

Using Product Quality Assessment to Broaden the Evaluation of Software Engineering Capability

Robert A. Martin and Mary T. Drozd

Presented At

The 1996 Software Engineering Process Group (SEPG) Conference

"Broadening the Perspective for the Next Century"

20-23 May 1996
Trump's Worlds Fair Casino Hotel
Atlantic City, New Jersey



Sponsored by the

**Software Engineering Institute
and the
New Jersey SPIN**

MITRE

The MITRE Corporation
202 Burlington Road
Bedford, MA 01730-1420

Introduction

This paper will discuss an approach to evaluating an organization's software engineering capability that focuses on the degree to which development processes exhibit characteristics that reduce cost and risk across the entire life-cycle of a system. In today's dynamic software development climate, with its ever-increasing pressure on efficiency and economic payoffs, there needs to be an increased focus on the quality of the systems that are being developed. Augmenting the evaluation of an organization's process with an analysis of the quality of the software products that are produced by that process can identify areas where process improvement would result in improved product quality. Whether an organization conducts an evaluation of their own processes as a basis for process improvement planning, or an outside organization conducts such an evaluation to analyze the risks associated with contracted software development efforts, the planning and measurement of process improvement should not just focus on plotting an organization's progress against a model of process maturity; increased efficiency and product quality are the real objectives.

While appraisals based on the Software Engineering Institute's (SEI) Capability Maturity Model (CMM) have gathered many advocates, process maturity is only one facet of an organization's software engineering capability. Our experience indicates that coupling process evaluation with an assessment of the quality of the products that are produced by that process provides a more accurate analysis of an organization's present capability. Analysis of software product quality provides insight into the effectiveness of an organization's processes and identifies areas where changes in an organization's process can positively impact product quality. Both the planning of process improvement and the measurement of software engineering capability should include more than an assessment of progress against the CMM maturity scale. In the enthusiasm for CMM-based process improvement, we must not forget that process efficiency and product quality are the real goals.

A Definition of Software Product Quality

Product quality is frequently thought of as "absences of defects," or as "conformance to requirements," or as "customer satisfaction." This view of software quality focuses on the characteristics of a product relative to current needs and requirements. Since most of the systems being developed today are expected to evolve over time, it is critical that the software also lend itself to modification, thereby reducing the effort needed to rework systems to accommodate new or changed requirements. In addition, the increased focus on reuse as a means to reduce cost and time to field increases the value of software components that can be easily integrated in new systems.

ISO 8402 defines quality as: "The totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs."

In the context of this paper, software quality will refer to those *characteristics of a software product that minimize life-cycle cost and enhance the reusability of the software components*. This definition is based on the notion that need for software that is easy to modify and reuse is implied, even when not explicit in requirements, given the continuous evolution and changing focus of the user's needs.

The Relationship of Software Quality to Process Improvement

Mature processes help ensure consistent quality of products, but they are not sufficient. Coupling process evaluation with an assessment of the quality of the products that the process produces can provide a more accurate analysis of an organization's capability than only evaluating the processes. Ignoring the quality of the products developed by a process can lead to an incomplete understanding of the risks that a development effort presents. Ignoring product quality can also result in the misallocation of resources in process improvement planning.

Design and improvement of the processes for building software systems generally focuses on increasing the efficiency of the processes by eliminating rework. If software product quality is understood as referring to those product characteristics that minimize life-cycle cost by reducing rework, then a high quality product provides objective evidence of a mature process. However, the converse is not true. In some cases, an organization's processes may not deal with the entire system life-cycle. For example, a supplier may be concerned with developing and delivering a system that meets a well-defined set of requirements, which may not explicitly address factors that will facilitate future modification, enhancement, or reuse. In such cases, the delivered product may not meet software quality standards despite a well defined, mature development process that is designed to deliver a product on schedule and within cost. Minimizing rework that may be needed later to satisfy evolving requirements is not within the scope of the supplier's process; therefore, assuming that a mature process will produce a quality product is not valid when examined from a life-cycle perspective. Since suppliers may not be motivated to design their processes to minimize system life-cycle cost, the acquirer must determine the risks associated with the product quality that is typically provided by their supplier and then ensure that any process changes needed to reduce life-cycle cost and risk are implemented.

MITRE's software product quality assessment is based on the measurement of product attributes that highlight life-cycle risks and their causes. Management can use this analysis to direct appropriate corrective action to reduce life-cycle costs and risks. When a product assessment is coupled with a process assessment, the product assessment results must be interpreted in the context of the scope of the processes that are being evaluated; not all software development processes are intended to produce products that meet the software quality standards, as defined in the MITRE assessment method.

The Background of Software Quality Assessment

Software quality assessment is a field that is receiving increasing attention as the drive for systematic quality assurance gathers momentum. There is general consensus within the field on the framework elements needed to measure the quality of software products, which include documentation as well as code.

The original analysis of software quality assessment structures was done by Boehm and associates at TRW (1 - Characteristics of Software Quality) and subsequently revised and incorporated in a Rome Air Defense Center (RADC) report (2 - Factors in Software Quality) by McCall and others in 1978. The basic structure involves quality attributes that are associated with quality factors, which are decomposed into particular quality criteria and lead to quality measures. This framework has been studied and discussed in detail, most recently in complete form by Kitchenham and various co-authors (3 - The Architecture of Software Quality, 4 - Towards a Constructive Quality Model, 5 - Quality Factors, Criteria, and Metrics) in connection with the ESPRIT work that was a major input to the standards work that produced the IEEE standard (6- A Standard for Software Quality Metrics Methodology). Work establishing the

relationships between quality factors to real-world system concepts, such as reliability and maintainability is currently in various stages of analysis or trial on a variety of projects.

A detailed discussion of the foundations of MITRE's approach to software quality assessment, as well as a detailed description of the method can be found in the proceedings of the 1996 Software Engineering & Economics Conference. The following section will provide a brief summary of MITRE's assessment methodology.

The MITRE Software Quality Assessment Method

In 1991, MITRE began development of a methodology for evaluating the quality of software products. We leveraged many of the previous efforts referenced above. Our work focused on the central issues of product quality with respect to life-cycle maintenance. One of our major contributions was to revise the original frameworks to reflect modern software engineering concepts, and to add automation support wherever possible. As defined and practiced, MITRE's assessment methodology, which is referred to as the Software Quality Assessment Exercise (SQAE), is consistent with the current draft of the ISO standard for product evaluation (ISO/IEC 14598).

In designing the SQAE methodology, we focused our attention on the development of a language- and system-independent framework designed to address the issues associated with repairing latent defects, implementing new functionality, and rehosting a product throughout the life-cycle. Concepts and criteria presented in the earlier works have been generalized and modernized to capitalize on open standards and commercially available analysis tools. To ensure meaningful and repeatable results, the SQAE uses analysts' observations, rather than a simple numerical score in the compilation of results, and rigid scoring criteria and guidance to constrain the subjectivity of the evaluation.

The SQAE methodology establishes a hierarchy in which abstract, high-level concepts are defined by more tangible, measurable factors. The term *quality area* refers to the high-level concepts, which relate to sources and types of life-cycle risk. Although most software engineers agree that these qualities are characteristic "good" software systems, there is no general consensus on the definition of the terms or on how to determine the existence of a particular feature. To constrain the subjectivity in the terms used to designate *quality areas*, each area is implicitly defined as the summation of the measurable *quality factors* that collectively address the concepts inherent in the area.

As shown in Figure 1 below, each *quality factor* contributes to one or more *quality areas*. The weighting of each factor's contribution was initially empirically derived based on the assessors' perception of the software quality for sample programs. Subsequent analysis refined and refocused the weightings of both attributes and factors. Each factor is further defined by a set of quality attributes; these attributes are measurable questions or metrics that explore different facets of how the factor is implemented in a specific product. The following list provides an example of representative attributes for each of the quality factors:

Consistency: Have the project products (code and documentation) been built with uniform style to a documented standard?

Independence: Have ties to specific systems, extensions, etc. been minimized to facilitate eventual migration, evolution, and/or enhancement of the product?

Modularity: Has the code been structured into manageable segments which minimize gross coupling and simplify understanding?

Documentation: Is the hard copy documentation adequate to support maintenance, porting, enhancement and re-engineering of the project?

Self-Descriptiveness: Does the embedded documentation, naming conventions, etc. provide sufficient and succinct insight into the functioning of the code itself?

Anomaly Control: Have provisions for comprehensive error handling and exception processing been detailed and applied?

Design Simplicity: Does the code lend itself to legibility and traceability where dynamic behavior can be readily predicted from static analysis?

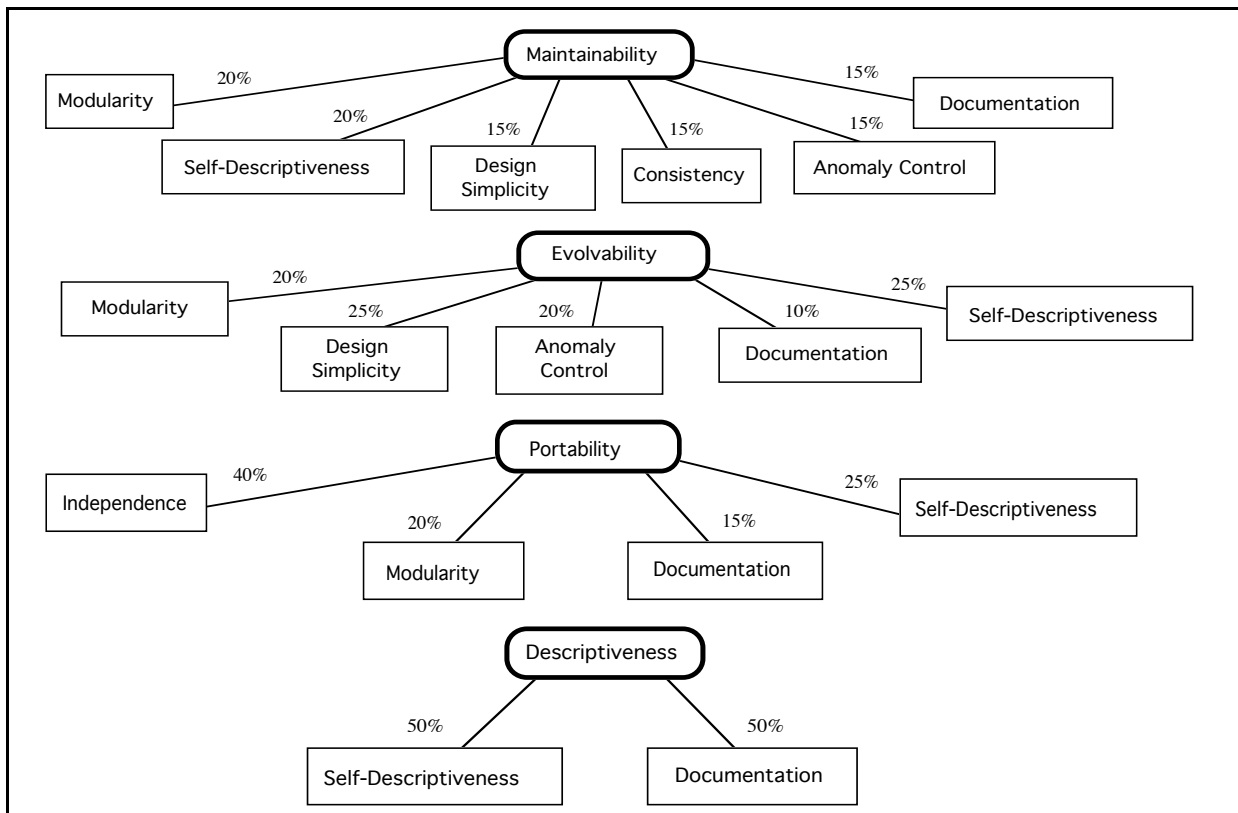


Figure 1: Quality Areas' Weighting of Quality Factors

The process for determining the quality of a system's product(s) includes both automated and semi-automated assessment. Custom tools have been developed to handle a wide range of languages. These tools were designed to preclude the necessity of having a completed system available. All attributes that characterize each factor are evaluated by an analyst responding to specific questions. The scope of review varies for different attributes: for some attributes, where evaluation is fully automated, all code is reviewed; where evaluation is performed by an analyst, a representative sample consisting of at least the seven largest, seven smallest, and seven average size files is used. Documentation quality and consistency attributes are evaluated qualitatively, as excellent, good, marginal, poor. When the analysts have answered all questions for each attribute, scores for each quality factor are calculated based on the evaluation of the underlying attributes. The final step in the assessment is to identify risk drivers and mitigators based on the analysts' comments, observations, and scores for each attribute within the factors.

Experience in Coupling Product and Process Assessments

During the 1992 - 1994 time frame, we participated in various evaluations of both the processes and quality of the products produced by numerous software engineering organizations. Having observed that the results obtained sometimes appeared to be inconsistent, we analyzed the cause of the apparent inconsistencies and determined that processes, which appeared to conform to the SEI's CMM standard, lacked scope; that is the processes did not include all the elements that were needed to ensure the delivery of quality products. A simple example would be that although a software development plan was prepared and followed, the design and coding standards included were not comprehensive; checklists used by quality assurance and peer review processes were similarly deficient. Therefore, despite well-defined processes, the quality of the product was poor. In an effort to better integrate the results of these process and product evaluations, we defined an approach, which we pilot tested on two programs. On these pilot programs, we coupled a CMM-based evaluation of processes with a MITRE SQA assessment of the software products.

The combined process and product assessment had three goals. The first was to provide the sponsor with more information than was typically provided by a standard SCE. The second goal was to have greater confidence in the accuracy of the findings by using the results of the product assessment as objective evidence of process compliance. Our third goal was to perform the combined assessment within the same elapsed time and at lower cost than a "standard" SCE.

The pilot assessments were conducted in a source selection context; this mandated a highly disciplined approach, as well as agreement on the evaluation criteria that would be used for reporting results. A small team (three MITRE staff) performed the on-site process evaluation in parallel with the product assessment, which was conducted off-site by two analysts. Preliminary findings were exchanged while the on-site team was still at each developer's facility. This provided an opportunity for the on-site team to resolve any issues arising from inconsistencies between the findings relative to process and product quality. The results of the process assessment were reported as strengths and weaknesses relative to the CMM. The overall assessment of risk combined with the assessment of product quality findings relative to whether the scope and effectiveness of the processes in use provided a firm basis for the proposed approach for the new development effort. The pilot assessment was documented as a point of departure for the further refinement of the approach (7 - The Augmented Software Capability Evaluation Lessons learned on a Pilot Program).

The core of the process assessment was a CMM-based Software Capability Evaluation (SCE). This provided the discipline and structure needed for the source selection context. The scope of the process assessment included a thorough investigation of each organization's effective use of currently available tools and methods, as well as their training and experience in software development technologies proposed for use on new efforts. The product assessment focused on the SQA's standard four quality areas: maintainability, portability, evolvability, and descriptiveness.

Our experience in product and process assessments of approximately two dozen past development efforts indicates that coupling the process and product assessments produces a more accurate understanding of a software development organization's capabilities than the use of the standard SCE, which uses process maturity as the sole criteria for evaluating software engineering capability. Our experience has shown that a "good" process, which appears to meet the goals of the CMM, can produce poor products, and that a "bad" process (i.e., a process that fails to meet the goals of the CMM), is capable of producing "good" products due to factors such as highly skilled individuals or a strong cultural bias toward good software engineering practices.

Additionally, we found that stable processes that are primarily manual may be less efficient and constitute greater risk to program success than less mature processes that have extensive support of state-of-the-practice methods and automated tools.

Our overall goals for coupling process and product assessments were met. The assessments were completed at a rate of two organizations every three weeks. The cost to perform the combined product and process assessment was low due to a relatively small team of five experienced individuals with travel costs for only the three members of the team that performed the on-site process evaluation. The scope of the insight gained and the synergy achieved by coupling the assessments increased our confidence in the accuracy of the results and provide data to our sponsor that would not have been otherwise available. Subsequent efforts have continued our work in applying the combination of process and product assessments to acquisitions.

Opportunities for Applying Combined Process and Product Assessments

Coupling software process and product assessments can provide valuable insight under a variety of circumstances. During source selection, combining assessments can determine whether an organization's institutionalized processes, as applied on past projects, provided rigorous enforcement of project standards. This can be accomplished by examining the consistency of the quality of the products that were produced with each project as well as across different project teams. A second application in source selection is the assessment of demonstration software and processes, including tools and methods used to produce the demonstration system. In this context, the primary objective of the assessments is to determine whether the proposed combination of processes, tools, and methods were effective in producing high quality software. In both cases, risks associated with software quality are identified prior to the beginning of development. This provides the acquirer an opportunity work with the supplier to ensure that process improvements are implemented as needed to avoid similar problems in the software that will eventually be delivered.

Another situation that provides a particularly appropriate opportunity for coupling process and product assessments occurs at the down-select point in procurements that include a competitive phase during which more than one contractor develops a subset of the system requirements. Since the software developed during the competitive phase is generally used as the basis for development of the deliverable system, the quality of the software is a critical element in determining, and subsequently managing, risk. The competitive pressure that this type of contract creates can tempt contractors to take short cuts in order to maximize the functionality that is demonstrated at the down-select. Assessment of product quality can identify deficiencies that will increase the risk of using the product as the basis for further development. In addition, problems with product quality can also provide an indication of lack of commitment to disciplined development practices and rigorous process enforcement. Conducting a combined process and product assessment as input to the down-select decision can identify risks early in the follow-on development while there is an opportunity to modify the processes to prevent quality problems propagating throughout the software during the follow-on development effort.

A forth application is the use of periodic process and product quality assessments to monitor the quality of products as development progresses. Where analysis of product attributes indicates that product quality is not satisfactory, process improvements can be implemented. This application is particularly useful for systems that are developed incrementally since problems detected in one increment can be prevented in successive increments. This approach is fully consistent with process improvement concepts. Assessing product quality is a mechanism for identifying defects. Causal analysis can then identify areas where process improvement is needed to prevent defects in the future.

Conclusion

Increasing pressure to manage system development and acquisition from the perspective of reducing life-cycle cost and risk has resulted in increased interest in the quality of the software products an organization delivers. Our experience suggests a mature process that is consistent with models of good software engineering practice does not guarantee quality software products. A product quality assessment performed in conjunction with a process assessment can help in determining whether an organization's processes can be expected to reliably produce quality products from a life-cycle perspective.

When a software development effort is contracted, this approach can help identify risks by providing insight into a development organization's software engineering capability from a point of view that complements that of process maturity. If a process evaluation can ensure that an organization's way of developing products has a firm foundation in processes used in the past, then the quality of the products that have been produced in the past provides an excellent predictor of the product that will be delivered on a new contract. During development, early samples of products can be assessed to identify weaknesses; so that development process controls can be put in place to ensure that the weaknesses are addressed.

This approach can also be applied to process improvement by using product quality as a measure of process maturity. As software engineering organizations achieve the goals of the higher CMM maturity levels, coupling of product quality assessment with process assessments can be used to refine process improvement efforts. The benefits and importance of ensuring software quality will increase as processes to create and reuse software components evolve.

In summary, coupling product quality assessment with process assessments provides a valuable method for determining the degree to which an organization's processes reliably produce quality products that reduce the cost and risk across the entire life-cycle of the system. Based on our experience, this approach provides a well defined, repeatable, cost effective method to evaluate software engineering capability and extends the benefits of process improvement by using product quality as a measure of process maturity.

ACKNOWLEDGMENTS: This work was funded by the MITRE Corporation.

BIBLIOGRAPHY

- (1) Boehm, B. W., J. R. Brown, M. Lipow, G. J. MacLeod, and M. J. Merit, "Characteristics of Software Quality", New York: Elsevier North-Holland, 1978.
- (2) McCall, J. A., P. K. Richards, and G. F. Walters, "Factors in Software Quality", Volumes I, II and III, US. Rome Air Development Center Reports NTIS AD/A-049 014, NTIS AD/A-049 015 and NTIS AD/A-049 016, U. S. Department of Commerce, 1977.
- (3) Kaposi, A. and B. Kitchenham, "The Architecture of Software Quality", Software Eng J, Vol. 2, No. 1, Jan. 87, 2-8.
- (4) Kitchenham, B., "Towards a Constructive Quality Model", Software Eng J, Vol. 2, No. 4, July 87, 105-123.
- (5) Kitchenham, B., L. M. Wood, and S. P. Davies, "Quality Factors, Criteria and Metrics", ESPRIT Report R1. 6. 1, REQUEST/ICL-bak/028/S1/QL-RP/01, 1985.
- (6) "A Standard for Software Quality Metrics Methodology", IEEE Computer Society Standard P1061, 1991.
- (7) Drozd, M. T., "The Augmented Software Capability Evaluation Lessons learned on a Pilot Program", MITRE Technical Report, June 1995.