An Economic Model of Software Quality Costs
AMEL KOLAŠINAC, LJUBOMIR LAZIĆ, DŽENAN AVDIĆ
Faculty of Technical Science
University of Novi Pazar
Vuka Karadžica bb,36300 Novi Pazar
SERBIA

Abstract: - As of today there is very little knowledge is available about the economics of software quality. The costs incurred and benefits of implementing different quality practices over the software development life cycle are not well understood. There are some prepositions, which are not being tested comprehensively, but some useful Economic Model of Software Quality Costs (CoSQ) and data from industry are described in this article. Significant research is needed to understand the economics of implementing quality practices and its behaviour. Such research must evaluate the cost benefit trade-offs in investing in quality practices where the returns are maximized over the software development life cycle.

Key-Words: - Quality Cost Model, Software Quality, TQM

1 Introduction
Most important thing in analysis of the cost of quality is visibility in development cycle. It is the visibility gained form cost of quality analysis that enable the QA people involved to focus their attention on those activities which discover, and correct the root cause of the software defects. This root cause analysis (through Pareto technique or any other method) allows the QA people to determine how the development process can be improved to prevent major area cause of defects. There are many definitions of quality being given by experts that explains quality for manufacturing industry but still unable to define it with absolute clarity for software engineering [1]. In cost quality analysis we should identify what we are trying to achieve, the goals should be defined and that should be measurable. So that analysis can verify that it is actually increasing the quality level or not.
There is very little research is available about what quality initiative should be taken and how it reduces your development cycle and improves product quality. Significant research is needed to understand the economics of implementing quality practices and its behaviour.
Main focus in quality management is how to make profitable decisions on quality expenditures [2-6].
With respect to quality initiatives we can divide the organizations into two categories, one are those who do not believe that process improvement and training of the human resources would bring in any improvement in quality. They think this is extra cost. Whereas second kind of organizations who have realized the importance of processes and its continuous improvement, plus good care of the staff and their knowledge upgrading.
Organizations those who have realized the importance of high quality and process efficiency normally find it difficult to start the improvement cycle [6-9]. They find it difficult to convince their top management to allocate budget for the quality initiatives. The real issue is of investment, not the cost. The investment in software quality, like any investment has an immediate cost, with an expected net payback. There is where Quality Cost Analysis could be used as effective tool to make them understand the ROI [8-9]. As we all know that top management does understand the language of money very well. They would like to increase sales and have more profits. Task of QA people is to relate the Cost of quality as investment and its benefits are increased revenue.
Case studies of the success stories can be presented to the top management as tool to increase their understanding high quality and how to go about that. The major problems we see with these case studies are that there is no local research available [8-9]. All material available discusses organizations in West. The possible argument of the top management could be; that these practices do not suits our culture or our environment. What is required is the research in preparing the local case studies and research for the organizations that have implemented the TQM and those who have not. The comparison of both will provide good

1 This work was supported in part by the Ministry of Science and Technological Development of the Republic of Serbia under Grant No. TR-13018.
starting point for the management of such organization. There are some prepositions, which are not being tested comprehensively, but some useful Economic Model of Software Quality Costs and data from industry are described in this article. Significant research is needed to understand the economics of implementing quality practices and its behaviour. Such research must evaluate the cost benefit trade-offs in investing in quality practices where the returns are maximized over the software development life cycle.

2 Software Quality Dimensions and Models

2.1 Software quality

Although no standard industry definition exists for what constitutes good quality in software, it is generally taken to mean that a software product provides value (satisfaction) to its users, makes a profit, generates few serious complaints, and contributes in some way to the goals of humanity (or at least does no harm) [1]. Software quality is difficult to define because there is no single comprehensive and complete standard definition of its lexicon. Various aspects and terms are found in sources such as ISO 9000-3, Institute of Electrical and Electronics Engineers Software Engineering Standards, and various books on the subject.

The following are the key dimensions of software quality.

• Level of satisfaction: The degree to which customers or users perceive that a software product meets their composite needs and expectations.
• Product value: The degree to which a software product has value for its various stakeholders relative to the competitive environment.
• Key attributes (“ilities”): The degree to which a software product possesses a combination of desired properties, e.g., reliability, portability, maintainability.
• Defectiveness: The degree to which a software product works incorrectly in target user environments due to debilitating operational defects.
• Process quality: In relation to the development process by which the product is produced, it means good people doing the right things in an effective way.

A definition fashioned from the above aspects should be created for your organization and for each project. Every application or business domain faces a specific set of software quality issues, and software quality must be defined accordingly. For example, mission-critical applications have extremely stringent operational needs, whereas typical information system applications must focus on general measures of customer satisfaction. It also is important for each software development project to define its specific meaning of software quality during the planning phase. Such a definition contributes to the basis for setting objectives and practical measures of quality progress and determination of readiness for release to customers.

The new standard SQuaRE consists of 14 documents grouped under five thematic headings:

• Quality Management, defining all common models, terms and definitions referred to by all other standards in the SQuaRE series,
• Quality Model, probably updated version of ISO/IEC 9126-1
• Quality Measures, derived from ISO/IEC 9126 and ISO/IEC 14598,
• Quality Requirements, standard for supporting the specification of quality requirements, and
• Quality Evaluation, providing requirements, recommendations and guidelines for software product evaluation.

2.2 Comparison of the Quality Models

The following table from [2] compares characteristics of different quality models. The table illustrates the characteristics and their updates during the last 30 years. ISO 9126 in the table is based on revision from 1998, which is version between ISO/IEC 9126:1991 and ISO/IEC 9126:2001.

TABLE1: Comparison between the quality models [2]

<table>
<thead>
<tr>
<th>Quality Characteristic</th>
<th>Boehm</th>
<th>McCall</th>
<th>FURPS</th>
<th>ISO 9126</th>
<th>Drummy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Testability</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Correctness</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Efficiency</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Understandability</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Reliability</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Flexibility</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Functionality</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Human Engineering</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Integrity</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interoperability</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Process Maturity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maintainability</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Changeability</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Portability</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Reusability</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

3 COST OF SOFTWARE QUALITY (COSQ)

3.1 Economics of Quality - Literature review
Cost of quality represents any and all costs that organization incurs from having to repeat a process more than once in order to complete the work correctly. Cost of quality (CoQ) is an accounting technique introduced by Juran in 1951 as a means of providing justification to management for investments in process improvements. Cost of software accounting (CoSQ) is useful to enable our understanding of the economic trade-offs involved in delivering good-quality software. Commonly used in manufacturing, its adaptation to software offers the promise of preventing poor quality but, unfortunately, has seen little use to date [3]. Different authors and researcher have used different ways to classify components for quality cost, if we look carefully their understanding about various components are approximately the same.

Pressman [7] has divided the cost of quality into Prevention, appraisal, and failures:

$$\text{CoSQ} = \text{Cost Conformance} + \text{Cost non-conformance}$$

As explained by Rex Black [4] that: “Investing in Software Testing, decrease The Cost of Software Quality”. He has mentioned costs of quality into two major types: conformance and nonconformance as shown on Fig. 1.

![Fig. 1 The Cost of Software Quality](image)

Dan Houston [3] has defined Cost of quality in his article "Cost of Software Quality: A Means of Promoting Software Process Improvement" as follows:

$$\text{CoSQ} = \text{Prevention Cost} + \text{Appraisal Cost} + \text{Intern. Fail. Cost} + \text{External Failure Cost}$$

By now we have clear understanding of four components of the Quality cost. With the help of these four components we will discuss the theoretical model suggested by researcher based on the results gathered from the manufacturing industries. Following is (Fig. 2) graphical presentation of the CoSQ given by most researchers [2-3], [5-6]. The graph in Fig. 2 is showing that for achieving high reliability, close to red dot (almost zero defect) the cost is very high but achieving a reasonable level (area between two green dots) of quality does not require very high cost. To remove defect after reaching at very low defect density the cost of detection would be very high (Rs.500/KLOC) whereas the defect detection was relatively easy as numbers of defect were high (high defect density) the cost to remove defect is approximately 10 times lesser. Cost mentioned on the graph are imaginary numbers just to give an idea that cost of defect removal at high defect density would be lower and cost at low defect density would be high.

![Fig. 2. The cost of high reliability](image)

The costs of achieving quality and the costs due to lack of quality have an inverse relationship to one another: as the investment in achieving quality increases, the costs due to lack of quality decrease. This theoretical model is shown below in Fig. 3. This shows that as appraisal and prevention cost increases, the failure cost will decrease until an optimum point is reached. After this optimum point, the increase in appraisal will no be offset by the decreased in failure cost.

![Fig. 3. Model of software quality](image)
Researcher have noticed that in the initial phase appraisal measures cause internal failure to increase as these measures detect more errors at early stages, but error removal at early stage is much cheaper compare to error removal at later stage. But overall appraisal activities decrease external failure as a result total failure decreases. A small increase in prevention measures will normally create a major decrease in total quality cost.

3.2 Quality Cost Analysis

The objective of the quality cost analysis is not to reduce the cost, but to make sure that the cost spent are the right kind of cost and that maximize benefit derived form that investment. Traditional view of the cost of quality revolved around failure related activities. Due to quality cost analysis the major emphasis has been shifted to prevention and appraisal. As we all know that corporate understand the language of money, quality cost analysis emerged the concept of studying quality related cost as means of communication between the quality staff department and company managers. Challenge is how do you go about taking economic considerations into account when designing or modifying a system?

• How do you account for the costs involved?
• How can costs and benefits be "traded-off" against quality attributes or functionality?

A cost benefit analysis is done to determine how well, or how poorly, a planned action will turn out. Although a cost benefit analysis can be used for almost anything, it is most commonly done on financial questions. Since the cost benefit analysis relies on the addition of positive factors and the subtraction of negative ones to determine a net result, it is also known as running the numbers.

A cost benefit analysis finds, quantifies, and adds all the positive factors. These are the benefits, and then it identifies, quantifies, and subtracts all the negatives, the costs. The difference between the two indicates whether the planned action is advisable. The real trick to do a cost benefit analysis well is making sure you include all the costs and all the benefits and properly quantify them.

The key consideration in any analysis of the cost of quality is visibility. It is the visibility gained form cost of quality analysis that enable the QA people involved to focus their attention on those activities which discover, and correct the root cause of the software defects. This root cause analysis allows the QA people to determine how the development process can be improved to prevent further defects. Following is the graph, that is showing the theoretical model of CoSQ, adopted form [3]. CoQ is a proven technique in manufacturing industries both for communicating the value of quality initiatives and for indicating quality initiative candidates. CoSQ offers the same promise for the software industry, but has seen little use to date. Initial uses of CoSQ show that it can be a very large percentage of development costs, 60 percent or higher for organizations which are unaware of improvement opportunities. CoSQ has demonstrated its value in measuring the ROI of a software improvement program across the software industry.

4 Hypothetical case study

Let's use a hypothetical case study to illustrate the use of this cost of quality technique to analyze return on the testing investment. Suppose we have a software product in the field, with one new release every quarter. On average, each release contains 1,000 “must-fix” bugs—unacceptable defects—which we identify and repair over the life of the release. Currently, developers find and fix 250 of those bugs during development, while the customers find the rest.

Suppose that you have analyzed the costs of internal and external failure. Bugs found by programmers costs $10 to fix. Bugs found by customers cost $1,000 to fix.

As shown in the “No Formal Testing” column in Fig. 5, our cost of quality is three-quarters of a million dollars. It’s not like this $750,000 expenditure is buying us anything, either. Given that 750 bugs escape to the field, it’s a safe bet that customers are mad!

Suppose we calculate that bugs found by testers would cost $100 to fix. This is one-tenth what a bug costs if it escapes to our customers. So, we invest $70,000 per quarterly release in a manual testing process. The “Manual Testing” column shows how profitable this investment is. The testers find 350 bugs before the release, which cuts almost in half the number
of bugs found by customers. This certainly will make the customers happier. This process improvement will also make the Chief Financial Officer happier, too: Our total cost of quality has dropped to about half a million dollars and we enjoy a nice fat 350% return on our $70,000 investment.

### Table: Testing Investment Options: ROI Analysis

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Staff</td>
<td>$0</td>
<td>$60,000</td>
<td>$60,000</td>
</tr>
<tr>
<td>Infrastructure</td>
<td>0</td>
<td>10,000</td>
<td>10,000</td>
</tr>
<tr>
<td>Tools</td>
<td>0</td>
<td>0</td>
<td>12,500</td>
</tr>
<tr>
<td>Total Investment</td>
<td>0</td>
<td>70,000</td>
<td>82,500</td>
</tr>
</tbody>
</table>

**Fig. 5. Using Cost of Quality to Analyze Return on Investment**

In some cases, we can do even better. For example, suppose that we invest $150,000 in test automation tools. Let’s assume we intend to recapture a return on that investment across the next twelve quarterly releases. Would we be happy if that investment in test automation helped us find about 40% more bugs? Finding 500 bugs in the test process would lower the overall customer bug find count for each release to 250. Certainly, customers would be much happier to have the more-thoroughly tested system. In addition, cost of quality would fall to a little under $400,000, a 445% return on investment (ROI).

This is a huge improvement over the initial situation. We are realizing a quantifiable and substantial return on our testing investment. We are also making our customers happier.

### 5 CONCLUSION

CoQ is a proven technique in manufacturing industries both for communicating the value of quality initiatives and for indicating quality initiative candidates. CoSQ offers the same promise for the software industry, but has seen little use to date. CoSQ is a technique that is most useful in enabling our understanding of the economic tradeoffs involved in delivering good quality software.

**References:**


