Data Collecting and Recording – important steps in software quality measurement

Lect. Liviu Ion CIORA PhD., assist.lect. Ion BULIGIU
Informatics Economic Department, Economic Science Faculty of Craiova

The purpose of this paper is to describe some important phases in the process of measuring software performance, which can be applied in software settings for the achievement of technical goals and for business organizations. Although the concepts that are illustrated here are often applicable to individual projects, the primary focus is on the enduring issues that enable organizations to improve not just today's performance, but the long-term success and profitability of their business and technical endeavors.
We consider that data collecting and recording are the beginning phases in evaluating the quality of software systems and they represent the starting point in any analysis process and the basis for further studies.
Keywords: *software quality, resources, environment, costs.*

The fundamental reason for measuring software and the software process is to obtain data that helps us to better control the schedule, cost, and quality of software products. It is important to be able to consistently count and measure basic entities that are directly measurable, such as size, defects, effort, and time (schedule). Consistent measurements provide data for doing the following:

- Quantitatively expressing requirements, goals, and acceptance criteria.
- Monitoring progress and anticipating problems.
- Quantifying tradeoffs used in allocating resources.
- Predicting the software attributes for schedule, cost, and quality.

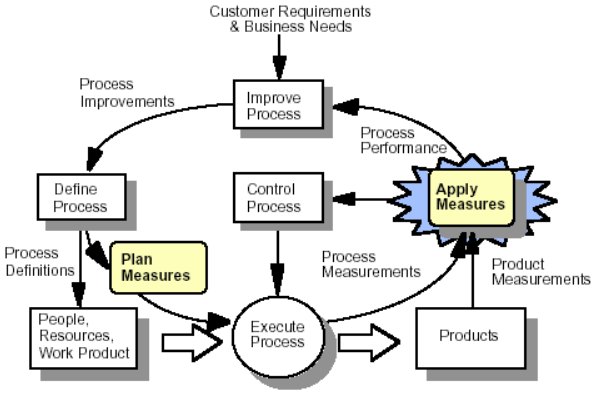

Fig. 1. How Applying Measures Relates to the Key Responsibilities of Process Management

In the process of software systems development, an important questions are how and under what conditions observations may contribute to a rational decision to change or not to change a process to accomplish improvements. A record of observations must accordingly contain all the information that anyone might need in order to make his own decision.
This process is about collecting and retaining data in the measurement planning activities associated with process management. In this paper, we turn our attention to applying the measures in ways that support

the decisions associated with process management. Figure 1 highlights the focus of our discussions.

Applying measures is the operational phase of the measurement process. It consists of collecting and retaining process management data, analyzing the data, and acting on the results. Collecting and retaining data are prerequisites for analysis. Analyzing the data in the context of the issues at hand then leads to using the results to control and improve the software process and its sub processes.

ACTIVITIES	Find Problems In:
Product synthesis	Requirement specs Design specs Source code User publications Test procedures
Inspections	Requirement specs Design specs Source code User publications Test procedures
Formal reviews	Requirement specs Design specs Implementation Installation
Testing	Modules Components Products Systems User publications Installation procedures
Customer service	Installation procedures Operating procedures Maintenance updates Support documents

To establish a software measurement environment, the software organization must define a data collection process and recording media. Software problem reports typically are the vehicles used to collect data about problems and defects. It is worthwhile to note that the data that is assembled as part of the problem analysis and correction process is precisely the same data that characterizes or gives attribute values to the problems and defects we wish to measure. Although this process facilitates the data collection aspects of software problem and defects measurement, the variety of finding activities and related problem reports make it difficult to communicate clearly and precisely when we define or specify problem and defect measurements.

The primary points of origin for problem reports are activities whose function is to find problems using a wide variety of problem discovery or detection methodologies, including using the software product (see Figure 2). During software development, these activities would include design and code inspections, various formal reviews, and all testing activities. In addition, activities such as planning, designing, technical writing, and coding are also sources of problems reports. Technical staff engaged in these activities frequently will encounter what appears to be an defect in a software artifact on which they are dependent to complete their work and will generate a problem report. Following product development, the software product customer is another source of problem reports.

To facilitate the communication aspect, we have identified five major finding activities:

- Software product synthesis[1]
- Inspections
- Formal reviews
- Testing
- Customer service

[1] By *product synthesis* we mean the activity of planning creating and documenting the requirements, design, code, user publications, and other software artifacts that constitute a software product. This would exclude all types of peer reviews.

This classification retains the functional identity of the finding activities without relying on a specific development process model, and therefore becomes a communicative attribute for problem or defect measurement.

Problem reports give rise to additional measurement communication issues. Problem reports generated by the finding activities are typically tuned to the needs of the activity and vary in content and format. For example, inspection reports, requirement review reports, test reports, and customer service reports carry data not required by or available to the others. The problems are recorded and reported at different points in time (e.g., before and after configuration management control), in batches or continuously, by different organizations, by people with varying degrees of understanding of the software product. Often, the data is captured in separate databases or record-keeping mechanisms. The problem and defect attributes and attribute values bridge these differences and provide a consistent basis for communicating (Figure 2).

In spite of the variances in the way software problems may be reported and recorded, there are remarkable similarities among reports, particularly if the organization or activity specific data is removed. There are several ways of categorizing this similarity. The approach we use to arrive at a set of attributes and attribute values that encompass the various problem reports is to apply the "who, what, why, when, where, and how" questions in the context of a software measurement framework.

In developing the attribute values, we are careful to ensure they are mutually exclusive; that is, any given problem may have one, and only one, value for each attribute. We also work to ensure that the values for each attribute are exhaustive so that inconsistent or erroneous counts do not occur.

We have identified the following attributes with the intention of transcending the variations in problem reports generated by the finding activities.

These attributes provide a basis for communicating, descriptively or prescriptively, the meaning of problem and defect measurements:

• **Identification:** What software product or software work product is involved?
• **Finding Activity:** What activity discovered the problem or defect?
• **Finding Mode:** How was the problem or defect found?
• **Criticality:** How critical or severe is the problem or defect?
• **Problem Status:** What work needs to be done to dispose of the problem?
• **Problem Type:** What is the nature of the problem? If a defect, what kind?
• **Uniqueness:** What is the similarity to previous problems or defects?
• **Urgency:** What urgency or priority has been assigned?
• **Environment:** Where was the problem discovered?
• **Timing:** When was the problem reported? When was it discovered? When was it corrected?
• **Originator:** Who reported the problem?
• **Defects Found In:** What software artifacts caused or contain the defect?
• **Changes Made To:** What software artifacts were changed to correct the defect?
• **Related Changes:** What are the prerequisite changes?
• **Projected Availability:** When are changes expected?
• **Released/Shipped:** What configuration level contains the changes?
• **Applied:** When was the change made to the baseline configuration?

General Principles

The operational activities of measurement begin with collecting and retaining data. The procedures that you defined for collecting and retaining data must now be integrated into your software processes and made operational. This means putting the right people, sensors, tools, and practices into the processes in the right places. It also means capturing and storing the data for subsequent use in analysis and process improvement.

The principal tasks associated with collecting and retaining data for process management are as follows:
• Design the methods and obtain the tools that will be used to support data collection and retention.
• Obtain and train the staff that will execute the data collection procedures.
• Capture and record the data for each process that is targeted for measurement.
• Use defined forms and formats to supply the collected data to the individuals and groups who perform analyses.
• Monitor the execution (compliance) and performance of the activities for collecting and retaining data.

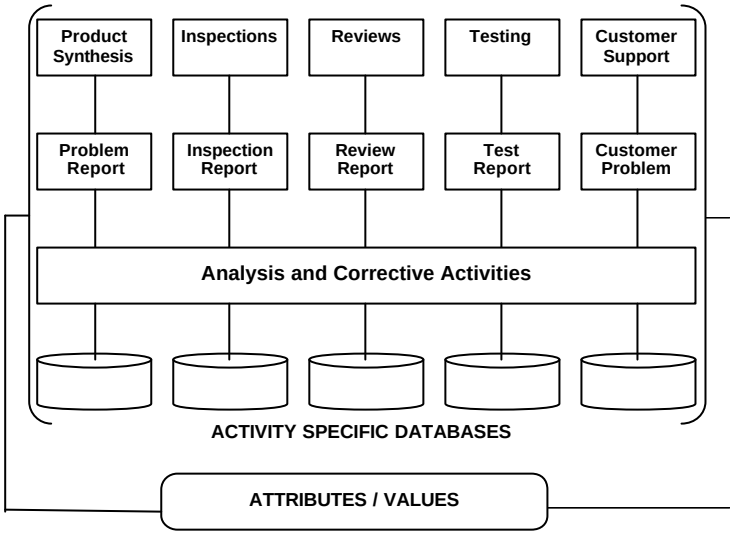

Fig. 2. Problem and Defect Data Collection and Recording

Collecting Data

Once you have selected and defined your measures and planned your implementation actions, you are ready to begin collecting data. Collecting data is more than just making measurements. It consists of implementing your plans, ensuring that they work, and sustaining the measurement activities that result. These actions will be facilitated if you document in detail your procedures for collecting, recording, and reporting data. Documenting your procedures involves:
• identifying the responsible persons and organizations
• specifying where, when, and how measurements will be made
• defining the procedures to be used for recording and reporting results
• providing standard "fill-in-the-blank" forms to simplify manual recording of the data

The complexity of your data collection processes will increase as additional organizations or software processes become involved. Each organization or process may use a different tool or method to obtain the "same" data. If you wish to compare or aggregate the data you collect, you must ensure that the methods different people use to collect the data are, in fact, collecting exactly the same kinds of data. That is, your different tools and procedures should be counting, extracting, or otherwise processing data in ways that produce equivalent results. When it is possible to do so, providing standard tools such as code counters and work breakdown structures will make achieving consistency and common understandings easier.

Having a data collection guide that fully describes your data definitions and collection processes will shorten the amount of time required to stabilize your measurement results. It will also improve the quality of the data you collect.

Collecting data is a process. Like any other process, it must be monitored to ensure not only that the data are being collected, but that they are timely, complete, authentic,

accurate, and otherwise of good quality. In short, if the results are to be reliable, the collecting process must be stable and under control. This implies that the performance of the measurement process itself should also be measured.

Process Performance Data

The fact that data will subsequently be analyzed can impose requirements on data collection in ways that may not be obvious. For example, measures of process performance that are used to assess process stability and capability require that special attention be paid to four important issues:

• **time sequence**. The order in which observations are made contains crucial information for estimating the inherent variability in a process. Moreover, knowledge of the sequence and of its relationships to time and process related events is what enables you to identify the point where assignable causes of variation entered a process. This helps greatly in identifying the causes and preventing recurrences. You should ensure that any measurements collected for estimating, controlling, or improving process performance are accompanied by records of the sequence of observations. Where feasible, it helps also to relate the sequence to time and to any events or milestones that might affect measured values.

• **context data**. Analyzing control charts requires information about the context in which the data were produced in order to properly interpret the record of performance that is plotted on the charts. Your data collection process must include practices that ensure that context data are captured when reporting product and process measurements.

• **rounding of data values**. Your collection processes must ensure that the scales for measuring and recording data are of appropriate granularity and that recorded values are not rounded inappropriately. Either condition can cause control charts to generate out-of-control signals even when the process is in a state of statistical control.

• **measurement stability.** When you institute new measures or revise existing measures, your data collection procedures may have to be changed as well. This can result in destabilizing the measurement process. Testing and evaluating new or changed data collection processes with pilot runs can help shake out problems with the procedures and avoid collecting and retaining inappropriate data.

Process Compliance Data

Collecting compliance data typically requires seeking out sources other than those available directly from measures of operating processes. Two ways to obtain process compliance data are enumerated below. As always, the types and kinds of data required, together with the cost and availability of data, will determine which methods are best for you.

1. One way to investigate the extent of compliance is to review the process(es) in question. Generally this means conducting a series of structured interviews with project personnel. The responses can be combined with reviews of other organizational data such as budgets, reports, policies, product characteristics, or process documentation to provide measures of both fitness and use. This approach is particularly useful when looking for potential causes of instability or excessive variability. It is also useful when establishing benchmarks for process improvement.

2. A second approach is to conduct periodic surveys where software managers and technical staff members answer questions regarding their compliance to the processes of interest. This approach is generally less expensive than the first approach, but the results may be less reliable due to differing interpretations and perceptions and to instincts for self-preservation among the people answering the survey. A well-designed survey can compensate for the inherent subjectivity in responses by ask-

ing questions that elicit related evidentiary data.

Process managers will often find it useful to conduct reviews or surveys to obtain a fitness profile of their project's personnel, tools, or methodologies. This can be especially true at the start of a project (or at periodic intervals in a long-term project) to help determine the training, tools, and technologies that may be needed to execute at the desired levels of process performance.

The use of surveys can be even more helpful at the organizational level. A survey is a relatively inexpensive and rapid method for gathering data to identify trends across projects. With careful survey design and by capturing relevant project, process, and environmental data and retaining the results over several survey periods, you can obtain trend information that can serve as a basis for stabilizing and sustaining process activities and for developing better tools and training. In addition, survey data can provide benchmarks and context data for project estimating and process improvement. At both project and functional levels, capturing process compliance data dynamically can alert managers to deteriorating compliance and help head off process instabilities and performance problems before they occur.

Retaining Data

Retaining data inherently involves creating and using one or more databases to organize and save the data for later use. Depending on the nature of your measurement activities, this may be a reasonably simple task or a very complex and technically demanding one. In either case, it is important to give serious consideration to the data retention system that will be employed. For example, while hard-copy forms may suffice for some data collection purposes, experience has shown that paper forms are often inadequate for retaining and aggregating measured results.

A personal computer database system and a full-functioned spreadsheet program may be sufficient for retaining and analyzing data for many processes. However, the size and complexity of the retention system will increase significantly if there is a need to support multiple projects or multiple organizations, or if you are using the measurement results for multiple purposes. The length of time you must retain the data can also influence your choice of a database system. For example, data for process management will often be retained and used well beyond the duration of individual projects.

A project management database, if it exists, may well serve as the basis for retaining process measurements for process management. This is something to be considered seriously before undertaking to develop a separate process management database, as there are often many overlaps between data collected for managing projects and data collected for managing processes.

You should consider the issues listed below when planning a process management database.

Database Planning Issues

Measurement Definitions

• The desire to standardize measures for data retention and analysis may conflict with software process tailoring or with the legitimate needs of projects to define and collect data in ways that address issues that are important to them. It may be unwise or impossible (or require added effort) to insist that all projects use the same measurement definitions.

• One alternative to standardizing measurement definitions is to permit freedom of definition (perhaps within prescribed limits), but to require standardized reporting via standardized formats for the definitions of the measures and measurement processes used.

Multiple Databases

• Do differing user needs, responsibilities, or levels of management require separate databases? If the answer is yes, a monolithic "software process database" is an

unlikely choice, and the following issues arise:
- How many databases?
- Who will operate them, and where?
- How will the databases be coordinated? (This involves addressing issues of concurrency, consistency, and propagation of data corrections and updates)

Database Design Goals (Recommendations)

• Capture and retain definitions and context descriptions, not just direct measurement data.
• Tie measured values to measurement definitions, rules, practices, and tools used.
• Tie measured values to the entities and attributes measured.
• Tie measured values to the contexts and environments in which they were collected (product, environment, and process descriptors; process and project status; time and place measured; method of measurement; and so forth).
• Accommodate process tailoring (by recording descriptions of process specializations, tailoring, and other differences among processes).
• Accommodate evolving measurement definitions and process descriptions.
• Address linking to, accessing, and coordinating with other databases, such as those used for time and cost reporting, cost estimating, configuration management, quality assurance, personnel, and so forth.
• Avoid storing indirect measures (such as defect densities and rates of change) that can be computed by users from directly measured results. There are three reasons for this advice:
- Storing computed values introduces redundancies in databases that are difficult to keep synchronized. When the underlying data change, indirect measures may not get recomputed.
- If only the results of computations are stored, essential information easily becomes lost.
- Other people may want to compute alternative indirect measures or compute them differently, so the underlying values will need to be retained anyway.

Logistical and Timeline Issues

• What are the media and mechanisms for moving data from the point of measurement to the database?
• How fast is the process from measurement to data entry? Will the data be timely and up to date?
• What are the provisions for coordinating the database with automated measurement tools? Can these provisions be automated?

Rules and Policy Issues

• What are your privacy objectives?
• What are your proprietary data objectives?
• What are your data access objectives?
• What are your retention objectives?
• What are your archiving objectives?

Database Operation Issues

Once you have settled the database planning issues, you should document your operational procedures in detail. This includes identifying:
• who will enter and maintain the data
• who can access the data
• levels of access—for example, you may not want certain financial data to be available to everyone who has access to staff-hour time records.
• where the data will be retained
• the tools you will use, including the editing and retrieval mechanisms.

Bibliography

1. Florac, William A. - *Software Quality Measurement: A Framework for Counting Problems and Defects, Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1997*
2. Florac, William A.; Park , Robert E.; Carleton Anita D. - *Practical Software Measurement: Measuring for Process Management and Improvement, 1997*
3. Grady, Robert B. - *Practical Software Metrics for Project Management and Process Improvement. Englewood Cliffs, N.J.: Prentice-Hall,1992.*
4. Musa, John D.; Iannino, Anthony; & Okumuto, Kazihra. Software Reliability Measurement, Prediction, Application. New York, N.Y.: McGraw-Hill, 1987.
5. Zahedi, Fatemeh – *Quality Information Systems, ITP Company, 1995*
6. IEEE Standard for a Software Quality Metrics Methodology (IEEE Standard P-1061/D21). New York, N.Y.: Institute of Electrical and Electronic Engineers, Inc., 1990.