KPA
Insights through analytics

The Israel ESPINODE

Software Process Improvement Series

Part II

EVALUATION OF PROCESS AND PRODUCT QUALITY[1]

A key element in the Software Quality Program introduced in Part I pertains to those activities necessary to evaluate the software development process and the resultant products. This element is referred to as the Software Quality Evaluation (SQE) program. SQE is a set of assessment and measurement activities performed throughout software development and maintenance to evaluate the quality of software products and to evaluate associated documentation, processes, and activities that impact the quality of the products. "Assessment" pertains to qualitative evaluation while "measurement" pertains to quantitative evaluation.

Measurement encompasses all quantitative evaluations, specifically, tests, demonstrations, metrics, and inspections. For these kinds of activities, direct measures can be recorded and compared against pre-established values to determine if the software meets the requirements. Accordingly, unit level testing, software integration, and software performance testing can be considered as measurement activities. Similarly, the output of a compare program or a path analyzer program can also be considered measurement.

Measurement also includes the use of metrics and measures that can be used to directly determine the attainment of numerical software quality goals. Measures of reliability, such as the number of faults per 1000 lines of source code (faults/KSLOC), are examples of such measures. In future parts of this series we will discuss the use of such measures.

On the other hand, any evaluative undertaking that requires reasoning or subjective judgment to reach a conclusion as to whether the software meets requirements is considered to be an assessment. It includes analyses, audits, surveys, and both document and project reviews.

---

[1] Adapted from the book by Kenett and Baker: **SOFTWARE PROCESS QUALITY: Management and control**, M. Dekker Inc., 1999

The "set of evaluation activities" refers to the actions, such as reviews, evaluation, test, analysis, inspections, and so on, which are performed to determine that (1) technical requirements have been established, (2) products and processes conform to these established technical requirements, and ultimately, (3) to determine software quality. The focus of these activities vary as a function of the stage of the development effort. For instance, peer reviews conducted during the software requirements analysis activities will focus on the adequacy of the evolving software processing requirements and their compatibility with the requirements for the design of the database(s). On the other hand, peer reviews conducted during detailed design will focus on how well the design of each unit of software is implementing the requirements allocated to it.

These evaluations may be performed by a number of different organizations or functions, some or all of which may be within the project organization. Furthermore, any one evaluation may be performed by different organizations or functions. As an example, the evaluators of a software requirements specification for a flight control system may include a flight controls engineer (to ensure that all the technical requirements have been implemented), a software engineer (to ensure that the requirements, as specified, can be implemented in software), a test engineer (to ensure testability), and Software Quality Assurance (to ensure that the overall requirements for the content of the document, as well as it's quality, have been addressed). The "set of evaluation activities" to be performed are generally documented in project plans, software development plans, project-specific software quality procedures, and/or company quality plans and related software quality procedures.

In the case of the products, determination of the software quality can be performed by comparing the products against pre-established criteria. However, evaluation of product quality is difficult, especially since the definition of software quality is hard to express quantitatively within the current state-of-the-art. A large volume of research has been directed toward the establishment of quantitative definitions of quality, for example, software reliability. We see much of this effort expressed in terms of software metrics. Numerous definitions for software metrics have been proposed and applied to specific cases. No one set of metrics will work for all cases. There is no silver bullet that will satisfy everyone's needs. (In later chapters of this book, we will discuss some of these metrics and measures, and illustrate how they can be applied).

On the other hand, the technology is available to establish and enforce various forms of meaningful quality criteria. The approach is relatively straight forward: Establish criteria based upon measurable entities; entities that lend themselves to validation during software development. As long as these criteria can be related to the attribute of quality desired ("quality is in the eye of the beholder"), a project can use them to gauge quality. For example, the cyclomatic complexity metric is a measurable entity. Experience indicates that a limit of 7 for cyclomatic complexity is wise to impose for units and aggregates of avionics software. At levels above 7, the software is more error-prone and more difficult to maintain. Accordingly, avionics software development projects would likely use this limit of the metric as one measure of the software product's quality.

The evaluation program also includes assessment and measurement of the software development and maintenance processes and the activities/ procedures comprising them. It may be

that the process is being properly implemented, but the products are not attaining the desired level of quality. These evaluations constitute a check of the existing process against the initial analysis performed prior to the start of the development, that is, during the methodology selection process discussed above. A principle of quality management is that product quality can be improved through the continuous improvement of the processes used to produce it. Continuous improvement is achieved by focusing on the processes, and using product evaluations, as necessary, as indicators of process adequacy. This evaluation may be implemented by examining interim software products, such as initial specifications, materials presented at walkthroughs or inspections, or the artifacts (products) which result from the process itself, such as Software Development Folders. Such measurements are aimed at determining the quality of the content of these products as the determinant of the process quality.

Generally, the basis for determining which process to implement has been to look at a candidate process or set of processes and evaluate the proposed process on the basis of its track record. The assumption is that if the process has been shown to produce "high quality software" in the past, then proper implementation of the process will result in a high quality product. This argument is somewhat misleading. There has been anecdotal evidence (for example: Software Engineering Institute presentation by David Zubrow to the Los Angeles Software Process Improvement Network (SPIN), 29 May 1996 on "Software Process Improvement: Business Impacts and Values") of the relationship, but no conclusive demonstration of a definite link between the process selected for the software development and the resultant quality of the software product itself. Establishment of such links has been more heuristic or intuitive rather than analytical. For example, the use of the Ada programming language promotes information hiding which, in turn, should make the software more adaptable. But, the actual cause and effect link has not been clearly demonstrated through hypothesis testing.

Typical of the difficulty in such research is the work of the Department of Defense (specifically, the Air Force) in defining a software quality framework leading to software quality metrics (see. for example: Arthur, James D. and Richard E. Nance. "Developing an Automated Procedure for Evaluating Software Development Methodologies and Associated Products", Technical Report SRC-87-007. System Research Center, Virginia Polytechnic Institute, Blacksburg, VA). Upon close examination, it may be seen that this framework actually measures the adherence to good programming practices rather than the actual quality of the software. Other work attempting to link processes to product quality can be found in Bowen, Thomas P., Wigle, Gary B., and Tsai, Jay T. "Specification of Software Quality Attributes", Technical Report RADC-TR-85-37, Rome Air Development Center, Griffis Air Force Base, Rome, NY.

What is crucial to any software development project is the definition and implementation of the activities necessary to assess and measure the quality of the software products produced by that project, in accordance with the requirements established for the project. Equally crucial is the definition and implementation of the activities necessary to evaluate the adequacy of the processes used by that project to produce the software products.

Part III of this series will provide some examples of organisations that achieved, through process improvements, gains both in quality and productivity.