

## Factors Affecting Effective Software Quality Management Revisited

Nasib S. Gill

Department of Computer Science & Applications,  
Maharshi Dayanand University, Rohtak – 124001

Haryana (India)

Email: [nsgill\\_2k4@yahoo.com](mailto:nsgill_2k4@yahoo.com)

### Abstract

Developing a good software system is a very complex task. In order to produce a good software product, several measures for software quality attributes need to be taken into account. System complexity measurement plays a vital role in controlling and managing software quality because it generally affects the software quality attributes like software reliability, software testability and software maintainability. Thus, software quality assurance (SQA) needs to be addressed keeping in view the new strategies, tool, methodologies and techniques applicable to software development life cycle.

This paper is primarily aimed at revisiting and examining peculiar aspects of software development process that affect software quality management process. These aspects of software development process include software reliability measurement, ISO approach applicable to software quality and some aspects related to software testing improvement. Software testing and evaluation methods/tools/techniques do not guarantee effective testing and ensure high software quality. The way to improve the effectiveness of testing is to improve the attitude of software developers towards testing.

In this paper, all these factors affecting software quality management have been discussed as well as all the possible improvements have been suggested. The results of this paper may be quite helpful to the researchers in quantifying the specific measuring tools for these software quality attributes.

**Keywords:** Software Quality, Software Quality Management, Software Reliability, ISO, Software Quality Assurance.

### 1. Introduction

Software quality is gaining much more attention these days as well as much more emphasis is being placed on production of high quality software products.

Software development is a complex process requiring careful integration of diverse disciplines, technical activities, project management etc. Most software are produced by the co-operative effort of many designers and programmers working over a period of man years. The resulting product can't be fully understood by any person. No matter how elegant the methods used to test the final product, how complete the documentation, how structured the methodology, the development plans, the project reviews, the walkthroughs, the database management, the configuration control, no matter how advanced the tools and techniques - all will

come to nothing and the project will fail if the quality management system is not effective.

### 2. Software Quality Management

Quality standards can only be achieved by employing effective quality management system. Quality is built into software products through the management and technical procedures that are defined and implemented to assure:

- Quality,
- Schedule and
- Budget Compliance

A number of technologies have been for Software Improvement, which has been the most important goal of Software Engineering. Few examples of the important technologies are:

- Requirement definition
- Defect prevention
- Defect detection
- Defect removal.

### 3. Software Reliability Measurement

Software reliability can be measured, directed, and estimated using historical and development data. In statistical terms, 'software reliability' is defined as "the probability of failure free operation of a computer program in a specified environment for a specified time".

Many researchers attempted to extrapolate the mathematics of hardware reliability theory to the prediction of software reliability. Most hardware related reliability models are predicated on failure due to wear rather than failure due to design defects.

In hardware, failures due to physical wear such as effects of shock, temperature, corrosion, etc. are more likely than a design related failure. Unfortunately, the opposite is true for software. In fact, all software failures can be traced to design or implementation problems; wear does not enter into the picture.

If we consider a computer-based system, a simple measure of reliability is MTBF (mean Time Between Failure) and the same is given as:

$$MTBF = MTTF + MTTR$$

where

MTTF : Mean Time To Failure  
MTTR : Meant Time To Repair

In addition to a reliability measure, another measure is defined as availability measure, where availability measure is defined as defined as the probability that a program is operating according to requirements at a given point in time and is defined as:

$$\text{Availability} = \text{MTTF} / (\text{MTTF} + \text{MTTR}) \times 100\%$$

#### 4. Software Quality Assurance

Software quality assurance is the mapping of the managerial precepts and design disciplines of quality assurance onto the applicable managerial and technological space of software engineering.

SQA is an *“umbrella activity”* that is applied at each step in the software process. SQA is complicated by the complex nature of the software quality that is defined as an attribute of computer programs that is defined as “conformance to explicitly and implicitly defined requirements”. SQA encompasses the following:

- Quality management approach
- Effective software engineering technology
- Formal technical reviews that are applied throughout the software process
- Multi-tiered testing strategy
- Control of software documentation
- Procedure to assure compliance with software development standards
- Measurement and reporting mechanisms.

Software safety and hazard analysis are software quality assurance activities that focus on the identification and assessment of potential hazards that may impact software negatively and cause an entire system to fail.

If hazards can be identified early in the software engineering process, software design features can be specified that will either eliminate or control potential hazards. A modelling and analysis process is conducted as part of software safety. Initially, hazards are identified and categorized by criticality and risk.

#### 5. ISO Approach To Software Quality Management

ISO 9000 describes quality assurance elements that can be applied to any business regardless of the products or services offered. The ISO 9000 quality assurance models treat an enterprise as a network of interconnected processes. For a quality system to be an ISO-compliant, these processes must address the areas identified in the standard. ISO 9000 describes the elements of a quality assurance system in general terms.

ISO 9000/IS 14000 is intended to provide common mean for establishing an effective quality management system when related to software, which together with procedures for the specifications, design, implementations and evaluation, allow development of software in a controlled manner. This should result in creation of software in the most cost effective way, having due regard to whole life cycle of the product, and should instil a high degree of confidence that the software will meet the operational requirements.

ISO 9000/IS 14000 defines the essential features of the system and does not attempt to prescribe how the system will be implemented. It is for developers to establish procedures appropriate to their own scale, methodology and organisation to achieve the requirements of the standard. The standard establishes requirements for software quality management system to be designed, developed and maintained with the objective of ensuring that the software will meet the requirements of a contract, purchase order or other agreement.

In every organisation, anybody having anything to do with developing of software contributes to the quality. All responsibilities and authorities, which includes for quality, should therefore be clearly established and understood. Working relationships between sub-groups of the organisation require clear understanding and co-ordination between managers and employees.

An effective system to quality management, planned and developed in conjunction with other functions, should be documented. Requirements should be met by the establishment and implemented of procedures with the specific purpose of ensuring that only software conforming to contractual requirements is delivered.

In pursuance of these requirements, the system should:

- Demonstrate both recognition of the factors which may affect quality
- Ensure that quality requirements are determined and that standards and procedures are established to satisfy such requirements, including development, acquisition, inspection and testing, packaging, shipping, storage, installation and maintenance
- Provide for the early and prompt detection of actual or potential deficiencies, trend or conditions which could result in non-compliance with requirements and for timely and effective corrective action; and
- Make available evidence that the quality system is effective.

All standards and procedures to be used in development of software should be well documented and readily accessible. They should detail standards of code and documentation, including planned content and format, and procedures to be executed in the testing and implementation of the software, and should cover following additional subjects:

- **Project Management:** It includes milestones, resource allocation, sub-contractor control, significant timing and reporting structure.
- **Design Techniques:** It covers methodology selection and reference to other manuals where necessary.
- **Review Procedures:** It defines the method of review, responsibilities and corrective action, where appropriate.

## 5.1 ISO 9001

ISO 9001 is the quality assurance standard that applies to software engineering. ISO 9001/IS 14001 is the quality system model appropriate for the software industry. The standard contains the following 20 requirements:

- *Statistical techniques*
- *Management responsibility*
- *Quality system*
- *Contract review*
- *Design control*
- *Document and data control*
- *Purchasing*
- *Control of customer supplied product*
- *Product identification and trace ability*
- *Process control*
- *Inspection and testing*
- *Control of inspection, measuring, and test equipment*
- *Inspection and test status*
- *Control of non-conforming product*
- *Corrective and preventive action*
- *Handling, storage, packaging, preservation, and delivery*
- *Control of quality audits*
- *Training*
- *Servicing.*

However, ISO 9001/IS 14001 being generic in nature requires to be understood in the context of software industry. Guidance in this respect has been provided by another standard, namely, ISO 9000-3/IS 14000 (Part 3), by correlating the 20 elements of ISO 9001/IS 14001 to the activities in the software industry.

To properly conduct software quality assurance, data about the software engineering process should be collected, evaluated and disseminated. Statistical SQA helps to improve the quality of the product and the software process itself. Software reliability models extend measurements, enabling collected defect data to be extrapolated into projected failure rates and reliability predictions.

## 6. Foundation of Software Testing

As the complexity of applications and the software increases, software testing and evaluation becomes more difficult and its effectiveness falls below expectations. Software testing is not an exact science but is both an art and a science. But testing has often been pursued purely on technical grounds and during the past two decades there have been considerable advancement in software testing techniques and methodologies.

The foundations of software testing are:

- *Test process, test cases and test plan*
- *Techniques, methodologies, tools and standards*
- *The people and the organisation.*

The test process, techniques and tools are significant contributors to effective and efficient testing and quality assurance. They can

offer better results only when they are built upon the *“people foundation: and sound managerial and organisational culture”*. It is the people and the culture of the organisation that determines how any system is practised.

Testing has to concentrate on critical, significant, high value software elements from the customers’ perspective. This requires a greater awareness of the application environment and deeper understanding of the customer’s requirements. The following are three different types of requirements:

- Implied requirements
- Expected requirements
- Exciting requirements

The above requirements have a different impact on customer satisfaction. As the objective of software development to satisfy the customer’s requirements, software should be tested not only for implied requirements, but also for expected and exciting requirements. Further, software be tested for its robustness and test cases should include invalid inputs to check the operation of the software under invalid or erroneous conditions.

## 7. Key-problems of Current Software Testing Practices

The existing software practices are badly suffering from many ills caused due to:

- Testing practices
- Attitudes of users
- Culture of organisation.

Ills of existing software testing practices, attitudes and organisational culture include:

- Shortcuts in testing
- Reduction in testing time
- ‘Let go’ - deliver now, correct errors later - attitude
- Poor planning and co-ordination
- Lack of user involvement
- Poor documentation
- Lack of management support
- Inadequate knowledge of application environment
- Improper staffing
- Poor testability

## 8. Improving Software Quality Through Proper Testing

Quality is everybody’s job, but management’s responsibility. The quality culture of the organisation is the first aspect that need to be appraised and improved, if required. By culture we mean the way in which quality is viewed, talked about and implemented in the organisation.

Potential areas of improvement include:

- Management and organisational commitment and culture
- Participative testing
- Focus of testing
- Better planning and effective co-ordination

- Feedback and quest for continuous improvement
- Design of testability.

## 9. Conclusions

Loosely tested software system lowers down the system reliability that thereafter negatively affects 'Software Quality'. In the present paper 'Software Reliability Measurement' has been discussed besides ISO approach applicable to software quality assurance (SQA).

In order to enhance the effectiveness of testing and to improve the software quality, software houses must make transitions to higher software culture. Software testing techniques, methodologies, tools and standards can only aid in testing, but it is the management and the people involved who have to plan for and carry out effective testing. Testing need to focus on maximising 'customer satisfaction', rather than just detecting and correcting errors involved in delivered software. In this paper, all these factors affecting software quality management have been discussed as well as all the possible improvements have been suggested. The results of this paper may be quite helpful to the researchers in quantifying the specific measuring tools for these software quality attributes.

## References

1. Aurum, A., Jeffrey, R., Wholin, C., and Handzic, M. (2003) *Managing Software Engineering Knowledge*, Springer, 2003
2. Rosenberg, Linda and Gallo, Albert (2002). Software Quality Assurance at NASA. IEEE Aerospace Conference, 2002.
3. Gill N.S. (2002). Software Engineering: Software Reliability, Testing and Quality Assurance. Khanna Books Pub. Co.(P) Ltd., New Delhi. 2002.
4. Ghezzi Carlo, Jazayeri Mehdi, Mandrioli Dino (1996). Fundamentals of Software Engineering. *Prentice-Hall of India Pvt. Ltd.*, 1996.
5. Gillies, Alan C. (1997) Software Quality, Theory and Management. *International Thomson Computer Press*, 1997
6. Kitchenham, Barbara, Pfleeger, Shari Lawrence (1996). Software Quality: The Elusive Target. *IEEE Software*, Vol 13, No 1 (January 1996) 12-21
7. Gill N.S., Grover P.S., Taneja D.R. (1994). System Complexity As An Aid to Quality Assurance. *IEEE's Proceedings of 1st International Conference on Software Testing, Reliability and Quality Assurance*, Dec. 21-22, 1994, pp. 116-120.
8. Pressman Roger S. (1997). Software Engineering: A Practitioner's Approach. *McGraw-Hill International Editions*, 1997.
9. Rosenberg, Linda, and Hyatt, Lawrence (1996). Developing a Successful Metrics Program. *8th Annual Software Technology Conference*, 4/96
10. Schulmeyer, G. Gordon and McManus, James I., *Handbook of Software Quality Assurance*, 3rd Edition, Prentice Hall PRT, 1998
11. Wilson, W., Rosenberg, L., Hyatt, L. (1996). Automated Quality Analysis of Natural Language Requirement Specifications. *Proceedings of the Fourteenth Annual Pacific Northwest Software Quality Conference*, 1996
12. Aggarwal K.K. (1993). Reliability Engineering. *Kluwer Academic Publishers*, 1993.
13. Conte S.D., Dunsmore H.E., Shen V.Y. (1986). Software Engineering Metrics And Models. *The Benjamin/Cummings Publishing Company, Inc.*, 1986.
14. Murugesan S. (1994). Attitude Towards Testing: A Key Contributor to Software Quality. *IEEE's Proceedings of 1st International Conference on Software Testing, Reliability and Quality Assurance*, Dec. 21-22, 1994, pp. 111-115.
15. Shooman M. L. (1991). Software Engineering. *McGraw-Hill International Editions*, 1991.
16. Rawat A. S. (1994). ISO 9000: Software Perspective. *CSI Communication*, May 1994, pp. 4-9.
17. Tiwari Ashish, Tandon Amit (1994). Shaping Software Quality - The Quantitative Way. *IEEE's Proceedings of 1st International Conference on Software Testing, Reliability and Quality Assurance*", Dec. 21-22, 1994, pp. 84-94.
18. P.S. Grover, N.S. Gill, and R Singh (1994). Measuring Software Systems. In *Theory and Practice on Operations Research*, 1994.
19. Mall Rajib (1999). Fundamentals of Software Engineering. *Prentice-Hall of India Pvt. Ltd.*, 1999.
20. Ghosh Hiranmay (1994). A Comparison of ISO 9000 and SEI/CMM for Software Engineering Organisation. *IEEE's Proceedings of 1st International Conference on Software Testing, Reliability and Quality Assurance*", Dec. 21-22, 1994, pp. 78-83.