customer on time and getting paid for it. That has taken a lot of coordination through a hundred production jobs under dozens of supervisors managed by a hierarchy of executives, abetted by a horde of clerical assistants. Or, at least it did in the old days, before IT (information technology). Now IT can take over much of the load of coordinating the new business process.

The complexity comes in the fact that the business process is being redefined. Part of that redefinition consists of moving much of the coordination load from the management hierarchy into IT. But IT must be redesigned to handle the new load. At the same time, it must carry on whatever it was already doing, because a going business has to keep going. BPR just adds another level of complexity to the more extensive software organization that reuse will bring in its wake. Will the new IT supporting the new business process be worth what it costs? Good metrics will give an early clue—before the market pronounces its final decision.