# The Best Measurement Tool Is Your Telephone

Don Scott Lucero
*U.S. Army Software Metrics Office*

Fred Hall
*Independent Engineering Inc.*

*More and more people are becoming interested in software measurement for three reasons: Measurement and analysis are a Level 2 process area of the Capability Maturity Model Integration[SM], the International Standards Organization measurement standard ISO 15939 is being drafted, and Practical Software and System Measurement version 4.0 is being released. This paper provides some practical advice on implementing a software measurement program based on lessons learned at the U.S. Army Software Metrics Office during the last 10 years.*

Everyone should know why the organization wants to measure. The most effective measurement programs are tailored to the important issues that must be managed in an organization. Although many software metrics programs are based on this issue-driven measurement approach, many are not effective because the selected issues are not commonly endorsed. One lesson learned by the Army Software Metrics Office (ASMO) is that the members of the organization that implement a measurement program must feel that they were involved in defining the issues to be measured.

The Army learned this lesson early. In 1990, the Software Test and Evaluation Panel (STEP) initiated an issue-driven process to define a standard set of software metrics for all Army programs. Members of the STEP metrics working group defined the Army-wide software management issues based on their experiences in Army software acquisition and test. Table 1 describes the Army issues and the metrics that were selected to address them. The selected set of 12 metrics was implemented in 1991 as a mandatory requirement for all Army programs. As expected, the STEP mandatory metrics program was received with extreme hostility from most acquisition program managers.

Hostility and marginal success were expected because the first objective of the mandatory metrics program was to shock managers to comply with software metrics policy requirements that had been ignored for years. However, the STEP members were surprised to learn that the most significant cause of this hostility was not the financial impact on the projects. Army managers were more upset by their perception that an *outside* organization had mandated how they would do their job. The underlying problem with this approach was that the Army program managers who had to implement the metrics were not involved in defining their issues.

The Army policy for the 12 mandatory metrics was superceded in 1996 with a policy [1] that requires each project to implement a set of software metrics to support its own project-specific issues in six common issue areas. The ASMO has found that the best approach to define project-specific issues is through a series of measurement-tailoring workshops. The workshops provide the forum to allow persons at all levels of the organization to define the project's issues in terms of their own problems, concerns, and lack of information in their day-to-day jobs. Employees tend to support a measurement program if they feel that they had a part in developing the process.

**Never change an existing process to obtain a measure.** The second most important lesson the ASMO learned is to only use data currently produced by an organization's process. This principle dictates that measurement programs should start small and not require any significant additional resources to implement. Radical change of any process in an organization (business, management, or technical) will usually render the process ineffective for a period of time until the new process is learned and becomes institutionalized. A radical process change may cause employees to no longer understand or support their daily activities. Employees will spend their time relearning their jobs or complaining about the new order. The overall result is that both the organization's new measurement process and other processes objectives are not achieved.

However, the lesson is not that an organization's process should not be improved to be able to provide more effective measures. The point is that radical change of any process is usually destructive. Managers should realize and accept the fact that organizational change to improve a process will be slow. For example, the empirical data that has been collected in the Software Engineering Information Repository (SEIR) shows that the median time to move from Capability Maturity Model® (CMM) Level 1 to Level 2 is 26 months for those organizations that began process improvement programs after 1992. The SEIR is sponsored by the Software Engineering Institute (SEI) and provides an extensive body of empirical evidence on software process improvement. The SEIR can be reached at www.seir.sei.cmu.edu

**Measurement provides managers with an important, but limited, opportunity.** The greatest benefit of a

Table 1. *Army issues, metrics selected to address them.*

| Army Experience Issue | Metric |
|---|---|
| Software user requirements are not achieved in the code. | Requirements Traceability |
| Software fails under unexpected operational load levels. | Reliability<br>Breadth of Testing |
| Unexpected changes are made to user requirements. | Requirements Stability |
| Software is released without adequate test. | Breadth of Testing<br>Depth of Testing |
| Excessive design changes are made to the software. | Design Stability |
| Inadequate cost estimates are overrun. | Cost |
| Inadequate schedule estimates are slipped. | Schedule |
| Software problems are not resolved prior to release. | Fault Profiles<br>Complexity |
| Physical computer resource capabilities are exceeded. | Computer Resource Utilization |
| The software developer does not have the capability to deliver an adequate product. | Software Engineering Environment |

[SM] The Capability Maturity Model Integration and CMMI are service marks of Carnegie Mellon University.

measurement process is to improve objective communication within an organization. Fred Brooks' experience [2] in developing the IBM Operating System/360 in the 1960s convinced him that "all programmers are optimists." Brooks also found that optimism was pervasive throughout most software development organizations: "When a first-line manager sees his small team slipping behind, he is rarely inclined to run to the boss with this woe. The team might be able to make it up, or he should be able to invent or reorganize to solve the problem."

The ASMO has learned that these observations are still generally true after 35 years. They define the underlying need for measurement in a software development organization. As software can be rewritten and replaced, persons at all industry levels tend to feel that any technical problem can be resolved if they work hard enough. Communication in the software industry on the current status of software products is generally optimistic. The result is that bad news is not easily reported, but is retained at various levels of a software organization because employees feel they can resolve a problem before they will report it.

It is best that customers and managers at all levels realize that any subjective communication will tend to be optimistic. Problems will only be reported when they have grown too big for an employee to admit that he can no longer resolve them. For example, software development projects get behind one day at a time; however, schedule slips are usually reported to a customer only when they are several months behind, and some product must now be delivered. What a manager needs is objective data that is routinely collected as a by-product of an established process in the organization. This objective data will provide unbiased testimony on the status of a process.

**Measurement is only one technique in an effective software management process.** The process to define the project issues may show that other actions are more important for an organization than collecting data. For example, product quality metrics such as software trouble reports are ineffective if the software complexity prohibits adequate testing. If the system does not require complex software, managers should dedicate their resources to reducing software design complexity before measuring product quality.

Also, if software managers are not willing to make decisions based on software measurement data, a software measurement program is of little use.

**Many software measurement reports are only as good as initial estimates of measures' target values.** Many software metrics (cost, schedule, computer resource utilization, etc.) report a comparison of planned versus actual values. The most obvious information that is obtained from the metrics data is a comparison of the actual values to the planned values at some point in time. In most cases, this information only reflects how good the initial estimate was to the actual values that are achieved.

Because a manager will gain little by revising the initial estimate, these metrics often provide nothing but bad news that a manager cannot change. Spending too much time and effort to report and evaluate these metrics will provide little return on investment. Remember that the primary return on investment for a metrics program is the information that allows a manager to make an informed decision on an issue.

The underlying fact is, going back to Brooks' advice, that many estimates in software engineering are optimistic and are not based on reality. For example, in a section entitled Gutless Estimating, Brooks made this observation on schedule estimates:

> "Now I do not think software managers have less inherent courage and firmness than … other engineering managers. But false scheduling to match the patron's desired date is much more common in our discipline than elsewhere in engineering. It is very difficult to make a vigorous, plausible, and job-risking defense of an estimate that is derived by no quantitative method, supported by little data, and certified chiefly by the hunches of the managers."

The key to using metrics that are based on an initial estimate is to understand the basis of the estimate and the link to reality. If an initial estimate was unrealistic, users should adjust to the fact that the variances between planned and actual values will always look bad. However, these measures should not be discarded. These measures' users must adhere to one of the lessons that the ASMO has learned to achieve a good measurement process: "Do not focus only on the quantitative value of the data, but capture other information from the objective communication that usually results from the measurement process." For example, if you know that the schedule estimate is unrealistic, encourage persons in the organization to think of actions that may be taken to reduce the adverse impact of the upcoming schedule slip.

**Late delivery makes any metrics data useless.** The fundamental return on investment in a measurement program is the action that can be taken to effectively manage an issue. Measures provide relatively unbiased testimony on the issue's status, but do nothing to resolve the issue. Therefore, the most common measurement program benefit is early warning of a problem that allows time to avoid it. A slow data reporting process will delay any information that can be derived in time to take corrective action. Late reports may eliminate management options and reduce usefulness of a measurement program. Long delays only report a painful history, and the measurement process becomes a management burden.

The optimum solution is to use the measurement process to support *instant* information through improved communication. Army managers have learned that a defined set of software metrics will identify the important issues in a software process to all members of the organization. If the members of an organization understand the importance of data on status reporting for an issue, they tend to derive and report the information before the data reports are formally delivered.

Improved communication is a valuable objective in managing a software process. By the time measurement data is formally delivered to the appropriate management level, some significant time has already elapsed. A reasonable time between collection and reporting usually is 30 days. A slow measurement process may take two to three times as long. This lapse in time may limit the ability of the manager to take effective action. Data that is reported too late to support management informa-

tion and action is useless. It is important that all participants in a measurement process understand this principle and report the results of data as soon as they are available.

Actually, the least important elements of an effective metrics program are the data reports and indicators that are eventually delivered. Since the objective is to allow managers to derive information as early as possible, the telephone is the most important measurement tool. If the tool is effectively used, managers at the data source will immediately broadcast any important information on the measured issues. In the best possible measurement process, the phone will ring long before measurement reports are delivered.

**Data reports symptoms, not causes.** Clever data presentation and intricate analysis methods may not be worth the effort. Data reports should confirm the expectations of the manager who understands the process and the issues that are measured. If the data is a surprise, the manager should pick up the phone and ask what happened. If the manager does not know whom to call to get the answer, that manager needs to learn more about the software process. More information on the underlying process is needed before the manager should use the data to draw conclusions and make decisions.

Data reports should not be used for a management decision without also reviewing other supporting information that provides insight into the software process. For example, a data report showing that some element of a project has exceeded planned cost does not independently allow a manager to derive information on the actual budget status. The reason for the high cost may have been that the organization made an early purchase of additional equipment to support a prototyping effort. Although expenditures are higher than scheduled, the early expense and prototype capability will allow a stable requirements baseline to be defined and will eventually lower the overall project costs. If a data report shows that a project has over planned cost, a manager is able to ask intelligent questions only if that manager understands current events in the project.

**The information value of metrics data depends on who reads the data.** Metrics data reports provide two limited opportunities to a manager:

1. The data confirms the manager's expectations and assumptions on the current status of an issue.
2. The data is a surprise, and it allows the manager to ask intelligent (issue-driven) questions.

A manager who obtains news on a project only by reading metrics data reports should not make significant management decisions. The manager who reacts to only data reports probably does not know enough about the project to make an informed decision. Management decisions and actions usually must balance several competing objectives and issues within an organization.

**Most objective metrics data is influenced by someone's subjective judgment.** Quantitative data usually is assumed to be accurate and unbiased because it is expressed as a precise number. However, managers should be aware that every data item that is defined and collected by another person will reflect some degree of subjective judgment. For example, metrics data on the number of design units that have passed integration test usually is assumed to be a precise number, determined only by technical characteristics of the software under test. However, the number of tests that have been passed depends primarily on the criteria used for the test. Examples of changes to integration test criteria follow:

- Level of assembly or the size of software components that are integrated for each test case.
- Number of inputs or conditions required for each test case.
- Level of conditional stress during test (concurrent tasking, shared memory availability, processor utilization, etc.).

The pass/fail criteria for a test is not reported in the metrics data, but any changes may radically drive data up or down. The principle that managers should understand is that the quantities that are reported by metrics data are a by-product of another process that may or may not be stable. Deriving information from metrics data requires a manager to understand that other process.

**Not all organizations benefit from a metrics program.** The process to identify the project issues will often determine that an organization is just not ready for measurement. The ASMO's experience in administering measurement-tailoring workshops has shown that many organizations must improve other software processes before an effective measurement process can be established.

For example, when selecting issue-driven measures, the data that is currently collected in the organization's configuration management (CM) process should be considered. If the data that is collected in the existing CM process cannot support a very basic measurement program, it is more important to first improve the CM process than to start collecting metrics data. Often, differences in the environments and methodologies within an organization or with software vendors make communication and software data collection difficult, at best.

**Not all numbers are equal.** Managers should have insight into the measured software project to know what numbers are most important. For example, it is understood that software problems or trouble reports must be assigned a priority according to the impact on the system or project. However, many other metrics should also be provided with a priority ranking. All software requirements, software components, test cases, etc. are not equal in their impact on the software process and products.

Managers must understand the priority or criticality of a measured element to the overall objectives of the system or project. The guideline is that managers should know which measured elements have the most impact on their project issues.

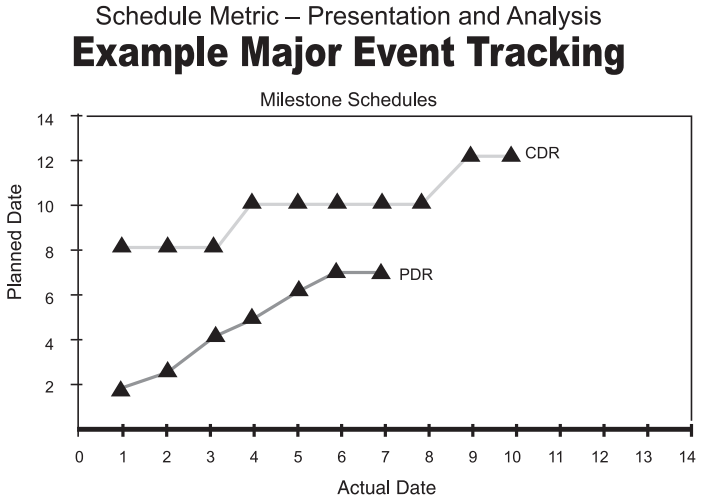
**Data indicators should be tailored to a project issue.** When building a data report or graphic indicator, managers should consider the specific kinds of information that is most useful. For example, when measuring schedule progress in a software development effort, the change of dates usually is not as important as the developer's reaction to the schedule changes. The ASMO's recommended display for the schedule metric is provided in Figure 1 as an example. In this indicator, we plot

planned versus actual dates of events. Although this indicator is confusing, the data display focuses attention on the changes to future events that are mandated by schedule slips.

The ASMO's experience is that the final delivery date for a software product usually is established by system delivery or marketing promises that have little to do with the schedule needed to develop the software. Early software schedule slips will rarely delay the promised final delivery date. When schedule slips occur, future activities between events are compressed to ensure the planned delivery date is achieved.

The data display in Figure 1 shows that in month six the Preliminary Design Review (PDR) has slipped by five months, but Critical Design Review (CDR) remains within two months of the original schedule. The data display highlights the most important schedule issue. The developer now must complete the detailed design between PDR and CDR in three months, not the six months that were originally proposed.

Figure 1. *The ASMO's recommended display for the schedule metric focuses attention on the changes to future events that are mandated by schedule slips.*




**Never implement an expensive measurement program.** The only realistic reason that a measurement process will incur some significant cost is that the metrics data is not currently produced by the organization.

Two basic rules that must be observed to define an economical metrics program are:

- Only use the metrics data that is currently available from the organization.
- Do not require a metric to be reported if it would require the organization to immediately adopt or change an engineering or management process.

The ASMO has found that most violations of these rules occur when a customer requires a software vendor to provide metrics data. Often a customer will request data that is an effective practice for software engineering but is not a requirement element for all software processes. A common example of this type of metric is "cyclomatic complexity." This metric is commonly misused when a customer requires this data to be reported, and the vendor's process does not currently measure cyclomatic complexity. To report this metric the vendor must either (1) measure the complexity of design and code after it is completed, or (2) adopt a new quality process that is based on cyclomatic complexity. The new process will require new tools, training, and procedures for the software design team.

The first option to report cyclomatic complexity will produce a number but does nothing to change the vendor's process or the product that is delivered. The second option may eventually improve the vendor's process and the complexity of delivered software. However, while the vendor is learning the new complexity-driven quality process, his overall productivity and product quality may be degraded. The customer should adopt neither option. All managers should follow rule No. 1 and only use data that is currently available from an organization.

Any software engineering organization should be able to report basic configuration management data for little additional cost to a customer. If a software vendor is unable to provide this basic data at reasonable cost, it is clear that the customer has not learned the most important guideline for an effective metric program: "Do not hire a dummy to develop your software."u

## References

1. Memorandum, Director of Information Systems for Command, Control, Communications and Computers (DISC4) policy SAIS-ADW, subject: Acquisition Reform and Software Metrics, Sept. 19, 1996.
2. Brooks, Fredrick P., *The Mythical Man-Month, Essays on Software Engineering,* Addison-Wesley Publishing Company, 1975.

## About the Authors

**Don Scott Lucero** is a software engineer on the headquarters staff of the Army's Evaluation Center. He is responsible for the Army's Software Metrics Office as well as Army Test and Evaluation Command's software test and evaluation policy and methods. Lucero has 17 years of experience working on Army software development projects and has both bachelor's and master's degrees in computer science.

U.S. Army Software Metrics Office, Attn: CSTE-AEC-MA
Park Center IV, 4501 Ford Avenue
Alexandria, Va. 22302-1458
Phone: 703/681-3823
DSN: 761-3823
Fax: 703/681-2840
E-mail: lucerodon@atec.army.mil
Internet: www.armysoftwaremetrics.org

**Fred Hall** provides training and product assurance engineering support, including software reliability, quality assurance, and systems reliability and maintainability. Hall has supported the Army Software Metrics Office from November 1989 to the present. He has also provided support to the Department of Defense Practical Software and Systems Measurement program. He received a master's degree from George Washington University in 1974, and a bachelor's degree in mechanical engineering from the U.S. Naval Academy in 1970.

Independent Engineering Inc.
4 Old Station Road
Severna Park, Md. 21146-4619
Phone: 703/979-9674
Fax: 703/979-8187
E-mail: fhalliei@aol.com