Architecting Software Quality: A Foundation For Software Productivity

Kamarudin Sa'adan
Senior Fellow (Computer Science), Faculty of Science and Technology,
Universiti Sains Islam Malaysia,
Bandar Baru Nilai, 71800 Nilai,
Negeri Sembilan, Malaysia
Email: kamarudin@usim.edu.my

Abstract **- The unique nature of Software production compared to other industrial products creates the need for different quality approach and tools for software development and maintenance. Although a wide range of Software Quality Assurance (SQA) models are available, six SQA system components which form the foundation for software quality system will be discussed in this paper. These components if implemented properly will minimize the multitude of software errors in pursuing an acceptable level of software quality. The first component known as pre-project component comprises of contract review and the development of project and quality plans. These activities are carried out to ensure project commitments have been adequately defined. The second component is devoted to the assessment and enhancement of project life cycle activities. Error prevention and software improvement which are applied throughout the entire organisation are components to eliminate or reduce the rate of errors. Next component is software quality management which is geared towards several goals. They include the control of development and maintenance tasks. Standardisation, certification and SQA system assessment are the fifth components discussed. Finally, component relating to organising people involved in the software development process will be discussed since all of them contribute significantly to software quality. The model proposed is part of popular software quality models such as the ISO 9001 standards and the Software Engineering Institute's Capability Maturity Model (CMM) and Capability Maturity Model Integration (CMMI). In the proposed model the characteristics of the organisation itself which greatly influences the design of an efficient and effective software quality assurance system will be highlighted.**

Keywords : Software quality; Software Quality Assurance System

I. INTRODUCTION

Software quality assurance is now becoming an important component of every software development activity since complex and large software is increasingly being developed in important and critical applications. Over the years many standards and documents at national or International levels have evolved. The ISO 9001:2000 is one of the International standards defining and describing the requirements of a satisfactory quality management system. Other well-known models include the Capability Maturity Model (CMM) and Capability Maturity Model Integration (CMMI) [1]. Sometimes Total Quality Management (TQM) models such as Six Sigma and Plan, Do, Check, and Act (PCDA) [2, p. 67] were also applied to software development process. Six Sigma model strives to lower the defect rate to 3.4 per million manufactured elements. In PCDA model "Plan" is planning an activity. "Do" is to perform the designated activity. "Check" is to evaluate if the activity was a success or failure while "Act" is to continue to fix the process. Although each of these models has different requirements, most of them subscribe to the basic premise that software quality assurance is a continuous process improvement and the production of quality products are the output of high quality processes. Great emphasis is placed on early errors detection and correction. While quality control (QC) is an operational issues, quality assurance is a strategic issue requiring due attention from top management. It is a proactive approach through proper planning and establishment of control activities embedded in the system life cycle so that the functionality and robustness of software developed can be observed, measured and controlled.

Software quality is defined as conformance to explicitly state functional and performance requirements, explicitly documented development standards, and characteristics that are expected of all professionally developed software [3, p. 199]. Consequently software quality assurance is the systematic planned set of actions necessary to provide adequate confidence that a software development or maintenance process conforms to established functional technical requirements as well as the managerial requirements of keeping to schedules and operating within budget [4, p. 27]. An organisational freedom is important for quality assurance to ensure integrity and to avoid biasness. Quality assurance may make use the results of other supporting processes such as verification, validation, joint reviews, audits and problem resolution. In additions an Software Quality Assurance (SQA) system also employs activities for the improvement and greater efficiency of software development, maintenance and the SQA activities themselves.

Unique characteristics of software products compared to other industrial products determined the need

for different quality approach to software systems. The high number of operational modes permissible in any software system makes software a highly complex product. Defects detection in software development and production is a very challenging task. For most other industrial products defects can be detected by checking and testing the product prototype. Additional opportunities are also available during production planning to inspect the products where the production process and tools are designed and prepared. This process does not exist for software production process since software copies are made automatically. Finally, the manufacturing of software is limited to copying the products into compact disc (CDs) or digital versatile/video disc (DVD) resulting in limited opportunity to apply SQA procedures to detect product failures.

II. SOFTWARE QUALITY ASSURANCE AS AN IMPORTANT BUSINESS DETERMINANT

Software quality is part of a greater field of TQM championed by W. Edwards Deming in manufacturing over 50 years ago [5]. The four basic principles of TQM in his model state that everyone has a customer; there is quantitative feedback from all customers, corrective action occurs based on the measures and every process has an owner. Schulmeyer and McManus [6] proposed the application of these four basic principles in software production.

Godbale [7, pp. 3-7] outlines six factors why software quality is crucial to the success of today's organisations. All the six factors can only be achieved if the software production and maintenance process is carried out within the framework of software quality assurance. These factors are:

- Software quality as a competitive factor
  Software system is now an important competitive issue in organisation. Functionality alone is not sufficient to be competitive. Quality of the software and it support is now the differentiating factors to gain competitive edge. This applies for both systems developed for external as well as internal clients.
- Software Quality for survival
  Delivering high quality products demanded by customers in a highly competitive and rapidly changing software market will determine the long term survival of the organisation. Quality certification is one of the important criteria sought after by most customers to determine quality.
- Software quality as a prerequisite for global reach
  Software is now a global business. Organisations irrespective of their size can enter the export market if they can demonstrate their credibility which largely determined by the quality of their products and services. Certification as one component of SQA is again the determining factor for software quality.
- Software quality for cost-effective
  It is a well known fact that quality system leads to increased productivity and reduced operating costs. The investment made in establishing a quality software system often more than compensated by the rewards of high productivity and low organisational costs. It is important to realize that the cost of correcting defects in software late in the development cycle is greater than the cost of correcting them early. Analyzing software processes to identify targets for improving initiatives through the technique of "cost of poor quality" can drastically increase profits.
- Software quality helps retain customers and increase profit
  Most customers will not tolerate poor quality software since it means more costs forcing them to choose another supplier. On the other hands better quality software leads to improved customer satisfaction, good customer relationship, and ultimately longer retention of the customer. The cost of attracting a new customer is high. Studies show that for a software company; a 5% increase in customer retention increased profits by 35% [7, p. 6].
- Software quality as hallmark of global business
  Quality is often used by world-class players as strategic thrust for growth and to outperform their competitors. The "chain reaction" as advocated by Deming [7, p. 7] suggested improve in quality will trigger productivity improvement. Cost decrease will result in price decrease. This in turn will increase sales; resulting in business growth. With business growth the organisation will be able to get better return to investment

III. THE PROPOSED SQA ARCHITECTURE

Value Chain Model developed by Michael Porter [8] is one of the powerful tool for competitive advantage. This model identified five generic primary value chain activities to create value that exceeds the cost of providing a product or service. The primary value chain activities are inbound logistics, operations, outbound logistics, marketing and sales; and service. Those activities are supported by support activities which include procurement, technology development, human resource management and infrastructure. We propose the SQA system to utilize the basic principles of the Porter Value Chain Model. The primary value chain activities identified are activities in the pre-project and project life cycle SQA components. The four supporting activities that underlie the entire value chain are Quality Infrastructure, Standards, Human Organisation and Quality Management.

As depicted in Figure 1 below, the SQA model comprises of six main components. The first two components are the primary value chain activities which are vital to create value that exceeds the production and maintenance cost of the software. They are the key ingredients for developing competitive advantage. The next four components comprising of infrastructure, quality management, standardization and human resources are the supporting activities. Although these activities may be viewed as "overhead"; cost advantage through the use information systems and information technology will normally offset the costs. The six components of the SQA system are as follows:

i) **Pre-project components:** These components are basically to ensure commitments of the projects undertaken have been properly defined considering resources required, the schedule and budget. At this stage project development plan and quality plan are also developed. IEEE Std. 730 requires minimum documentation for the plan to include software requirements description (SRD), software design description (SDD), verification and validation plans; and verification result report and validation results report.

ii) **Project life cycle activities assessment:** Two main phases of software system life cycle are development life cycle and the operation-maintenance phase. Quality assurance activities in development life cycle are devoted to detect design and programming errors involving formal design review, peer review, expert opinion and software testing. The quality components used during the operation-maintenance phase are geared to improve the functionality of system and maintenance tasks. In some cases, activities to ensure the quality of external participants in the development and maintenance should also be addressed

iii) **Infrastructure components for error prevention and improvement:** Infrastructure components are applied throughout the entire organisation with the objective of eliminating or reducing the rate of errors based on the organisation's accumulated experience. This objective is achieved with the use of tools such as procedures, supporting devices such as templates and checklists, training instructions, preventive actions, configuration management and documentation control.

iv) **Software quality management components:** The main objectives of these components are to control the development and maintenance activities. Early management intervention is important to prevent or minimize schedule and budget failures. Tools in this component includes project progress control, software quality metrics and software quality costs.

v) **Standardization, certification and SQA system assessment:** The main objectives of this component are to assess the achievements of quality systems according to common national or international standard benchmark besides improving the coordination of the organisational quality systems with other organisations. Two main categories of standards are usually used. They are management standards and project process standards

vi) **Human components of SQA**: These components include managers, testing personnel, the SQA unit and practitioners interested in software quality. These people contribute to software quality through their initiatives and support to detect deviations from SQA procedures and methodology; and finally suggesting appropriates improvement actions. Various approaches can be utilized, including the establishment of SQA Unit, SQA Committee or SQA Forum.

IV. CONSIDERATIONS GUIDING THE CONSTRUCTION OF AN ORGANISATION'S SQA SYSTEM

The development of software quality assurance system is greatly influenced by the characteristics of the organisation itself. These include the nature of software development projects, the maintenance services to be performed and the size and skill level of their professional staff. Hence different organisations need to customize their own quality systems using the building blocks discussed above. In general, considerations regarding the establishment of quality assurance system can be classified into two main categories; namely The SQA organisational base and the SQA components to be implemented within the organisation and the extent of their use.

A. Organisational considerations

Organisational factors determining the type of an SQA system includes the nature of software to be developed. Software clienteles may range from buyers of software packages, customers of custom-made software packages to internal clientele such as the organisation's departments or unit. The type of software maintenance clientele may differ substantially from software development clienteles. As an example a software house may employ a subcontractor to maintain its software packages sold to clients. Likewise an internal maintenance unit may be assigned to maintain software packages purchased or custom-made for the organisation. The range of products and the size of professionals employed by the organisation are other important considerations in the development an SQA system. In cases where the project involved corporation with other organisations in software production, more components of SQA should be considered. For the purpose of optimization, organisations must take into consideration on software quality, team

productivity, process efficiency and financial savings in devising an effective SQA system.

B. Project maintenance service considerations

Complexity and difficulties of software systems are normally determined by the algorithms applied, the project size, the variety of development tools used, interfaces to other software and firmware systems required. Hence these factors are the underlying considerations for effective SQA system. The degree of staff experience with project technology will determine the intensity of SQA components required. Higher proportions of software reuse allow for the reduction of SQA efforts and the employment of fewer SQA components.

C. Professional staff considerations

In general a highly qualified professional staff enables a reduction in the efforts required to complete and maintain a project. As big-scale software development work requires efficient effort, the level of acquaintance of the department with the team members becomes an important SQA consideration

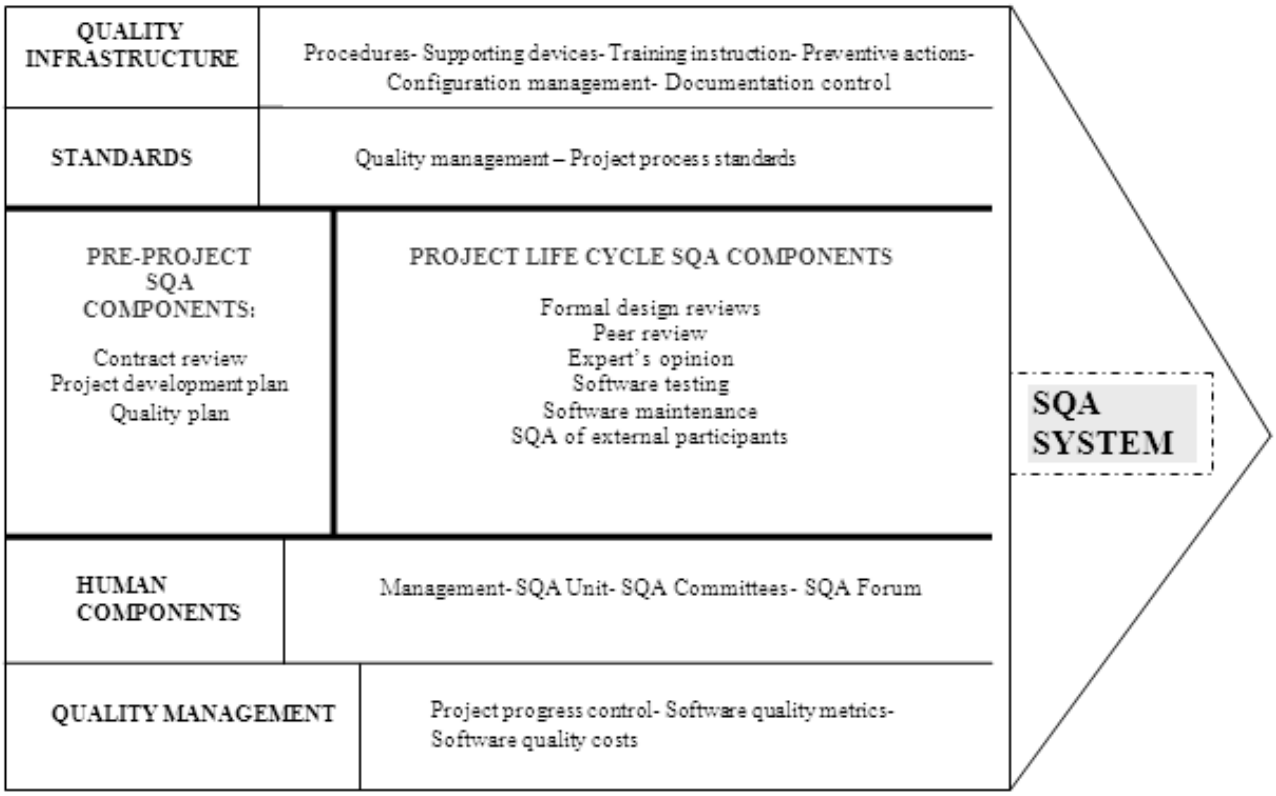

FIGURE 1: SOFTWARE QUALITY ASSURANCE BUILDING BLOCKS

V. CONCLUDING REMARKS

The SQA components and tools discussed in this paper are specifically aimed at the needs of professional software development and maintenance to address quality problem and in most cases to avoid painful losses. The application of these tools is dependent on the environmental characteristics of the organisation. Among others the characteristics are:

- Contract condition and commitment defining the contents and timetable of the project
- The nature of relationship between developers and customers
- Requirements to address teamwork complexity in the project

- The level of corporation and coordination with other software and hardware development teams both internally and externally
- The level of interface required with other software systems
- The commitment to continue carrying out the project when there is a change in team composition
- The need and level of maintenance for software system for a number of years after system handover

These environmental characteristics also apply to internal development of software and firmware where formal contract between customer and supplier may not exist. Monitoring the quality criteria should focus on the desirable and undesirable attributes of the software product and process and the criteria must be evaluated with a metric. The activities demand intensive and continuous managerial efforts in parallel to the professional efforts to ensure the success of the project with an acceptable level of quality

REFERENCES

[1] Shrum, S. Choosing a CMMI Model Representation. The Journal of Defense Software Engineering, Vol. 13, No. 7, July 2000

[2] Lohr, C. Software Quality, Software Engineering, Vol. 2, John Wiley, 2005

[3] Pressman, R. S. Software Engineering – A Practitioner's Approach. McGraw-Hill International, 2000

[4] Danial, G. Software Quality Assurance - From theory to implementation. Pearson. Addison Wesley, 2004

[5] Deming, W. E. Out of Crisis. MIT CAE. Cambridge, MA, 1989

[6] Schulmeyer, G. G. and McManus, J. I., Total Quality Management for software. International Thompson Computer Press, Boston, 1996

[7] Godbole, N. S. Software Quality Assurance, Principles and Practice. Alpha Science International Ltd, Oxford, U.K, 2006

[8] Porter, M. E. Creating and Sustaining Superior Performance. The Free Press, New York, 1998