Familiar Metric Management: Experience the Wisdom that Goes with Knowing Where You've Been

"Few executives yet know how to ask, What information do I need to do my job? When do I need it? In what form? And from whom should I be getting it?" **Peter F. Drucker** [fn]Peter F. Drucker, *Managing in a Time of Great Change,* Dutton/Penguin Group, New York, 1995, 371 pp.[/fn]

The application of metrics to knowledge development, particularly as that development takes place in software, is still in progress. Many believe that a software developer, sitting at his desk, apparently in repose, offers little to measure.

That is a deceptive illusion. In truth, the output of software developers is systems that do things, as surely as shoes protect tender feet from sharp rocks. Producing that output takes human effort over calendar time at a productivity rate resulting in a product quality level—all measurable, as we have demonstrated in earlier columns.

Measures by themselves—a single number standing alone—are of little value. It is in comparing such a number with some one's experience with the same metric that the measurement achieves management value. It follows, then, that a software organization needs a metrics repository to hold these numbers with which it can compare its current results.

Can you answer these real-world questions?

How long will it take us to develop our next software product?

How will this calendar time compare with our own past efforts?

Are we shortening our development time as compared to past, comparable projects? How does the time we take compare with other development organizations doing similar work?

How much effort (person-hours) will our next software development take?

How does this effort (cost) compare with that of our own past similar projects?

Are we making consistent progress in reducing effort?

How do our costs compare with those of competitive organizations?

Do we have a means of classifying projects, our own and others, into similar categories to facilitate comparison?

Do we have a means of measuring the level of productivity with which we carry on software development?

Are we obtaining benefits from process improvement justifying the investment we are making in it?

Can we measure and, hence, compare the software reliability level we achieve in our software process?

We base this batch of questions on the five basic metrics that govern software development:

1. size or functionality;
2. calendar time;
3. effort;
4. process productivity, and
5. product reliability (or absence of defects).

There are, of course, many other metrics that can be, and often should be employed in software development. These five are the inescapable minimum!

Part of the answer is knowing where you have been

“Where we have been is contained in cardboard file boxes out in the abandoned foundry building,” the young project manager informed us. “I had not been born when the company gave up foundry operations. I suppose we ought to destroy those records, but no one has the time to sort through them. I’m sure they are of no value, because no two projects kept data in the same way.”

“Are you keeping consistent data now?” we inquired politely.

The project manager smiled sheepishly. “No, I suppose not.”

Knowing where you have been is too big an issue for a single project manager to get his arms around. It is a task for the whole company, but each project manager has to carry out his part of it. It goes something like this.

Define the information about each project that the company is to keep. If information is to be meaningful, everyone concerned has to understand what it means. If a company can adopt industry-wide definitions, then it can compare where it has been with where its competitors are.

Delimit the defined data to be kept. It is not uncommon for software organizations when first they begin to keep metrics to keep too many. “It might be nice to have this item later,” they tell each other. Well, it costs time and effort to add data to a database, though the costs of the data’s sitting there may be miniscule. The point is: keep a correspondence between the information you decide to keep and the information you will later use.

Discipline the collection of this defined, delimited data. Somebody has to be responsible for collecting the data, usually several somebodies in a large organization. Management has to pay attention to the collection process. Otherwise, with the passage of time, the incentive to pay attention to data collection runs down.

Deposit the defined, delimited, disciplined data in an established depository. In this age of databases, of course, the depository is itself a database. A database is accessible in a way that old foundry buildings are not. Management may have to limit access to it to managers, estimators, and accountants and protect it from competitors.

Deliver the needed information in readily usable formats. Computer programs can accomplish steps like fitting curves, drawing mean and standard deviation lines, plotting curves of the current project on a background of previous projects, and so on, as illustrated in the figures.

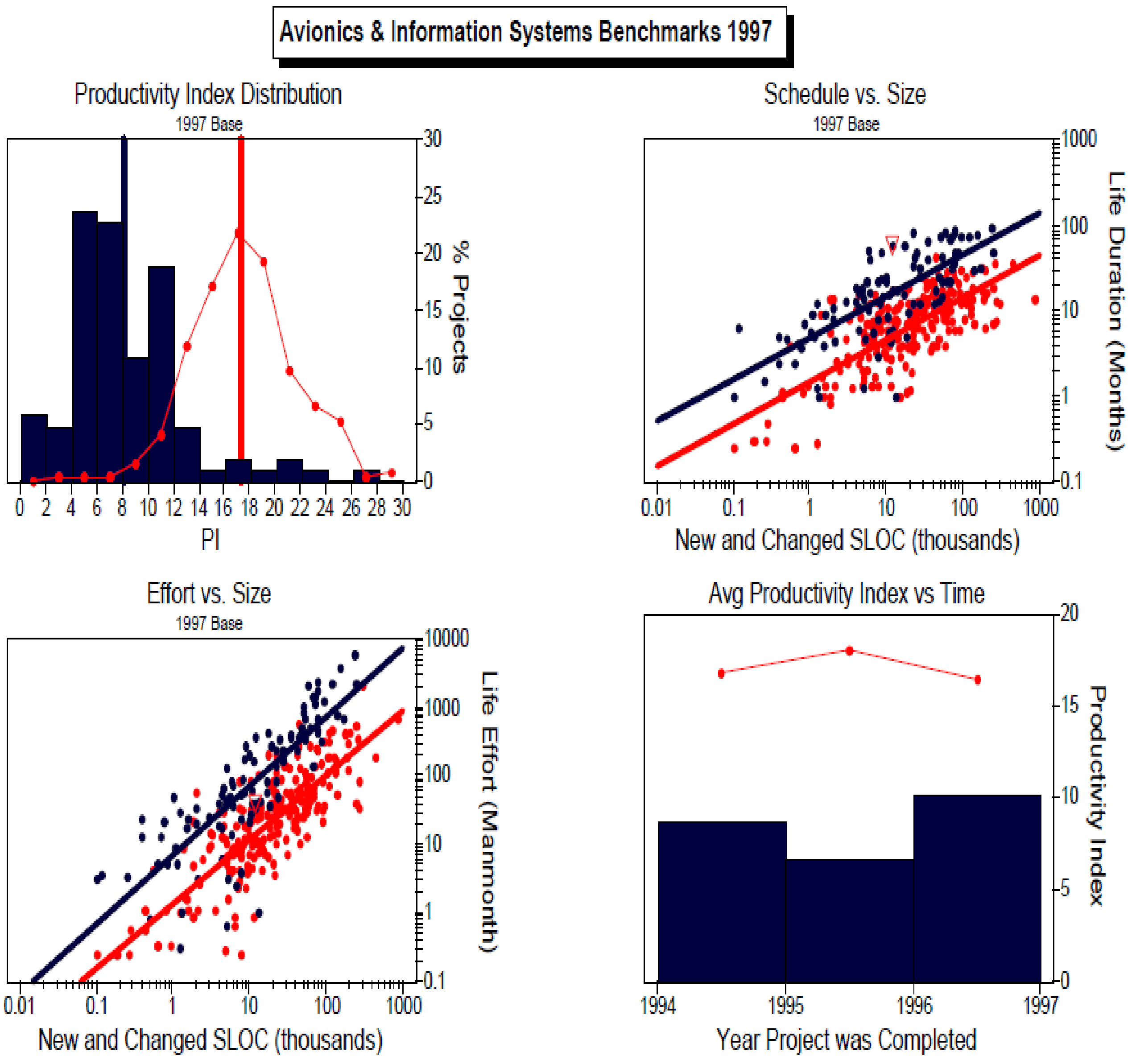

Figure 1. These four figures compare avionics software and information-systems software projects as of 1997. (Figures courtesy of QSM's SLIM-Metrics tool)

Top Left. Compares process productivity index of information-systems projects (single line) with avionics projects (bar chart).

Top Right. Compares schedule performance of information-systems projects (lower set of dots) with that of avionics projects (upper set of dots).

Lower Left. Compares effort performance of information-systems projects (lower set of dots) with that of avionics projects (upper set of dots).

Lower Right. Compares average process productivity index of information systems projects (upper single line) over a three-year period with that of avionics projects (bar chart).

When you keep your own data in a database tool, you can depict your own situation in comparable ways. For example, you can show the estimates for a forthcoming project against a background of your own past similar projects. Managers can see that the new estimate is within the limits of the company's own past experience.

Depend on tools. Developers need no longer master the underlying techniques of analytical geometry and statistical analysis that used to take up semesters of college time. As is often the case now in software development, tool builders, including those who build estimating tools, incorporate more and more know how in the tools themselves.

Managers and developers no longer need spend days on adding up columns of figures to determine means and standard deviations. They merely have to know what these descriptors of data mean and how to have the tool provide them.

"... Practically no one asks, What information do I owe? To whom? When? In what form?" **Peter F. Drucker**