International Journal of Computer Applications (0975 – 8887)
Volume 5– No.8, August 2010

An Approach to Optimize the Cost of Software Quality Assurance Analysis

Manju Lata
Dept. of Computer Science & Engg.
IIMT, Engineering, College, Meerut

Rajendra Kumar
Dept. of Computer Science & Engg.
Vidya College of Engineering, Meerut

ABSTRACT
In this paper we present an approach to optimize the cost of software quality assurance. It points out, how to optimize the investment into various software quality assurance techniques and software quality. The expected and reliable development of high quality software regularly becomes a major problem due to late removal of defect. The detection and removal of defect is a software inspection providing technical support for the defect detection activity, and large volume of documentation are related to software inspection in the development of the software quality assurance as a cost effective. The value of an inspection improves the quality and saves defect cost. We describe the optimization model for selecting the best commercial off-the-self (COTS) software product among alternatives for each module. As objective function of the models is to maximize quality within a budgetary constraint and standard quality assurance (QA) methodologies cuts maintenance costs. Increase reliability, and reduces cycle time for new distribution modeling system. An analytical and stochastic model of the economics of analytical software quality assurance (SQA) is based on expected values. The model is able to handle different type of techniques such as static and dynamic. The model can be used to analysis different type of techniques theoretically, and to optimize the software quality assurance.

Keywords: Defect detection; modeling system; software acquisitation; analytical SQA; quality Assurance.

1. Introduction

As an approach to optimize the cost, software quality can be boiled down to cost and benefit in the economical sense because usually software use for some business reason. Business value for the vendor as well as for the customer depends on the quality. Software quality assurance (SQA) is an important factor in the development of the software quality, and followed throughout the software acquisition life cycle. Software development and control processes should include quality assurance. Inspection and testing are used for defect detection and removal. Software design inspection saves on average 44% of the testing cost and code inspection save on average 39% of the cost [7]. Quality cost analysis shows the companies send between 50% to 80% of their development effort on testing [6], and the cost of analytical SQA is significant. Many estimates say that analytical SQA constitutes about 50% of the total development costs.

Cost and benefits of various software quality assurance techniques allows for economically decision-making. The software quality measures how well software is designed. As the cost of SQA, we need to optimize the development process with the aim to reduce costs and increase benefits. There are two approaches: (1) Develop existing techniques, and (2) the existing techniques use in a cost optimal way. In the development process, this approach identify defect–prone component based on detailed UML models, and contains several case-studies that ratify the proposed approach.

2. Related Work
Steve McConnell's code [16] divides software into two parts: internal and external quality characteristic. External are those parts of the product that face its users, while internal are those that do not. Tom De Marco [17] says "a product's quality is a function of how much it changes the world for the better" that means user satisfaction is more important than anything in determining software quality [15] as in [18]. Software quality assurance is defined by the theoretical model of the effectiveness and efficiency of either test techniques or inspections, and by the economics-oriented, abstract models for quality assurance, approaches to identify defect-prone components, and small number of components of a system contains most of the defects [12]. Detailed design models offer the possibility to analysis the system early in the development life-cycle. Humphrey [11] presents an understanding of software quality economics. The defined cost metrics do not represent monetary values but only fractions of the total development time.

2.1 Defect Introduction and removal

An analytical quality assurance typically accounts for about 50% cost during development. In this approach describes analytical model of the effectiveness and efficiency of defect introduction and removal. The most detailed and comprehensive model of defect introduction and removal was developed by Chulani and Boehm [2, 3]. This is a part of COQUALMO. COQUALMO is an extension of COCOMO. According to Boehm [1], different phase and defect classes are introduced such as requirements, design, code, and documentation are introduced in the defect introduction and removal model. Requirement can also be applied specifically to the analysis proper, as opposed to elicitation or documentation of requirements.

Software designing phase usually involves the use of more abstract and general means of specifying the parts of software, and break the large code into small code as given in figure 1.

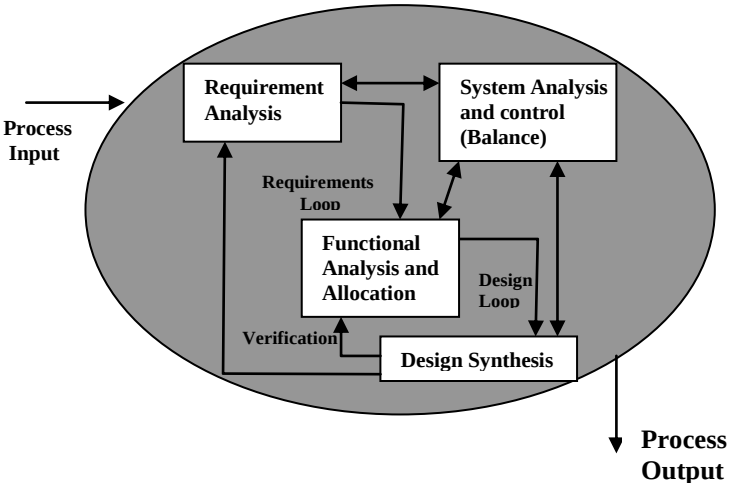

Fig. 1. Illustration of design synthesis

2.2 Cost-Benefit of SQA

Mandeville [9] describes software quality costs as an adoption of the PAF model. Quality costs are the costs associated with preventing, finding, and correcting defective work. PAF (Prevention, Appraisal, Failure) model define the first step of the quality cost and software quality cost. Many software quality cost model are also based on the PAF model but do not refine the cost factors [13, 14]. Furthermore, it includes technical factors of the quality assurance process, and general methodology for cost collection. Figure 2 represents the categorization of cost of quality.

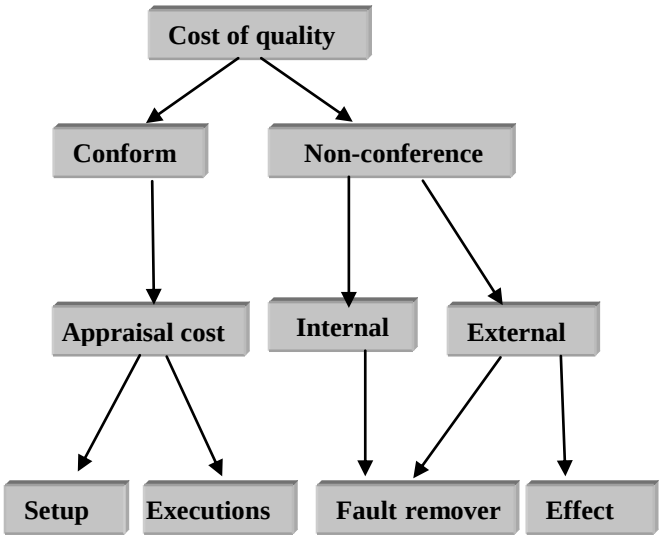

Fig. 2. Categorization of cost of quality

Prevention cost is the cost of activities that are specifically designed to poor quality. Examples of poor quality includes coding errors, design errors. Prevention cost eliminate from the software quality cost. AF (Appraisal, Failure) model essentially reduced the PAF model. Appraisal cost defines by the setup and execution costs. Fault removal cost means found the fault and remove it. It can be attributing to the internal failure costs and external failure costs. External failure also cause effect-cost associated with the failure apart from the removal costs.

Total cost of quality

The sum of the costs is defined as

Prevention + Appraisal + Internal failure + External failure.

As define in [4] Software inspection ensures the software quality by finding the defect in development process. Cost and benefit is a factor in planning software quality assurance and formal techniques include cost-effectiveness analysis, impact analysis, fiscal impact analysis and social return on investment (SROI) analysis. In the software inspection process, first review software artifacts individually and then team finding many defect using two techniques: Check-list Based Reading (CBR) and Perspective-Based Reading (PBR). In [5] cost-benefit model use for inspection and re-inspection, and justify the cost and benefit assumption.

3. Cost Estimation Model

Software cost estimation model developed by Barry Boehm. As define in [1], the Constructive cost model (COCOMO) was first published in 1981, used for estimating effort, cost, and schedule for software project. It is based on waterfall model. This model is typically calling COCOMO 81. In 1997 COCOMO II was developed and finally published in 2000. COCOMO consists of three increasingly detailed and accurate forms.

***Basic COCOMO*:** It is a first level, basic COCOMO is use for quick estimate of software cost, and program size is expressed in estimated thousand of lines of code (KLOC). COCOMO uses three classes of software project-

1. Organic projects: small teams with good experience working with less than rigid requirements.
2. Semi-detached projects: medium teams with mixed experience working with a mix of rigid and less than rigid requirements.
3. Embedded project: developed within a set of tight constraints (hardware, software, operational).

Basic COCOMO equation takes the form.
Effort Applied = a_b (KLOC) b_b [man-month]
Development Time = c_b (Effort Applied) d_b [months]
People required = Effort Applied / Development Time [count]

The coefficients a_b, b_b, c_b and d_b are given in this table 1.

Table 1. Coefficient Table

Software project	a_b	b_b	c_b	d_b
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Basic COCOMO is good for quick estimate of software costs.

***Intermediate COCOMO*:** It is a second level, intermediate COCOMO takes these Cost Drivers into account. Cost Derivers include subjective assessment of product, hardware, personnel, project attributes. Each of the 15 attributes of four Cost Derivers receives a rating on a six-point scale that ranges from very low to extra high (in importance or value). An effort multiplier from the table blow applies to the rating. The product of all effort multipliers results in an effort adjustment factor (EAF). Typical values for EAF range from 0.9 to 1.4 in table 2.

Table 2. EAF table

	Ratings					
Cost Derivers	Very Low	Low	Nominal	High	Very High	Extra High
Product attributes						
Required software reliability	0.75	0.88	1.00	1.15	1.40	--
Size of application database	--	0.94	1.00	1.08	1.16	--
Complexity of the product	0.70	0.85	1.00	1.15	1.30	1.65
Hardware attributes						
Run-time performance constraints	--	--	1.00	1.11	1.30	1.66
Memory constraints	--	--	1.00	1.06	1.21	1.56
Volatility of the virtual machine environment	--	0.87	1.00	1.15	1.30	--
Required turn about time	--	0.87	1.00	1.07	1.15	--

Personnel attributes						
Analyst capability	1.46	1.19	1.00	0.86	0.71	--
Application experiences	1.29	1.13	1.00	0.91	0.82	--
Software engineer capability	1.42	1.17	1.00	0.86	0.70	--
Virtual machine experience	1.21	1.10	1.00	0.90	--	--
Prog. language experience	1.14	1.07	1.00	0.95	--	--
Project attributes						
Application of s/w Engg. methods	1.24	1.10	1.00	0.91	0.82	--
Use of s/w tools	1.24	1.10	1.00	0.91	0.83	--
Required development schedule	1.23	1.08	1.00	1.04	1.10	--

The Intermediate COCOMO formula now takes the form:

$E = a_i (\text{KLOC})^{b}i .\text{EAF}$

Where E is the effort applied in person-month, KLOC is the estimated number of thousands of delivered lines of code for the project, and EAF is the factor calculated above. The coefficient a_i and the exponent b_i are given in the table 3.

Table 3. The coefficient and exponent table

Software Project	a_i	b_i
Organic	3.2	1.05
Semi-detached	3.0	1.12
Embedded	2.8	1.20

Third level is Detailed COCOMO follow the each step of analysis, design, etc. with all characteristics of the intermediate cost driver's impact. Boehm et al. present in [10] the iDAVE model that is based on COCOMO II and COQUALMO. This model allows a through analysis of the return on investment (ROI) of dependability.

4. Our Analytical Model for Cost Defect-Detection

The model is divided into three components, all components depends on the spent effort (say t) as a global parameter.

1. Direct costs $d(t)$
2. Future costs $f(t)$
3. Revenues or saved costs $r(t)$

***Direct costs*:** The direct costs are those costs that can be directly measured from the application of a defect-detection technique. They are dependent on the length t of the direct costs for an application of technique A.

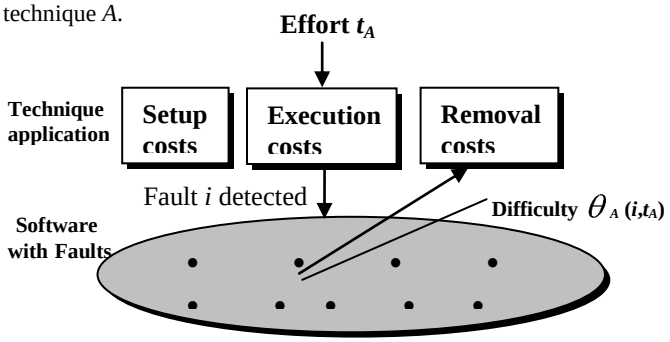

Fig. 3. The components of the direct costs

It contains two main cost blocks- setup costs and execution costs. It is dependent on the spent effort for A denoted by t_A. From the execution costs we can derive the difficulty of detecting the faults in the software which represents the probability that the fault is not detected. However, if a fault is detected it incurs costs for its removal. The expected value of the direct costs $E[d_A(t_A)]$:

$$E[d_A(t_A)] = u_A + e_A(t_A) + \sum_i (1 - \theta_A(i, t_A)) v_A(i)$$

Where u_A are the setup costs, $e_A(t_A)$ the execution costs, and $v_A(i)$ the fault removal costs specific to that technique.

***Future costs*:** The future costs denoted by $E[f_A(t_A)]$. It is divided into two parts - fault removal costs $v_F(i)$ and failure effect costs $c_F(i)$.

$$E[f_A(t_A)] = \sum_i \pi_i \theta_A(i, t_A)(v_F(i) + c_F(i))$$

where π_i = P (fault i is activated by randomly selected input and is detected and fixed) [8]. Hence, it describes the probability.

***Revenues*:** It is considering not only the costs of the defect detection technique but also their revenues. They are essentially saved future costs. We denote the revenues with $E[r_A(t_A)]$

$$E[r_A(t_A)] = \sum_i \pi_i (1 - \theta_A(i, t_A))(v_F(i) + c_F(i))$$

5. Working of the Model

Based on the three components of the model, we are able to calculate several different economical metric of the quality assurance process. There are metrics total cost, profit, and return on investment. All these metrics can be used for two purposes: (1) an up-front evolution of the quality assurance plan as the expected total cost, profit, or return on investment, and (2) a single post-evolution of the quality assurance of the project.

***Total Cost*:** The total cost describes the sum of all economic costs for producing products. It is one possible metric that can be optimized. Total cost can be calculated by adding the direct costs and the future costs. The expected value of the direct costs dx and future costs fx of the sequence of defect-detection technique applications X.

Total cost = $dx + fx$

***Profit*:** We describe the gain provided by the quality assurance with the term profit. Hence, it is the revenues less the total cost. It is defined using the three components as: direct costs, future costs, revenues. The expected value of the revenues rx of the sequence of defect-detection technique applications X.

Profit = $rx - dx - fx$

***ROI*:** Another metric used in economic analyses is the return on investment (ROI) of the defect-detection techniques. The ROI- also called rate of return - is commonly defined as the gain divided by the used capital. We use Boehm et al. [10] equation for (Benefits - Costs) / Costs, to calculate the total return on investment (ROI).

$ROI = rx - dx - fx / dx + fx$

5. CONCLUSION

In conclusion an overview on the debate concerning quality and cost ascertaining in general we will be given. There are the numbers of techniques to verify the cost effectiveness of quality assurance. Cost optimal use analytical quality assurance but we do not distribute the effort between different techniques but we analysis how the effort is best distributed over the components of the system. This is done by

identifying the most fault- and failure-prone components based on a metrics suite and detailed design models. The approaches that exist for models either have slightly different aims, analysis dependability attributes or readability, or concentrate on the static structure, analysis the fault-proneness. During the development and quality assurance, we use estimating quality models that assess the current state of the product and process. All the discussion so far viewed the system of which the quality is assure as the whole, however, there are the possibility to optimize cost of quality assurance on the architectural level. In particular, defect-detection techniques can be concentrated on components that are most defect-prone.

REFERENCES

[1] Barry Boehm. "Software engineering economics", Englewood, Cliffs, NJ: Prentice- Hall, 1998.

[2] Sunita Chulani and Barry Boehm, "Modeling Software Defect Introduction and Removal: COQUALMO (COnstructive QUALity MOdel)", Technical Report USC-CSE-99-510, University of Southern California, enterforSoftwareEngineering,1999.

[3] Sunita Devnani Chulani, "Bayesian Analysis of Software Cost and Quality Models", PhD Dissertation, University of Southern California, 1999.

[4] Maria Victoria Cengarle, "Inspection and Testing – Towards combining both approaches", Technical Report 024.02/E, Fraunhofer IESE, 2002.

[5] Stefan Bill, Bernd Freimut, Oliver Lait emberger. "Investigating the cost- effectiveness of reinspections in software development", Proceedings of the 23rd International Conference of software Engineering, P. 155-164, May 12-19, 2001.

[6] J.S. Collofello, S.N. Woodfield, "Evaluating the Effectiveness of Reliability-Assurance techniques", Journal of systems and software 9 (3) (1989) 191-195.

[7] L. Briand, K. EI Emam, O. Laitenberger, "Using Simulation to Build Inspection Efficiency Benehmarks for development projects", T. Fussbroich, Proceedings of The 20th International Conference on Software Engineering, Kyoto, Japan, (1998) 340-349.

[8] Bev Littlewood, Peter T. Popov, Lorenzo Strigini, and Nick Shryane, "Modeling the Effects of Combining Diverse Software Fault Detection Techniques", IEEE Transactions on Software Engineering, 26(12): 1157-1967, 2000.

[9] William A. Mandeville, "Software costs of quality", IEEE Journal on Selected Areas in Communications, 8(2): 315-318, 1990.

[10] Barry Boehm, LiGuo Huang, Apurva Jain, and Ray Madachy, "The ROI of software Dependability: The iDAVE model", IEEE software, 219(3): 54-61, 2004.

[11] Watts S. Humphrey, "A Discipline for Software Engineering", The SEI Series in Software Engineering, Addison-Wesley, 1995.

[12] Barry Boehm and Victor R. Basili, "Software Defect Reduction Top 10 List. IEEE Computer", 34(1): 135-137, 2001.

[13] Daniel Galin, "Towards an inclusive model for the cost of software quality", Software quality Professional, 6(4): 25-31, 2004.

[14] Stephen T. Knox, "Modeling the costs of software quality", Digital Technical Journal, 5(4): 9-16, 1993.

[15] Crosby, P., "Quality is Free", McGraw-Hill, 1979.

[16] McConnell, "Code Complete: A Practical Handbook of Software Construction", Microsoft Press, 1993

[17] DeMarco, T., "Management can make quality impossible", Cutter IT Summit, Boston, April 1999.

18] Pressman, "Software Engineering", McGraw Hill, 2005.