Failure Costs of Software Quality

Ayşe Betül Cengiz[1] and Kökten Ulaş Birant[1]

[1] Dokuz Eylul University, Izmir 35390, Turkey

Abstract. Many studies have been conducted on software quality costs in the literature and industry. As a result of these studies, it has been seen that software can cause costs not only during production but also after it is delivered to the customer. This situation required software quality assurance (SQA) teams to focus on post-design as well. There are many different approaches to software error quality costs in the literature. This paper is a short research paper on software quality costs with a particular emphasis on the failure costs of software quality. In addition, since the mentioned subject was at the beginning of the industry, the approaches in the industry were also mentioned.

Keywords: Failure Costs of Software Quality, Cost of Software Quality, CoQ.

1 Introduction

The concept of Cost of Quality is an accounting technique first introduced by Juran in 1951 [1]. In 1979, Crosby developed this concept and made the definition of "to get management's attention and to provide a measurement base for seeing how quality improvement is doing" [2].

Although this concept has been developed for the business world, it has been made applicable to all areas such as the software world. There is a cost of activity in every project such as developing, testing, etc. In order to optimize the value of the business, these costs should be determined correctly. There should not be unnecessary testing or less testing. Otherwise, for the first case, it causes unnecessary delays and ends up incurring more costs, and for the second case, there may be a chance of defective products being handed to the end-users.

Cost of quality is calculated on two main bases: conformance and nonconformance, or, in terms of software, cost of control and cost of failure. Although this paper will focus on the cost of failure, the concept of cost of control will be briefly mentioned as it is relevant.

In the second part of this article Cost of Quality is presented from an overview. Then, in the third part, our main topic, Failure Costs of Software Quality, is mentioned. Although the model based on the third part is explained, there are many models in the literature. These models are mentioned in the fourth part. In the fifth part, the results of a general literature study are presented.

2 Cost of Software Quality

As Slaughter et. al. says, quality costs are important because every dollar and labor hour not spent on rework can be used for making better products more quickly or for improving existing products and processes. As mentioned previously, the cost of software quality (CoSQ) involves two other costs: the cost of control (CoC) and the cost of failure (CoF). We can basically show as in Eq (1).

$$CoSQ = CoC + CoF \tag{1}$$

Moreover, CoC and CoF have some sub costs. While CoC consists of prevention and appraisal costs, CoF has the external failure, internal failure, technical debt, and management failure costs (see Fig. 1) [3-9].

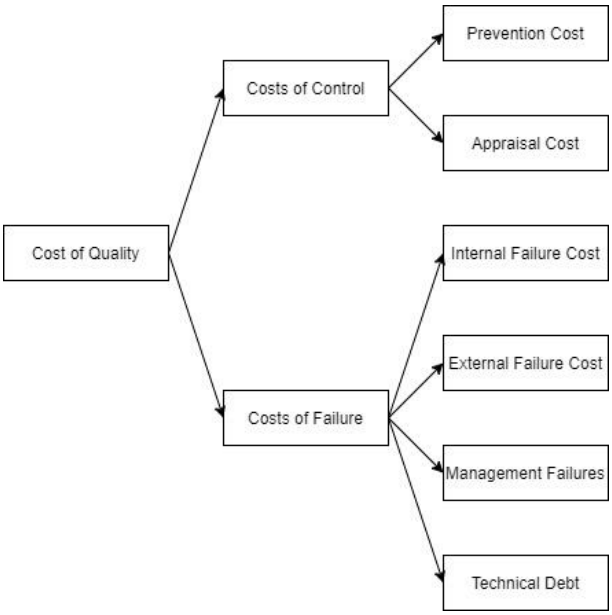

Fig. 1. Subbranches of the cost of software quality.

As previously mentioned, CoC is not in our scope. However, it is necessary that giving a brief introduction.

2.1 Cost of Contol

Costs of control relate to costs controlled by the software developer and includes the following subclasses:

- Prevention costs are quality infrastructure costs incurred throughout the organization.

- Appraisal costs are the error detection costs incurred before releasing the software.

Prevention Costs
The main reason for these types of costs is to train developers on writing secure and easily maintainable code. Also, the cost arises from efforts to prevent defects. One example of these can be quality assurance costs. Measurements, audits, evaluations, inspections are also prevention cos

Appraisal Costs
These types of costs are the costs incurred to correct the errors detected during the testing phase after the software process is over. Quality control costs are in this type. Also, requirement reviews, design reviews, and code reviews are other examples.

3 Cost of Failure

Many sources in the literature divide CoF into two sub-branches as internal and external (see Section 4). However, technical debt and management failures should also be mentioned. Hence, these two costs will be mentioned in this paper.

3.1 Internal Failure Cost

Internal failure costs (IFC) represent those incurred through correcting errors that were detected through design reviews, tests performed before the software was installed at the customer's system. The tests can be software tests and acceptance tests. This means, IFC the cost of error correction after formal examinations of the software during its development. During each test and review, the project leader or a member, who is responsible for it, should keep tracking due to the fact that each step is another cost. The most common reasons for IFC are [3]:

- Cost of rework that incurred after design review and test findings,
- Design correction that has the same reasons with the cost of rework,
- Program correction due to test findings,
- Unnecessarily repeated design reviews and retesting,
- Domino effect: Delayed software projects, besides their incurred failure costs, may cause damages to other projects performed by other teams, due to the transfer of team members to enforce other projects under pressure.

3.2 External Failure Cost

Sometimes, although the test is done, some defects can be overlooked. These faults do not always interfere with system operation. Sometimes it is at the bug level. In such cases, the product reaches the customer. External failure cost (EFC) is the cost of the

defects that have reached the customers such as this. The most common reasons for EFC are [3]:

- Resolution costs during the warranty period,
- Correction cost of bugs,
- Correction costs incurred after the warranty period, even if the fix is not covered by the warranty,
- Customer-induced costs,
- Repay costs because of delays,

3.3 Management Failures

Management failure costs (MFC) relate to costs of correcting failures that occurred due to unsuccessful prevention activities. These are:

- Unplanned costs for professional and other resources, resulting from an underestimation of the resources in the planning stage.
- Damages paid to customers as compensation for late project completion.
- Damages to other projects planned to be performed by the same teams involved in the delayed projects.
- Excessive management crisis mode behaviors, like lots of meetings to solve urgent problems.

3.4 Technical Debt

Technical debt describes what results when development teams take actions to expedite the delivery of a piece of functionality or a project which later needs to be refactored. This can occur due to:

- Structural problems of the design
- Increased complexity due to shortcuts
- Future refactoring
- Debt service and interest

4 Cost of Quality Models

Since Juran proposed the cost of quality, many researchers have improved or introduced various approaches to measuring CoQ. Schiffauerova and Thomson [5], presents a survey of published literature about various quality costing approaches and reports of their success to provide a better understanding of the cost of quality models in 2006. In addition, ten years before them, in 1998, Plunkett and Dale [27] categorizes and discusses the cost of quality models in the light of the research experience. This paper is concluded that many of the models are inaccurate and misleading and serious doubts are cast on the concept of an optimum quality level corresponding to a minimum point on the total quality-cost curve. Kumar et al. published a paper in 2010 [28]. This paper reviews the surveys of quality costs conducted in various countries.

Based on the aforementioned and other researchers, there are several models to explain and calculate CoQ. In agreement with the majority of previous researchers present work classifies CoQ models into five discrete generic groups as in Table 1.

Table 1. Generic CoQ models and cost categories.

Model	Founder	Category
P-A-F models [29]	Feigenbaum	Prevention + appraisal + failure
Crosby's model [2]	Crosby	Prevention + appraisal + failure + opportunity
Intangible cost models	This model is a common product that has emerged as a result of many studies.	- Conformance + non-conformance - Conformance + non-conformance + opportunity - Tangibles + intangibles - P-A-F (failure cost includes opportunity cost)
Process cost models [30]	British Standards Institution	Conformance + non-conformance
ABC models [31]	Cooper and Kaplan	Value-added + non-value-added

4.1 The old and the new models

The first model was firstly proposed by Juran [1] and had been applied extensively till the '90s. This model which is presented in Figure 2 suggests that the costs of poor quality decrease with higher quality levels, while the costs of achieving good quality increase.

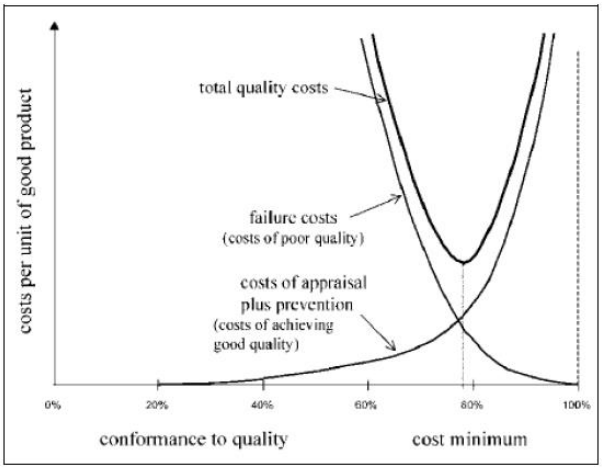

Fig. 2. The old COQ model (Juran's model).

However, things are different in the industry. After several studies with the industry, Juran's model was improved and Figure 3 has been obtained.

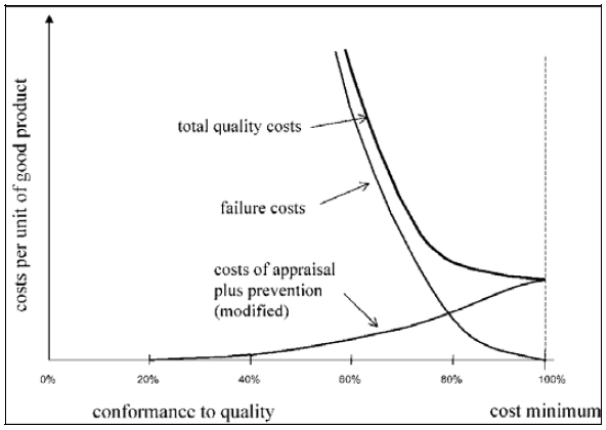

Fig. 3. The new COQ model (Juran's model).

It can be said that the new COQ model presents a much more rounded perspective on quality costs and seems to reflect business reality much closer than the old model. In 2004, Freiesleben [32], wrote a criticism paper about both the new and the old models.

5 Literature Review

As mentioned earlier, the quality cost was defined by Juran in 1951 as a technique for calculating the costs of quality processes required to provide the desired features and behaviors by the user, customer, and other system stakeholders [1]. Quality cost is used to control the costs of the work to be done to ensure quality in both the service and production industry, to define what can be done to reduce the quality costs, and to show the income profitability provided by these studies. You can see the most important studies as milestones on CoQ in Table 2.

Table 2. The most important studies on CoQ.

Authors	Work	Abstract
Daughtrey [10]	Fundamental Concepts for the Software Quality Engineer	This book is a collection of the best articles on software quality, taken from the professional journal The Software Quality Professional and recent International Conferences on Software Quality.
Flowers [11]	Software Failure: Management Failure	The purpose of this book is to bring together material about several significant information systems that did not perform as originally envisaged and to provide a coherent account of the surrounding events.

Galin, Avrahami [12]	Benefits of a Higher Quality Level of the Software Process: Two Organizations Compared.	The authors' study employed a methodology based on a comparison of observations in two organizations simultaneously. Six quality performance metrics were employed: 1) error density, 2) productivity, 3)percentage of rework, 4) time required for an error correction, 5) a percentage of recurrent repairs, and 6) error detection effectiveness.
Jones, Bonsignour [13]	The Economics of Software Quality	Using empirical data from hundreds of software organizations, the authors show how integrated inspection, structural quality measurement, static analysis, and testing can achieve defect removal rates exceeding 95 percent.
Karg, Grottke, Beckhaus [14]	Conformance Quality and Failure Costs in the Software Industry: An Empirical Analysis of Open Source Software	Recently, it has received a lot of attention in the field of software engineering. However, empirical studies of the association between failure costs and conformance quality have only been conducted for closed source software projects, but not for open source projects. This paper addresses this research gap.
Karg, Grottke, Beckhaus [15]	A systematic literature review of software quality cost research	This paper identifies the predominant researchers in the software quality cost domain and the related research clusters. Also, it classifies the articles according to three properties: research topic, research scope, and research approach.
Knox [16]	Modeling the Cost of Software Quality	This paper offers an extrapolation of the manufacturing and service industries' Cost of Quality Model to the business of software development.
Krasner [17]	Using The Cost of Quality Approach For Software	This article discusses the rationale and context for using CoSQ, then defines a basic CoSQ approach that differentiates the costs involved with: handling nonconformances due to a lack of quality, activities performed for the achievement of acceptable quality, and efforts to prevent poor quality.

Recently, software costs have been tried to be estimated by machine learning techniques. Important research has been done on software metrics for the error prediction process using a machine learning (ML) algorithm. After this point, these studies will

be focused more on. Most of the research used the NASA MDP (Metric Data Program) repository as the experimental dataset. Each dataset lists all faults discovered in the system and the number of modules containing the fault, while each module is described with a set of code level or design level attributes.

In 2009, Catal and Diri [18] used Random Forest, Naïve Bayes, j48, etc. to predict costs. They found that the Naïve Bayes algorithm is better for a smaller dataset, while the Random Forest algorithm performs best for the larger dataset. Before this work, in 2007, Catal et al. [19] used an immune-inspired supervised learning algorithm named Artificial Immune Recognition System (AIRS). Their data source was 6 object-oriented metrics of Chidamber-Kemerer metric and lines of code metric. According to them, the best fault prediction result is the combination between CK metrics and the lines of code metric.

Shanthaini found that precision, recall, and accuracy of SVM show better results compare to other machine learning methods for both class and method level metrics in 2012 [20]. In the same year, Bishnu and Bhattacherjee, chose AR3, AR4, AR5 from PROMISE as datasets and found that K-Means algorithm more efficient in the term of iterations except for AR5, and the SSE [22]. In the same year, Malhorta and Jain conducted a study using not only ML but also statistical methods [26].

In 2014, Okutan and Taner used PROMISE and discovered that the statistical comparison showed better fault classification after redundancy removal and log-transformed data than otherwise [23].

Using the PROMISE repository, Mundada et al proved that ANN shows better accuracy compared to previous ML researches in 2016 [21]. Again in 2016, Alighardashi et al used a feature selection method by using a combination of filter feature selection methods on NASA and PROMISE datasets [24]. Based on their findings, the results show the effectiveness of their method used. Therefore, the proposed method can find the best features with the highest speed for the improvement of the fault prediction accuracy. In the same year, Kumudha and Venkatesan proposed a classifier and test it on the NASA dataset. They concluded that their approach shows more effective results compare to early existing predictors [25].

References

1. Juran, J. M.: Juran's Quality Handbook. 5th edn. McGraw-Hill, New York (1998).
2. Crosby, P.: Quality Is Free: The Art of Making Quality Certain. McGraw-Hill, New York (1978).
3. Galin, D.: Software Quality: Concepts and Practice. 1st edn. John Wiley and Sons, Inc., New Jersey (2018).
4. Houston, D., Keats, J. B.: Cost of software quality: a means of promoting software process improvement. Quality Engineering 10(3), 563-573 (1998).
5. Schiffauerova, A., Thomson, V.: A review of research on cost of quality models and best practices. International Journal of Quality & Reliability Management 23(6), 647-669 (2006).
6. Slaughter, S. A., Harter D. E., Krishnan M. S.: evaluating the cost of software quality. Communications of the ACM 41(8), 67-73 (1998).
7. Garousi V., Coşkunçay A., Betin-Can A., Demirörs O.: A survey of software engineering practices in Turkey. Journal of Systems and Software 108, 148-177 (2015).

8. Cruzes, D. S., Dybå T.: Synthesizing evidence in software engineering research. In: 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ACM (2010).
9. Laporte, C. Y., Berrhouma, N., Doucet M., Palze-Vargas E.: Measuring the cost of software quality of a large software project at bombardier transportation: a case study. Software Qual. Manage 14(3), 14-31 (2012).
10. Daughtrey, T.: Fundamental Concepts for the Software Quality Engineer. Asq Pr (2001).
11. Flower, S.: Software Failure: Management Failure. John Wiley and Sons, Inc, Chichester, NY (1996).
12. Galin, D., Avrahami M.: Benefits of a Higher Quality Level of the Software Process: Two Organizations Compared. Software Quality Professional 9(4), 27–35 (2007).
13. Jones, C., Bonsignour O.: The Economics of Software Quality. 1st edn. Addison Wesley, Boston (2011).
14. Karg, L. M., Grottke M., Beckhaus A.: Conformance Quality and Failure Costs in the Software Industry: An Empirical Analysis of Open Source Software. In: International Conference on Industrial Engineering and Engineering Management, pp. 1386–1390 (2009).
15. Karg, L. M., Grottke M., Beckhaus A.: A systematic literature review of software quality cost research. Journal of Systems and Software 84(3), 415–427 (2011).
16. Knox, S. T.: Modeling the Cost of Software Quality. Digital Technical Journal, 5(4), 9–15 (1993).
17. Krasner H.: Using the cost of quality approach for software: CrossTalk – TheJournal of Defense Software Engineering 11(11), 6–11 (1998).
18. Catal, C., & Diri, B.: Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem. Information Sciences, 179(8), 1040-1058 (2009).
19. Catal, C., Diri, B., & Ozumut, B.: An artificial immune system approach for fault prediction in object-oriented software. In: Dependability of Computer Systems. DepCoS-RELCOMEX'07. 2nd International Conference, pp. 238-245, (2007).
20. A. Shanthini,: Applying Machine Learning for Fault Prediction Using Software Metrics, 2(6), 274–278 (2012).
21. Mundada D., Murade A., Vaidya, O., Swathi, J. N.: Software Fault Prediction Using Artificial Neural Network And Resilient Back Propagation. Int. J. Comput. Sci. Eng., 5(3), 173–179 (2016).
22. Bishnu, P. S., Bhattacherjee V.: Software Fault Prediction Using Quad Tree-Based K-Means Clustering Algorithm. 24(6), 1146–1150 (2012).
23. Okutan A., Taner O.: Software defect prediction using Bayesian networks. 2, 154–181 (2014).
24. Alighardashi, F., Ali, M., Chahooki, Z.:The Effectiveness of the Fused Weighted Filter Feature Selection Method to Improve Software Fault Prediction. 8, (2016).
25. Kumudha, P., Venkatesan, R.: Cost-Sensitive Radial Basis Function Neural Network Classifier for Software Defect Prediction. 2016 (2016).
26. Malhotra, R., Jain, A.: Fault prediction using statistical and machine learning methods for improving software quality. J. Inf. Process. Syst., 8(2), 241–262 (2012).
27. Plunkett, J. J., Dale, B. G.: Quality costs: a critique of some economic cost of quality models. International Journal of Production Research, 26 1713-1726 (1988).
28. Kumar, K., Shah, R., Fitzroy, P. T.: A review of quality cost surveys. Total Quality Management & Business Excellence, 9 479-486 (2010).
29. Feigenbaum, A.V.: Total quality control. Harvard Business Review, 34, 93-101 (1956).

30. BS 6143: Part 1, Guide to the Economics of Quality: The Process Cost Model, British Standards Institution, London, 1992.
31. Cooper, R., Kaplan, R.S.: Measure costs right: make the right decisions. Harvard Business Review, 66, 96-103 (1988).
32. Freiesleben, J.: On the Limited Value of Cost of Quality Models. Total Quality Management & Business Excellence, 15, 959-969 (2004).