Modeling the Cost of Software Quality

by

Stephen T. Knox

ABSTRACT

This paper offers an extrapolation of the manufacturing and service industries' Cost of Quality Model to the business of software development. The intent is to provide a theoretical account of the changing quality cost structure as a function of a maturing software development process. Thus, the trends in expenditures due to the four major quality cost categories --- appraisal, prevention, internal failures, and external failures --- are presented over the five levels of software process maturity, according to the Software Engineering Institute's (SEI's) Capability Maturity Model for Software (CMM). The Software Cost of Quality Model conservatively proposes that the total cost of quality, expressed as a percentage of the cost of development, can be decreased by approximately two-thirds as process maturity grows from Level 1 to Level 5 of the SEI's CMM.

INTRODUCTION

Two questions often asked of quality function professionals by a software project manager are, How much will working on these quality processes cost me? and What can I expect in return for my investment? The manager recognizes that to implement a quality improvement project, resources must be allocated toward processes not currently being undertaken, and prior management experience has proven that usually the resources available are barely adequate to meet aggressive project and schedule deliverables. Also implicit in the manager's questions is the expectation of some point of diminishing returns: Even if there is benefit from an investment in quality-related work, help me understand the point at which the investment will be more costly than what I can get in return.

Background --- The Traditional Cost of Quality Model

The concerns expressed by our present-day hypothetical software manager are the same concerns expressed by industrial management during the 1950s. At that time, the quality function professionals saw the need to extend quality attainment efforts beyond the traditional inspection and test activities to the processes further upstream in the manufacturing and product development groups. Quality function managers, hoping to increase the scope of the quality effort, were faced with the task of convincing upper management of the necessity to allocate additional resources to quality attainment. Management demanded that the quality function quantitatively demonstrate the amount of resource investment that was necessary and the expected return on that investment.

The quality function professionals responded by developing an investment model that expressed quality in terms of costs --- the cost of attaining quality (the investment) and the cost of not attaining quality (the return). Their argument was that moderate increases in the former (typically, appraisal processes, such as inspection and test, and some defect prevention processes) would result in significant decreases in the latter (e.g., defects, scrap, repair and warranty costs), up to some point of diminishing returns. The traditional Cost of Quality Model shown in Figure 1 graphically represents their investment model.[1] The three curves portray moderate increases in prevention and appraisal costs resulting in dramatic decreases in failure costs. The point of inflection in the total cost of quality quadratic curve represents the point of diminishing returns on quality investment.

Figure 1 reflects the belief of the 1950s' quality function professionals that attaining 100 percent conformance to specification would be prohibitively expensive. The rationale was that zero-defects production would require extensive testing and inspection at every point in the design, manufacture, and delivery process. Consequently, they conceived of a point of diminishing returns on quality-related investments. This point of maximum quality attainment for the minimum amount of investment is exactly the point of interest to our hypothetical software manager.

The modeled point of diminishing returns, however, was not verified by empirical cost of quality data.[2,3,4] In actual practice, investment in quality attainment shifted from appraisal to prevention processes as the quality function moved upstream into the manufacturing process and product design groups. Defect prevention processes, such as statistical process control and robust product designs, actually reduced the overall cost of attaining quality, contrary to the expectation of the quality function of the 1950s. Designing durable products to delight customers and manufacturing these products in a well-controlled environment resulted in fewer defects at the point of final inspection. Thus, appraisal costs were reduced significantly. (The author has participated in cases where successful application of defect prevention processes led to the complete elimination of expensive inspection and test.[5])

The Revised Cost of Quality Model

The quality function managers of the 1950s could not conceive of a quality investment model that did not rely heavily on inspection and test. Actual experience, however, uncovered that an increased emphasis on defect prevention processes led to significant reductions in appraisal costs and, in some cases, eliminated final inspection. The empirical cost of quality data resulted in a revised model, published in 1988.[2] As shown in Figure 2, the Revised Cost of Quality Model extracts the point of diminishing returns.

The three curves express the changing quality cost structure as quality attainment efforts shift from appraisal processes to the processes designed to achieve higher-quality output before final product test. In the revised model, the costs due to defect appraisal and defect prevention rise moderately as investments are made to improve product quality. The moderate increases in the costs of appraisal and prevention result in dramatic decreases in the failure costs. Unlike the corresponding curve in Figure 1, appraisal and pr do not increase exponentially, since the means of quality attainment shifts from defect appraisal to defect prevention. The total cost of quality curve in Figure 2 consistently decreases as quality improves; therefore, the curve does not have a point of diminishing returns.

The Software Cost of Quality Model

The Revised Cost of Quality Model has been used extensively in the manufacturing and service industries as a benchmark against which actual quality costs are compared. The model has thus helped organizations identify opportunities for continuous improvement.[4] Also, a leading government research corporation, MITRE Economic Analysis Center, recently advocated using this method for reducing the cost of quality in software development.[6] What is lacking, however, is a model of quality costs in the domain of software development.

Important differences exist between the domains of the industrial environment and the software development environment. While an extrapolation of the Revised Cost of Quality Model can be made to monitor software quality costs (as suggested by MITRE), the author believes greater detail on and adjustments to the cost trends are required to account for differences between the domains. This paper presents a model that incorporates these differences. The Software Cost of Quality Model offers a rationale that addresses the reasonable concerns expressed by our hypothetical software manager.

MODELING THE COST OF SOFTWARE QUALITY

As background for a discussion of the Software Cost of Quality Model, this section deals with the subject of attaining software quality cost data and lists the software quality cost categories.

Software Quality Cost Data

Whereas the literature has sufficient data to support estimates of the costs related to not attaining software quality (e.g., defect and software maintenance costs), the author has been unable to locate rigorous accounting of costs related to attaining quality (e.g., testing and defect prevention). This is not surprising, given the relative lack of cost metrics tracked in software development. Capers Jones asserts that full quality costs have been tracked in some projects; in a personal conversation with the author, Jones cited his own work at International Telephone and Telegraph (ITT).[7] Other consulting firms (e.g., Computer Power Group) reported to the author that some clients kept limited metrics of defect costs. In follow-up investigation, however, the author has not found any rigorous accounting of defect appraisal and defect prevention costs in software development.

Consequently, the Software Cost of Quality Model offered in this paper extrapolates two key concepts from Gryna's Revised Cost of Quality Model (shown in Figure 2): (1) moderate investments in quality attainment result in a significant decrease in the cost of not attaining quality, and (2) an emphasis on attaining quality through defect prevention processes results in an overall decrease in the cost of traditional testing activities.

Software Quality Cost Categories

Following the modern trend in the industrial and service industries, the Software Cost of Quality Model subdivides the driving cost elements into four categories: appraisal and prevention (the costs of attaining quality, i.e., the investment), and internal failures and external failures (the costs of not attaining quality, i.e., the return).[2,3,4] Table 1 provides some examples of these elements in software development. The list of elements within each cost category is meant to be exemplary, not exhaustive.

Table 1   Software Quality Cost Categories

Appraisal	Prevention	Internal Failures	External Failures
Unit/Integration Testing	Contextual Inquiry/ Quality Function Deployment (QFD)	Defect Management	Problem Report Management
Quality Assurance	Project Management	Test Failure Rework	Warranty Rework
Field/Acceptance Tests	Requirements Management	Design Change Rework	Customer Support

Audits/Assessments	Formal Inspections	Requirement Change Work	Lost Market Share

Appraisal Costs. Traditionally, the costs associated with appraisal activities are those incurred by product inspection, measurement, and test to assure the conformance to standards and performance requirements. In software development, these costs are usually related to the various levels of testing and to audits and assessments of the software development process. Appraisal costs also include costs (e.g., quality assurance) incurred by organizations that provide test support and/or monitor compliance to process standards.

Prevention Costs. While appraisal costs are those used to find defects, prevention costs are those incurred by process improvements aimed at preventing defects. The examples of prevention costs listed in Table 1 are the costs that worried our hypothetical software manager, because for the most part, defect prevention processes in software are not traditional. Such processes are perceived as "front-loaded" processes, which lengthen the initial development schedule and threaten the probability that a project will deliver on the scheduled target date. Ironically, field testing (an appraisal cost) and the subsequent rework of found defects (internal failure costs) are traditionally accepted by software managers as legitimate yet frustrating tasks in the development cycle. One goal of software defect prevention processes is to reduce (and possibly eliminate) the need for expensive field testing.

Internal/External Failure Costs. Failure costs are primarily due to the rework, maintenance, and management of software defects. Internal failures are software defects caught prior to customer release, whereas external failures are detected after release. Consistent with the initial cost of quality findings in the manufacturing industry data, the majority of quality costs in software are incurred by internal and external failures. The literature indicts the rework from software defects as the most significant driver of all development costs. Independent studies show costs associated with correcting software defects that range from 75 percent of the development effort at General Motors, to an average of 60 percent for U.S. Department of Defense projects, to an average of 49 percent, as reported in a survey by 487 respondents from academia and industry.[8,9,10]

THE MODEL

Figure 3 depicts the Software Cost of Quality Model. The curves represent how the quality cost structure changes as a software development environment improves its capability to deliver a high-quality, bug-free product. Whereas the x-axes in Figures 1 and 2 reflect improving process capability in an industrial environment, the x-axis in Figure 3 is based on the Software Engineering Institute's (SEI's) Capability Maturity Model for Software (CMM).[11] The Software Cost of Quality Model incorporates the CMM, which offers a descriptive road map for improving software development processes. The details of this road map provide a rationale for theorizing the changing quality cost structure within the domain of software development.

The Maturing Software Development Process

The CMM is too extensive to describe fully in this paper. (Humphrey presents a detailed accounting.[12]) The central concept of the CMM is that a software development environment has a measurable process capability analogous to industrial process capability. In the software domain, process capability can be measured through assessment. The CMM proposes five levels of capability, ranging from the chaotic, ad hoc development environment to the fully matured and continually optimizing, production-line environment.

The SEI estimates through their assessment data that most software development environments are at the initial, chaotic level of capability. The SEI has also declared that although some individual projects show the attributes of the highest level of capability, no organization measured has demonstrated full maturation. Since no organization has made the journey to full maturation, and since scant data exists on the appraisal and prevention costs as they apply to software development, the Software Cost of Quality Model uses CMM Levels 1 to 5 as the discrete milestones at which the appraisal, prevention, and internal and external failure cost trends can be theorized.

Software Cost of Quality Model Assumptions

Before the cost trends in Figure 3 are examined in detail, two data-driven assumptions need to be declared. First, the total cost of quality (the sum of the costs associated with appraisal, prevention, internal failures, and external failures) at CMM Level 1 is equal to approximately 60 percent of the total cost of development. This assumption is based primarily on internal failure cost data taken from the literature and external failure cost data tracked at Digital. The estimate of internal failure costs comes from recent data collected by Capers Jones. The data indicates that software rework due to internal failures consumes 30 to 35 percent of the development effort for projects the size of those typical at Digital.[13] The lower range of this figure has been added to the cost of the Customer Support Center (CSC) management of external failures, which an unpublished study by the Atlanta CSC estimates to be 33 percent of the development costs (available internally only, on TPSYS::Formal_Inspection, Cost of a Software Bug, Note 31.0). Thus, the estimate of a total cost of quality equal to 60 percent of the development cost is based on the sum of the estimates of just two of the many cost elements, namely, rework due to internal failures and CSC management of external failures.

The second assumption is that the total cost of quality will decrease by approximately two-thirds as the development process reaches full maturity, i.e., CMM Level 5. This assumption is based on normative case-study industrial data cited by Gryna.[2] The data details the recorded change in the total cost of quality at the Allison-Chalmers plant during seven years of its quality improvement program.[14] Table 2 summarizes the reduction in the total cost of quality at Allison-Chalmers and relates this reduction to a similar change theorized in the Software Cost of Quality Model.

Table 2 Reduction in Total Cost of Quality (TCQ)

	Allison-Chalmers (% of Cost of Sales)	Software Cost of Quality Model (% of Cost of Development)
Initial TCQ	4.5	60.0
Improved TCQ	1.5	18.0
TCQ Decrease	67.0%	67.0%

Although it may be unwise to assume that a normative trend for the manufacturing industry can be applied to software development, note that the assumed two-thirds decrease in the total cost of quality is more conservative than the estimates of SEI's Dr. Bill Curtis. He claimed return on investments (ROIs) in the range of 5:1 to 8:1, as an organization progresses in process maturity.[15] (Note: These claims have received empirical support from Quantitative Software Management [QSM] Associates, who report measured decreases in required effort and overall development cost on the order of 5:1.[16])

THE CHANGING COST STRUCTURE

Given the two grounding assumptions just discussed, the paper now presents a theoretical view of the changing cost trends between Level 1 and Level 5. The theory is based on the expected returns on investing in process maturity as outlined by the CMM. This section examines the details of Figure 3.

CMM Level 1

The SEI estimates that 90 percent of the software organizations today are at Level 1, which is characterized by an ad hoc, undefined, and sometimes chaotic development environment, highly dependent on heroic individual effort to meet delivery dates. Little attention is given to fundamental process management in this highly reactive atmosphere, and rework to correct internal and external failures is often perceived as necessary "fire fighting" to avoid disaster. At this level, the major costs of software quality are due to rework and maintenance. Testing is sporadic, so appraisal costs are minimal and most defects are experienced by the customers, resulting in expensive warranty costs and loss of market share. The costs associated with defect prevention approach zero.

CMM Level 2

A software organization at Level 2 has instituted the fundamental processes to manage resources, artifacts, and change. Project management, configuration management, and requirements management are the key processes that characterize a CMM Level 2 development environment that is, at the least, repeatable. In Figure 3, appraisal and internal failure costs increase at this level, primarily due to the formation of a quality assurance organization that monitors compliance to proscribed testing standards. Since, at Level 2, the organization applies testing activities more rigorously, more defects are found and reworked internally.

The increased testing activity and additional resources allocated to fix defects cause the apprehension that our hypothetical software manager expressed earlier. The manager experiences fear and uncertainty about being able to fix all the found defects and deliver the product on the scheduled date. Although our hypothetical software manager is probably aware that adherence to rigorous testing results in fewer defects shipped to the customer, a manager's success is often measured on the ability to deliver a product on time. The reduction in external failure costs at Level 2 occurs too late in the process to mitigate the career risk of seriously missing the delivery date.

CMM Level 3

According to the CMM literature, the major gains at Level 2 are the creation of repeatable processes that provide the base underpinning of a maturing development environment. Figure 3 illustrates that the investments to improve quality have been primarily in the appraisal category. But at CMM Level 3, the development environment has achieved a point of stability. A defined, documented framework exists within which the creative act of software design can be executed in a controlled manner. Quality attainment now emphasizes investing in the prevention activities, such as Contextual Inquiry into customer problems and Formal Inspections of specification and design documents. Such prevention processes are intended to ensure a more accurate understanding of and a greater conformance to customer requirements. Investing in prevention results in a steep decline in the external failure costs and gaining back lost market share.

Our hypothetical software manager is entitled to be more than skeptical about such claims; however, empirical data substantiates them. For example, Figure 4 details the 66 percent increase over projected revenue for VAX RALLY version 2.0, a direct result of improvements made to earlier versions --- improvements suggested by the Contextual Inquiries conducted with VAX RALLY version 1.0 customers.[17] Figure 5 clearly demonstrates that Contextual Inquiry leads not only to increased revenue but to the higher productivity and lower defect density experienced by POLYCENTER System Census version 1.0, when compared to four other system management applications.[18] These applications, represented in Figure 5 as A, B, C, and D, were developed without the use of this critical defect prevention process.

While generally considered to be part of the appraisal process, Formal Inspections, when applied to source documentation such as specifications and design, are similar to process control monitors. These inspections ensure that critical functionality is not omitted as the development process proceeds from the stated requirement for a solution to the specification and design of that solution. The effectiveness of the Formal Inspection process in preventing potential inconsistencies and omissions accounts for its rating as the most efficient defect removal method, as shown in Table 3.[19] Thus, applying Formal Inspections as a defect prevention process means fewer defects to test and fix internally and a more satisfied customer using the product.

Table 3 Defect Removal Efficiencies

Method	Efficiency (Percent)
Formal Inspections	65
Informal Reviews	45
Unit Testing	25-50
System Testing	25-50
Regression Testing	20-50
Field Testing	30
Beta Testing	25

The data in Table 3 is not intended to fully account for the magnitude of the trends at Level 3. Rather, the data offers a rationale for the overall direction of these trends. If a disparity exists between the data and the acceleration of decreasing failure costs in Figure 3, bear in mind that the model is the more conservative estimator.

CMM Levels 4 and 5

Although it has seen evidence of CMM Levels 4 and 5 in a few discrete projects (e.g., one Japanese project reported to be at Level 5), the SEI reports that it has not yet measured a Level 4 or a Level 5 organization. At these higher levels of maturity, the dominant cost of quality is due to the prevention elements, primarily from the cost elements of metric-driven continuous improvement and process control. The software process at these levels has become so well characterized by metrics that it has achieved a state where development schedules are predictable. Requirements are now understood quantitatively. The costs attributable to traditional appraisal activities, especially field testing, are dramatically decreasing, since product quality can now be appraised by monitoring the development process as opposed to expensive testing of the product. By Level 5, appraisal and failure costs have dropped to the level expected of a Six Sigma organization. The model proposes that the total cost of quality has decreased by approximately two-thirds, which is consistent with the normative industrial data.

CONCLUSION

This paper is not an endorsement of the SEI's Capability Maturity Model for Software, which is used here to describe discrete states within a maturing software development process. Although the CMM offers a rational, staged approach to achieving a predictable and highly productive development environment, the CMM is not the only road map to improving Digital's software process. For example, the variety of customers served in commercial software development offers special challenges to ensure that these customers' work experiences are brought into the design and development process. The CMM does not detail Voice of the Customer processes, which are practiced increasingly at Digital. In addition, some key processes specified for CMM Levels 3, 4, and 5 (e.g., Formal Inspections and metric-driven Continuous Improvement) are effective in reducing defects. These processes are already used in many of Digital's organizations, even though it is doubtful that any of the software development groups at Digital would be assessed as being beyond CMM Level 2.

The author believes that CMM Level 5 is the goal, regardless of the road map for attainment. The Software Cost of Quality Model explored in this paper offers the same argument for improving process capability that was offered in the manufacturing industries: the major costs of quality are the waste and the resource loss due to rework, scrap, and the lost market share when products do not possess the quality to address the problems faced by customers. The key to reducing quality costs is to invest in defect prevention processes, many of which are detailed by the CMM.

So, the response to the initial concern expressed by our hypothetical software manager is the following: You will not experience a point of diminishing returns from investing in quality-attaining processes. Certainly, there is a steep learning curve, and the short-term gains are not apparent. Given the software life cycle, most of the short-term gains will be experienced after the development is complete and the product has been shipped.

Since investments in quality, however, are not meant to realize quick, dramatic returns, the defect prevention processes probably offer the most immediate visible evidence that the overall cost of quality has been reduced. Yet, regardless of whether the investment is made according to the CMM road map or using some other quality attainment plan, software managers must keep in mind that quality attainment processes require a great deal of hard work. Also, the investment must be constant to achieve the significant, long-term payback, as reflected in the Software Cost of Quality Model.

REFERENCES

1. J. Juran and F. Gryna, Quality Planning and Analysis (New York: McGraw-Hill, 1980).

2. F. Gryna, "Quality Costs," Juran's Quality Control Handbook, 4th ed. (New York: McGraw-Hill, 1988).

3. J. Campanella, Principles of Quality Costs, 2d ed. (Milwaukee, WI: ASQC Quality Press, 1990).

4. J. Atkinson et al., Current Trends in Cost of Quality: Linking the Cost of Quality and Continuous Improvement (Montvale, NJ: National Association of Accountants, 1991).

5. S. Knox, "Combining Taguchi Signal-to-Noise Metrics with Classical Regression for Robust Process Optimization," Technical Report ISB Quality (Marlboro, MA: Digital Equipment Corporation, 1990).

6. E. deGuzman and T. Roesch, "Cost Analysis Provides Clues to Spot Quality Problems," The MITRE Washington Economic Analysis Center Newsletter (McLean, VA: April 1993): 2.

7. C. Jones, Applied Software Measurement (New York: McGraw-Hill, 1991).

8. B. Boehm, Software Engineering (Redondo Beach, CA: TRW, 1976).

9. J. Hager, "Software Cost Reduction Methods in Practice," IEEE Transactions on Software Engineering, vol. 15, no. 12 (December 1989).

10. W. Goeller, "The Cost of Software Quality Assurance," 1981 ASQC Quality Congress Transactions, San Francisco, CA (1981): 684-689.

11. M. Paulk, B. Curtis, and M. Chrissis, "Capability Maturity Model for Software," Technical Report CMU/SEI-91-TR-24 (Pittsburgh, PA: Software Engineering Institute, Carnegie-Mellon University, 1991).

12. W. Humphrey, Managing the Software Process (Reading, MA: Addison-Wesley, 1989).

13. E. Yourdon, The Decline and Fall of the American Programmer (Englewood Cliffs, NJ: Yourdon Press, Prentice-Hall, 1992).

14. O. Kolacek, "Quality Cost --- A Place for Financial Impact," Transactions of the 1976 Annual Conference of ASQC, Milwaukee, WI (1976): 131.

15. B. Curtis, "The Superior Software Organization," Software Process Improvement Network Meeting, Boston, MA (January 1993).

16. L. Putnam, "The Economic Value of Moving up the SEI Scale," Technical Report QSMTR93-01 (McLean, VA: Quantitative Software Management, Inc., 1993).

17. J. Neilsen, Usability Engineering (New York: Academic Press, 1993): 4.

18. S. Knox, Newsletter of Continuous Improvement for Networked Systems Management, vol. 2 (Maynard, MA: Digital Equipment Corporation, April 1993).

19. C. Jones, "Software Metrics and Total Quality Management," Case Outlook, vol. 6, no. 4 (1992): 1-11.

TRADEMARKS

BIOGRAPHY

Stephen T. Knox Steve Knox is a principal software engineer with the Software Engineering Technology Center. Currently, he is assigned to the Networked Systems Management organization to improve software and development processes. Steve came to Digital in 1989 from Tektronix, Inc., to further develop the Contextual Inquiry process. A quality engineer certified by the American Society of Quality Control, Steve received the 1991 High Performance Systems Technical Leader Award. He holds an M.S.

(1986) in psychology from Portland State University.