



Focusing Measurement on Managers' Informational Needs

Peter Baxter
Distributive Software

Establishing a measurement process has evolved from the days of, "If it moves, count it," through the goal-question-metric period to today's information-needs-based approach for identifying and defining what to measure. Measurement process guidance from ISO and Practical Software and Systems Measurement provides a robust and flexible framework for measurement, but they only identify the purpose of measurement as the "informational needs" of managers. What are these informational needs? This article describes simple techniques for identifying informational needs within your organization, i.e., informational needs that become the requirements of your measurement process and lead to a useful and effective measurement process.

In many organizations, a measurement process is a required element in managing technology programs. To meet this requirement, several groups have initiated projects to develop and standardize a set of *best practices* for setting up such a measurement process. Three principle sources of measurement guidance that have converged from these groups' efforts during the past two years are as follows:

- The emerging ISO standard, ISO/IEC 15939 Software Measurement Process.
- The Practical Software and Systems Measurement (PSM) Guidebook version 4 [1].
- The Software Engineering Institute's Capability Maturity Model® (CMM®) IntegrationSM (CMMISM) project [2].

Measurement guidance and principles are consistent among these three documents, with the basic measurement process model shown in Figure 1.

As shown in Figure 1, *informational needs* drive the planning, performance, and evaluation activities within the measurement process. Informational

needs are the requirements of essential measurement activities in the *core measurement process*. Once informational needs are defined, a *measurement plan* is developed by decomposing them into *analysis results and performance measures* (information products), which contain measures and associated guidance. These information products are delivered to managers and drive *improvement actions*.

By placing informational needs outside the scope of the core measurement process itself, the measurement process can be used to support a wide range of management and executive functions. While this makes measurement a flexible process, it also requires that informational needs be clearly reviewed, prioritized, and documented *before* initiating or expanding a measurement process. Because identifying informational needs is a critical step in ensuring measurement process success, organizations must take the time to correctly select and define these needs.

This article presents common types of informational needs that are found in

both commercial and government organizations involved in systems and software engineering. Many also address the informational needs of key practices from the CMMI. These common needs should encourage you to extract potential measurement process requirements from your organization's management, system or software, and support processes. The following nine techniques provide tools for extracting the real informational needs within your organization.

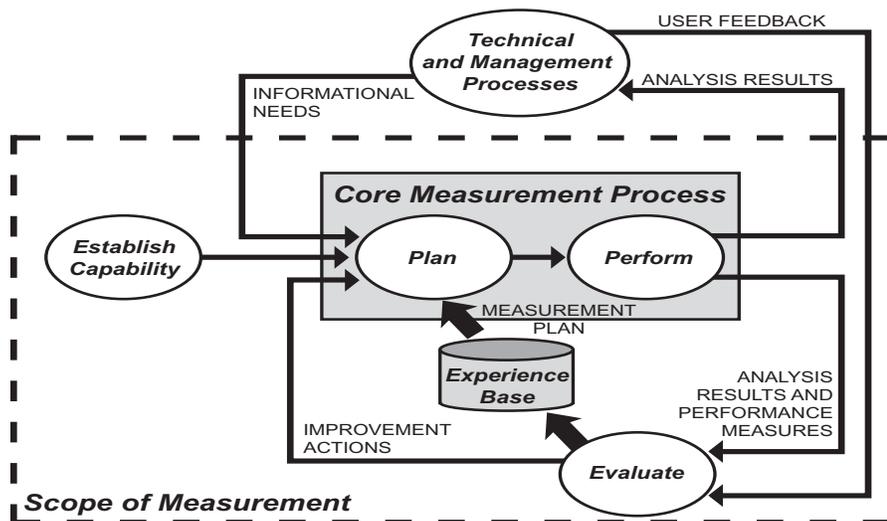
1. Identify Informational Needs for Current Management Practices

Organizations often develop a culture that encompasses technical management and engineering functions. This culture develops simultaneously as process, training, infrastructure, and habit evolve. For example, in a systems acquisition program, a monthly meeting may be held to review the supplier's progress and address potential risks. This meeting is a result of the culture and process within the organization, which has found that a periodic review of progress has led to a greater probability of on-time delivery.

From a measurement perspective, the periodic meetings represent a set of candidate informational needs that, when satisfied, makes the culture more effective and efficient. The information needed at the periodic meetings becomes a formal *information need* of the measurement process.

One of the primary barriers to measurement adoption is the misalignment between how managers work and what information the measurement process

Figure 1: Measurement Process Model



® Capability Maturity Model, CMM, Software Capability Maturity Model, and SW-CMM are registered in the U.S. Patent and Trademark Office.
SM CMMI Integration and CMMI are service marks of Carnegie Mellon University.

provides. By examining how managers really work, and using that to drive the measurement process, you are more likely to achieve greater measurement process adoption and success.

2. Identify Requirements' Measurements

Without exception, systems and software managers must measure the requirements engineering activities of the life cycle. Measuring requirements engineering activities involves quantifying the progression of software requirements from concept to formulation to design to test. Assessing these requirements ensures that your product contains all required functionality.

Typically, program plans and projections are based on estimates of software requirements, which are used as the basis for software size estimates. Because estimating requirements plays such a large part in developing the initial program plan, it is imperative to monitor that requirements are proceeding as expected. Consider a scenario where you are developing 25 percent more requirements than you planned – every life cycle activity may then be 25 percent or more over schedule and budget.

It is advisable to measure the number of requirements that each software process generates or accepts. Measure the number of system or top-level software requirements (i.e., *features* or *capabilities*), as well as the decomposition of system requirements into more detailed requirements. Measuring requirements helps you to keep tabs on the scope of your program. One of the most common issues detected by measuring requirements is *requirements creep*: the tendency to keep adding requirements to a program without considering how many additional resources or risks those new requirements represent.

In order to track differences between developed and planned requirements, it is necessary to also measure the status of each requirement as it moves through life cycle activities. A typical requirement status could be as follows: defined, approved, allocated, designed, implemented, tested, and verified. For example, in the CMMI requirements management process area, one of the typical work products identified for sub-process 1.3 Manage Requirements Changes is *requirements status*. A practical sample of how to define requirements status and then manage status changes can be found in

the “CMM Implementation Guide” [3].

A measure that shows the status of all requirements is essential to monitoring program status and acts as a scorecard to illustrate that requirements are being implemented. Early in the program schedule, ensure that requirements become defined, approved, and allocated as the system architecture is finalized. Near the end of the program schedule, you should see requirements move from implemented status to tested then to verified status. While valuable in detecting *requirements volatility*, this measure also supports monitoring effort, configuration management, and quality.

3. Identify Risks That Impacted Previous Programs

In most organizations, as well as in the experience of many managers, there is a

“By placing informational needs outside the scope of the core measurement process itself, the process can be used to support a wide range of management and executive decisions.”

history of project lessons that should not be repeated! This includes reasons that projects were never completed, or why projects were delivered late, over budget, and without needed functionality. These lessons learned in similar and recent software programs are prime candidates for measurement in the next (now current) program.

With historical risks, focus on identifying the cause of the risk rather than the symptom or the response. For example, with a project where the software was delivered late, try to remember and uncover the specific software components that led to the lateness. Perhaps

the reason for the program delay was that a key piece of commercial off-the-shelf (COTS) software was not available or that the integration took longer than expected.

Consider also that software managers are often aware of software problems but only react when a schedule delay is required. In our own software development, we have been *burned* by technical and schedule problems related to our COTS vendors, and we have essentially let their problems impact our projects. We now take an *act early* approach where our product manager immediately considers technical alternatives once a problem is identified. In your environment, consider whether you want to measure to *detect* technical problems for monitoring, or to *take management action* on known ones. In some cases, your measurement program will need to do both.

4. Identify Risks for the Current Program

During program planning, you may establish a risk management plan. For each risk, you typically develop risk identification, mitigation, impact, and probability. A measurement process can support a risk management plan by identifying risks that need to be mitigated and by quantifying the effect of mitigation activities. From a measurement perspective, risks can become informational needs that drive the measurement process. The measurement process can, and should, address these needs.

Since there are costs associated with measurement (as well as risk management), you may want to select a subset of risks to measure. You could use probability and estimated mitigation cost (or impact) as a discriminator in selecting risks to measure. For example, you may choose to measure only high- and medium-probability risks where the associated mitigation cost is greater than \$100,000.

A common risk within our organization is that software developers will not spend as much time as planned on a given product baseline. In the past, we found that senior developers were temporarily *borrowed* for other product development or support activities, for example, to investigate the cause of a field report from a customer. To address this risk, we developed an information need related to ensuring that resource expenditures correspond to our business priorities, i.e., first things first.

5. Measure What You Are Trying to Improve

It is the author's experience to frequently see organizations implementing large-scale, institutional change without implementing the corresponding means for managing the resultant process. Tom DeMarco, consultant and author, coined a phrase commonly heard in the world of software management: "If you don't measure it, you can't manage it" [4]. So, before you take a small step to improve an isolated software task, or a large step to improve an entire process, consider how you will demonstrate actual process improvement.

In addition to desiring to better manage your new or changed software process(es), be aware that there is an even more important reason to measure the processes that you are attempting to improve – to develop a quantitative understanding of why your software process behaves as it does. Developing this type of quantitative process understanding requires being able to mathematically describe the primary process factors. Once this mathematical relationship is established, the next step is to monitor and control the effects of these process factors. Furthermore, your estimates will become more accurate as a result of this better understanding.

Many measurement practitioners confuse a general quantitative understanding of their process with the quantitative management capability included in the CMM Level 4 Optimized. In practice though, organizations develop quantitative models for activities ranging from requirements engineering, inspections, defect detection and removal, to system testing software release activities – and few of them are rated at CMM Level 4. The point here is that by developing an understanding of your processes through measurement, you will be in a better position to estimate, control, and manage them, and less likely to rely on subjective guessing. (In other words, do not wait until you are attempting a CMM Level 4 assessment to start measuring; start now and you will be that much ahead of the game.)

6. Identify Software Quality Measures

Some years ago, a former market-leading technology company decided to counter its market slump by hiring a technology vice president. At the first

meeting of his direct reports, he walked around the table, put an airsickness bag in front of each person and said, "Your schedules make me sick." He went on to say that schedules without quality do not mean anything.

In essence, it does not matter how well you stick to the schedule if the system or software product is unusable by the customer. This vice president knew what the market demands – it is unfortunate that more companies do not take quality seriously. If they did, they would focus on building a quality product rather than racing to get a substandard product to market quickly.

System or software quality is more than measuring the quality of the end product. End product quality is the result of the systems and software quality activities employed during development. If you ignore the quality aspects of systems and software development,

“There is an even more important reason to measure the processes that you are attempting to improve – to develop a quantitative understanding of why your software process behaves as it does.”

it is anybody's guess what the quality of the end product will be.

One technique for addressing software quality is to use *quality gates*. This involves establishing reasonable and measurable thresholds at several points during development and then ensuring that the software or work products meet them before continuing. A quality gate could be all requirements approved, all unit tests passed, all code inspected, or all requirements (or subset) tested. Microsoft, for example, required developers to have no show-stoppers or priority one bugs in their code in order to release Windows NT to beta testing [5]. Microsoft managers plan on several zero defect releases during the development of a product. You should use the appropriate measures to

determine your progress in meeting one of these quality gates.

7. Identify Assumptions Used in Preparing the Project Plan

A typical systems or software development program plan includes a number of assumptions about progress, quality, and resources. Assumptions made during program planning are excellent informational needs for the measurement process. If the assumption is not realized, then many of the resulting schedule and resource plans may need to be examined, or re-planned.

For example, when performing an estimation using the Cost Constructive Model (COCOMO), the estimated number of lines of code (or other software sizing measure such as function points) is a primary driver in establishing the amount of effort. If your project exceeds the number of lines of code during development, this may indicate that more effort is needed during the *down stream* software activities such as unit testing, system testing, or integration.

8. Identify Resources Consumed and Products Produced to Understand Process Performance

At the beginning of initiating a measurement process, your organization will typically not have historical process data. In such cases, one of your goals should be to understand the behavior of the processes in the systems or software life cycle. Consider that each process in the life cycle consumes resources and produces a product (either an internal or a customer product). You should establish basic measurements to determine how many resources are being consumed and how much of the product is being produced. You might consider doing this for the processes that consume the majority of your budget or schedule.

9. Identify the Information Needed to Satisfy Organizational Policy

In many system or software shops, managers are required to use specific techniques in monitoring and controlling their programs. In large defense programs, for example, an earned value management system is required. When

managers are required to use specific management techniques, the organization should provide the data that managers need to effectively apply the technique. The measurement process is the method that the organization uses to deliver the information that managers need to use the technique.

For example, many defense organizations must use an earned value management system. In support of this, the measurement process should deliver required cost and schedule status information to the program in the form of cost performance index, schedule performance index, to-complete performance index, earned value (and its components), and variance at completion. Without a measurement process to ensure that the earned value data is collected, analyzed, and delivered in a timely fashion, even a very useful technique such as earned value can be inconsistently or (often) incorrectly applied.

Program or site policy and standards documentation provides many informational needs for the measurement process. While setting up a measurement process, analyze the systems and software management standards and policy to see what management techniques have been, or are being, mandated. Then, extract the informational needs from these mandated standards and address them during measurement process implementation.

Summary

Identifying informational needs is the first step in establishing an effective measurement process. The techniques explained above provide tools for extracting the real informational needs within your organization. Once all informational needs are identified, you can assign a relative priority to them, in case you need to balance the informational needs and the resources available to the measurement process. The measurement process will refine those informational needs into appropriate measurement activities, specific measures, and information products. Over time, you should review the effectiveness of the information products and individual informational needs as your organization adopts new technology and processes.

This approach for identifying informational needs ensures that the measurement information you and other managers receive is effective in helping you monitor and control your pro-

grams. By focusing measurement on true informational needs, managers are better armed to monitor and control their programs and to assess the likelihood of an on-time and on-budget completion. In addition, by saving managers time in gathering and analyzing the information they need to manage, managers can spend more time on their real role: decision making.◆

References

1. Department of Defense and U.S. Army. Practical Software and Systems Measurement Guidebook. Version 4.0b. Oct. 2000.
2. Software Engineering Institute. Capability Maturity Model® Integration for Systems Engineering /Software Engineering/Integrated Product and Process Development/Acquisition. Version 1.02d. Carnegie Mellon University. Dec. 2000.
3. Caputo, Kim. CMM® Implementation Guide. Addison-Wesley, 1998.
4. DeMarco, Tom. Controlling Software Projects. Prentice Hall, 1982.
5. Zachary, Pascal. Showstopper! The Free Press/Macmillan, 1994: 243-255.

About the Author



Peter Baxter is the development manager at Distributive Software, where he directs measurement services, products, and training. For the past eight years, he has assisted numerous government and commercial organizations in planning and implementing measurement programs. He is a frequent speaker and trainer on the subject of applying measurement to software, information technology, and systems engineering. He is the current chair of the International Council of Systems Engineering, Measurement Working Group and a member of ISO Subcommittee on Systems and Software Engineering. The author welcomes comments and discussion on this article.

Distributive Software
2300 Fall Hill Avenue, Suite 100
Fredericksburg, Virginia 22401
Phone: (540) 372-4500
Fax: (540) 372-6497
E-mail: pbaxter@distributive.com

COMING EVENTS

July 18-20

Shareware Industry Conference
 St. Louis, MO
www.sic.org

July 22-25

Joint Advanced Weapons Systems Sensors, Simulation, and Support Symposium (JAWS S3)
 Colorado Springs, CO
www.jawswg.hill.af.mil

July 22-26

6th Annual Practical Software Measurement Users' Group Conference
 Keystone, CO
www.psmc.com

August 19-22

The 2nd Software Product Line Conference
 San Diego, CA
www.sei.cmu.edu/SPLC2/

September 9-13

International Conference on Practical Software Quality Techniques (PSQT) 2002 North and International Conference on Practical Software Testing Techniques (PSTT) 2002 North
 St. Paul, MN
www.psqtconference.com

November 18-21

International Conference on Software Process Improvement
 Washington, DC
www.software-process-institute.com

April 28-May 1, 2003

Software Technology Conference 2003



Salt Lake City, UT
www.stc-online.org