European Journal of Business and Management
ISSN 2222-1905 (Paper) ISSN 2222-2839 (Online)
Vol.9, No.15, 2017

Software Quality: Concepts, Problems and Tools for Improving

Abeer Assiry, Manal Nasser, Yasmeen Aweed, Azrilah AbdlAziz
Department of Information Systems, Faculty of Computing and Information Technology, Jeddah, Kingdom of Saudi Arabia

Abstract
In this paper, we discuss the concept and principles of the total quality management (TQM). The paper describes the main Software Development Life Cycle (SDLC) models and pros and cons of each one. In addition, overview measures of quality during SDLC. Finally, the paper describes the software quality problems and how to achieve TQM via defect prevention.
Keywords: Quality; quality management; SDLC; software development; total quality management.

1. Introduction

The cost of software quality problems or software errors is a significant problem, not only to the vendor of the software but also to their customers or end users of the software. At the beginning, we introduce the quality concept which means continuous progress process of products and services to achieve customer satisfaction and met their needs. So that, the Customer satisfaction and product quality are tightly interrelated concepts and affect each other significantly [1]. The relationship between them is very strong, where the customer satisfaction increased as the product quality increased. This gives customers value and allows them to be members who decide the quality and cost of the products [1]. Organizations should have commitments to provide products and services with the degree of quality that meets the customer expectations. As the result, organizations will be evolved into what is referred to total quality management (TQM). The Total Quality Management (TQM) is a set of standards and rules that ensure the continuous improvement of the organization. The TQM approach may apply to any development process, in this paper we apply it to software development. For software development, the quality assurance is fundamental for both researchers and practitioners. Therefore, software quality increasing and ensuring is an essential concern. Since there are many software development approaches or models during software development process; such as waterfall model. Within a given software development life cycle (SDLC), TQM can be applied to increase the development process quality. This paper discusses TQM concepts for software development.

2. Literature review

Quality management effectiveness depends on the effectiveness with which performance and the results which are measured [2]. Also, to obtain the excellence in software develpment requires the software development community to look for new techniques, and concepts of Total Quality Management (TQM) which is essential here [3]. According to Helio Yang, Y. (2001). the functionality, integrity, reliability, maintainability, enhance ability, usability, portability, reusability of the software and the user interface appearance have effect on software quality [4].

Everhart states that, TQM methodologies have been successful in multiple fields and the reason for TQM fails in any organization is the individual organizations which treat TQM as a fad, give it lip-service, and depend on slogans to facilitate changes rather than assist actions [5]. In software development process, Automated tools take a great part for improving the software systems quality but few of them have success as expectations. Wherefore, the need for measuring software quality important when projects run over budget and schedule [6].

As Kanji (1996) pointed out, there is no standard method of how TQM should be implemented in an organization, and the organization committed to customer satisfaction and improvement varies from organization to another [7]. Parzinger, (2000) identified some measures as critical factors in the management of software quality [8].

According to, Kan, S. H., Basili, V. R., & Shapiro, L. N. (1994) the quality of product gains by meeting customer's needs and there are some stages must be followed to improve software development. Many companies adopted TQM to get customer satisfaction by studying customer needs deeply, identify the requirements, and implement it [9].

The software quality assurance techniques (SQA) in the development process described by Li, E. Y., Chen, H. G., & Cheung, W. (2000), but it is not enough to achieve the quality of software product needed by the customer. The using of TQM in software development includes the organization completely [10].

depending on Glowalla, P., & Sunyaev, A. (2015) the quality of the processes in development is important to get a successful software project and TQM provides a causal structure for quality improvement in software development [11]

3. Total quality management

3.1 Total quality management definition

According to Eldon Y. Li, Houn-Gee Chen, & Waiman Cheung (2000), the word quality is used to measure the degree of excellence. Quality is the degree to which a product lives up, acceptance, maintainable, and other characteristics that expected from the product. To produce products with high quality, the TQM concept should be instilled into product development process.

The word (Total) means all and everything in an organization, it involves all processes, services, jobs, resources, products, people, places and even time.

Total Quality Management (TQM) is defined as a management approach seeking towards success through customer satisfaction. It is based on the collaboration between all members of an organization to improve processes, products, services.

TQM benefits all members within an organization and the society in general.

There are so many methods for implementing TQM approach defined by quality leaders as Philip B. Crosby, W. Edwards Deming, Armand V. Feigenbaum, Kaoru Ishikawa, and J.M. Juran.

TQM also has been defined by The Department of Defense as: "both a philosophy and a set of guiding principles that represent the foundation for a continuously improving organization. TQM is the application of quantitative methods and human resources to improve the material and services supplied to an organization, and the degree to which the needs of the customers are met, now and in the future. TQM integrates fundamental management techniques, existing improvement efforts, and technical tools under a disciplined approach focused on continuous improvement" [12].

A very important point that we have to consider is that the TQM means many different things according to the organizations, that means its meaning differs from organization to organization and even it may differ within the same organization. Nowadays, TQM is one of the most common concepts but no one can find an accurate and concise definition for it or know how to achieve it exactly. This can be observed clearly in Japanese and American companies where the Japanese companies have national quality regulations and standards, while American companies are allowed to set their own standards individually without any limitations. TQM can be achieved by a lot of approaches, techniques, and tools; there isn't one specific path or way to achieve quality with its wide meaning.

There are so many initiatives to find approaches that achieve the TQM in organizations such as: Six Sigma, Benchmarking, and Hoshin Planning.

Total Quality Management concept can be defined as a combination of systems, organizations, and tools in which all members within the organization's departments work to maintain or improve quality, cost, and procedures to give customers high qualified products and services with less price and more benefits [13].

3.2 Software quality

Previously, the software quality is evaluated by its properties or capabilities such as reliability, integrity, maintainability, scalability, portability, and reusability. In some studies, the functionality and the user interface appearance may have effects on the software quality. So that, software quality can be directly or indirectly measured with the help of that attributes. These characteristics could affect the customer satisfaction, on the service or product the organization produced, which is already a very important indicator for the quality of the products [14].

There is no clear definition given for Software quality, hence make it complex to exercise. However, David Chappell (2012) said software quality can be segregated into three main aspects; functional, structural, and process quality.

Functional quality is the tasks that software should correctly perform and it is expected to do for its users.

Second aspect is structural quality, means the quality of the code itself, is should be well structured. Structural quality is difficult to test not like functional quality which is can be tested easily.

The third aspect, process quality, this aspect defines the development process quality, development teams, and sponsors [15].

4. Total quality management for software development

Quality is important matter in the process of software development. TQM philosophy can be used in a software development quality problem as any other product development. TQM gave a conceptual framework for software quality achievement.

Here we provide an overview of software development life cycle principles. We then go on to discuss the quality measures during SDLC.

4. 1 Software development life cycle (SDLC)

The process of Software development simply is a structure used to create a software. This process has several models, each describing the activities that is taken during the process of development. Structured programming, Client-Server, n-Tier, Object-Oriented, Event Driven, and Service-Oriented Development of Applications are all examples of constructs or software architecture pattern that used to deliver better systems. A methodology is the process or method that is used (utilized and controlled) in the development process [16].

The growing of software development organizations make a need to improves the development process. Without quality project management, software projects can be delivered late, over budget or fully fail. Of course, if the software products not meeting their expectations in the functionality, cost, or delivery schedule, means it lacking effective project management [17].

SDLC model is a framework describes set of activities performed at each stage in the project of software development. This is a brief description of the most common SDLC models [18]:

1. Waterfall Model: this model is the simplest SDLC methodologies and it is the oldest one. Using it the project finish one phase, then move to the next and each stage depends on the information from the previous one and no going back. This model used when all the requirements known. It is easy to understand and simply managed, but any delays can effect on the entire timeline of the project. Also, any problem in the early stages can't be fixed until reaching to the maintenance stage, because there are little revisions once a stage is completed.
2. V-Shaped Model (Verification and Validation model): it is considered as an enhanced version of waterfall model; each stage begins after the previous one has ended (such as waterfall). It contains a testing phase with each development stage. This model is useful when some of project requirements unknown, and it's still difficult to go back to change.
3. Iterative Model (repetition model): it is not start with clear and fully known requirements, instead it start with a set of software requirements implement it, then test, evaluate and then define the requirements of the next stage. At the end of each phase or iteration a new version of the software is produced. Complete repetition until arriving to the complete system (last version). This model gives you a working version early and it easier and less cost to make changes. But resources will have consumed by process repeating.
4. Spiral Model: it combines the idea of iterative model with the aspects of systematic, controlled of the waterfall model. A software project repeatedly passes through its four phases in iterations or Spirals which enable a multiple round of refinement. Spiral model good for building a customized product, it is one of the most flexible methodologies. Also, user feedback early considered. But the risk is the probability of entering in never-ending spiral for a project.
5. Big Bang Model: This model is a high-risk model where it follows no specific process with little time for planning and the customer may not have clear requirements. Most resources are used for the development. It is usually used for small projects, it is not suitable for large and complex projects, because it is a high-risk model.
6. Agile Model: project is divided into cycles. This model produces releases, with small, incremental changes in each one from the previous. At each iteration, the product will be tested and it gave an early working version. In this model the customers, developers and testers work together as a one development group. Here the project can go in a wrong way if the customer is not having clear requirements since it depends on the customer interaction.

4.2 Quality measures during the software development life cycle (SDLC)

The software quality is estimated by many attributes such as availability, integrity, reusability, extensibility, usability, portability, and flexibility. Also, the software quality could be affected by the software functionality and the user interface appearance. [19]

It is important to use suitable quality measures during the SDLC, to make a good quality software. System development life cycle is made up of set of phases each has a specific quality measures as described here:

1. Requirements analysis: many studies indicate that more than 60% of system development failures are a result of user requirements misunderstanding. Understanding and specifying the customer requirements is a costly part in the software development. The software program developer should have a complete and clear set of functional requirements. Programmers work to understand customers' needs and report the level of customer satisfaction on the product. Usually, software vendors use quality function development in the requirement analysis stage. Software Quality Function Deployment is a requirement gathering tool. It is a useful method, used to improve the quality of the software development process by implementing appropriate quality improvement approaches to the SDLC requirements [20]. Quality Function Deployment (QFD) used to capture and implement the customer's needs correctly. During the development of customer's requirements, attention to the following issues should be taken to avoid a weakness quality indicator:

- Assign the responsibility of establishing and planning of the customer requirement gathering.
- Take some methods to agreeing on the requirements and approving changes.
- Gave efforts to prevent misunderstandings such as well terms definition.
- Save and review discussion results.

2. Systems Design: this is the costly and lengthy stage. It is the most critical stage of quality software development because a lack in design is costlier to correct than a lacking in the other stages. Deficient design is a major reason of quality problems. Also, the essential quality aspects such as safety, performance, and dependability of the software are established during the design and development phase. [20] [21]
Design planning should include the following roles:

- Work schedules should be sequential and parallel with the timescales.
- Evaluating the safety, performance, and dependability in the software design;
- Using methods for software measurement, test, and acceptance criteria;
- Design verification activities;
- Clear responsibilities assignment.

The design of the software may be changed or modified for some reasons, errors for instance or modifying some requirements. Concurrent engineering is useful method of implementing TQM, it is widely-used to change systems design. [22]

3. Systems development: TQM in Software requires appropriate integration of quality into the process of software development. After establishing an effective quality process into the first stage and second stage of SDLC, the task of coding becomes simple and faster. The design and code inspections approach can be used for document inspections process. Also, Control charts can be used to track the metrics code inspections performance. [20] [21]

4. Testing activities must be arranged and managed properly right from the begin of the software development process, it is not a separate process in SDLC. It is an integral part of development. Otherwise, it will start later than it should and lacking may increase and the correction process will be costlier. TQM based software development process should have testing objectives. Testing process is not limited to specific activities but it can contain the following activities for improving: [23] [20]

- Prepare for the testing by organizing the team, testing objective, establishing test environment etc.
- Refactoring the test schedule if needed.
- Manage the test execution process and monitors the metrics.
- Documents the defects, write a test report and publish metric graphs.
- If the tested part met the criteria, go on in the project. Otherwise, correct and prepare for a test again.

5. Implementation and Maintenance: Most of the maintenance activities are reactive. Where programmers frequently zero in on the immediate problem, repair it, and wait until the next problem occur. As statistical process-control can be used to monitor the quality of software system maintenance, a TQM-based system must adapt to the statistical process-control to assure maintenance quality. [21]
To improve the software development process, the company should develop metrics to identify where improvements can be made. The following are some of the measurements type that can be considered:

- Defects: Total number of defects found and the time taken to fix them.
- Test coverage: Percentage of code that is covered by tests.
- Number of requirement changes: this measure reflects the quality of requirement specification.
- Work effort: this is measured in terms of cost, schedule, and performance.

5. Software quality problems

Bad quality is not a certain attribute of software. It come from several well-known causes. It can be predicted and controlled, but first you should understand and address these causes.
Many of important business processes start to implement in software, but first you should now that the quality problems are a primary business risk. Here are five primary problem of software quality and how to reduce their damaging impacts by using methods rather than testing after implementing [19].

5.1 Lack of domain knowledge:

The big problem in software quality is that the most developers don't have any experience in the business domain that will produce functions to their applications. But they will gain the knowledge about the domain over time, however, most of this knowledge will wasted on correcting error caused by poor understanding of the functional requirements. The best way to reduce this problem is to extend the access to business domain experts, and provide some orientation to developers in business domain, also the parallel reviews with domain experts can help to pass this poor domain knowledge.

5.2 Lack of technology knowledge:

Many of developers have good experience in different computer technologies and programming languages. However, recent multi-tier architecture of some business applications is very complicated to many programing languages and software platforms. So, few developers know all latest programing languages technologies and platform. Wrong assumptions about how the technologies work is a major cause of the non-functional errors that

increase damaging outages, data corruption, and security issues during operation.

The best way to reduce this problem is to cross-train developers in different software technologies, provide parallel reviews with other developers in other software tiers, and do some static and dynamic analyses of the code.

5.3 Unrealistic schedules:

When developers are forced to sacrifice of the software quality to finish the software within the deadline of schedules, the outcome will be mostly bad. Because working under pressure make developers more stress and make more mistakes. The only way to reduce this problem is to apply strong project management practices. Controlling obligation through planning, tracking progress and address end requirements.

5.4 Badly engineered software:

Most of software development process include changing or enhancing existing code. There is always unnecessarily complex code and it leads to many errors and unexpected negative side effects when it modified. The best way to reduce this problem is to re-factor risky section of the code planned by information from architectural and static code analyses.

5.5 Poor acquisition practices:

Many of big multi-tier software are construct by different teams, some of them may be outsourced from external companies. Consequently, the acquiring organization has little visibility over the quality of the software that are receiving. To reduce the danger of quality problems in outsourced software, acquiring managers should implement strong quality assurance for delivered software.

6. Achieving TQM via defect prevention

To achieve product quality, the software development community has relied mainly on defect detection and correction through reviews and inspections early in the development lifecycle and through extensive testing and checking during the final stages. However, relying only on the defect detection processes after they are created is extremely costly. Rework and testing require a significant amount of resources. In addition, finding and removing defects does not, in itself, add functionality to the end product, nor does it improve the overall process. A more effective approach to software development relies on preventing defects from being created throughout the lifecycle. Fewer defects permits more resources to be devoted to productive, value-added activities while resulting in greater product quality and associated customer satisfaction.

Described in this section is a defect prevention program which addresses defects from all phases of the development life cycle from customer requirements, spec, design, and test through field support. The defect prevention program focuses on and addresses the causes of defects and facilitates the enhancement of processes, tools, and methodologies required to eliminate defects and to ensure continuous improvement.

A basic model of a defect prevention process can apply to any phase of SDLC. This process contains four primer activities which are integrated into the development phase. The primer activities are: (1) causal analysis meetings, (2) an action team, (3) a stage kick-off meeting, and (4) a data collection and tracking system. The causal analysis meetings are to identify the root cause of errors and suggest preventive actions. The action team ensures that suggested actions are implemented [15].

The stage actions include process improvements, process document updates and corrections, education training, tool enhancements and good communication techniques. The stage kick-off meetings are held at the beginning of each stage to prepare the team for the work of that stage and to provide feedback from previous causal meetings. The data collection and tracking system facilitates the tracking of suggestions and associated action status.

Additionally, the mechanism should be in place to refer or transfer inherited defects found at any stage of the process to the stage of origin where causal analysis for the root cause is done and suggestions for preventive actions are made. For example, specifications defects found during the coding stage should be referred to the specification group to determine the main cause of the specification defect and analyze what can be done in the specification process to ensure that this type error never occurs again. This inter-functional defect referral aspect of the defect prevention process when integrated into the software lifecycle provides an important means of maintaining ongoing customer focus into the process. [24]

6.1 Tools for improving software quality

There are three part of software quality. Enhancing techniques of software quality need to addressing all of them, these three parts are: Functional quality, structural quality and process quality. Functional quality is essential but also, we need tools focused on structural and process quality [15].

Tools that is used to improve the software structural quality and produce services such as refactoring,

which allows the developer to enhance and organize the code without change it.

Tools for structural quality can help in static code analysis, examining code for security issues, along with analysis of dynamic code, which contains performance side and measures of test covering.

Tool for enhancing process quality which help in controlling the development process. They include assist for tracking the process status, by mapping requirements with progress measurements for the developer. Process quality tools can also produce insight into code, i.e. the number of lines added each week, improve finding and fixing bugs and test plan progress. These tools should be accessible through less technically focused interfaces and making them available to all developers.

Tools aren't the all thing in software quality, activities such as group code reviews and effective management can have big effect on many aspects of software quality [12].

7. Conclusion

Total quality management can be applied to the software development life cycle to improve the quality. You should continually improve your software processes after applying the TQM concepts. There are different tools to implement TQM to improve software quality. Also, good understanding of TQM implementation will help in improving all process of software development and result fault-free and good functionally software that and cover all needs of the users. Finally, the paper discusses achieving TQM via defect prevention that focuses and addresses the causes of defects and facilitates the enhancement of processes, tools, and methodologies required to eliminate defects and to ensure continuous improvement of software.

References

[1] I. Fečiková, "An index method for measurement of customer satisfaction," The TQM Magazine, vol. 16, no. 1, pp. 57-66, 2017.
[2] G. K. Kanji, Measuring Business Excellence, London: Routledge, 2005.
[3] T. Bradley, "The use of defect prevention in achieving total quality management in the software life cycle," in IEEE, Denver, CO, USA, USA, 1991.
[4] Y. H. Yang, "Software quality management and ISO 9000 implementation," Industrial Management & Data Systems, pp. 329-338, 2001.
[5] R. Everhart, "Applying TQ principles to the requirements phase of system development," in IEEE Xplore, Singapore, 2002.
[6] N. Ashrafi, "A decision making framework for software total quality management," in International Journal of Technology Management, 1998.
[7] G. Kanji, " Implementation of total quality management," Total Quality Management, pp. 331-343, 1996.
[8] M. J. a. R. N. Parzinger, "A study of the relationships between total quality management implementation factors and software quality," Total Quality Management, pp. 353-371, 2000.
[9] V. R. B. L. N. S. S. H. Kan, "Software quality: An overview from the perspective of total quality management," IeeeXplore, pp. 4 - 19, 1994 .
[10] M. C. &. C. T. Lee, "Applying TQM, CMM and ISO 9001 in knowledge management for software development process improvement," International Journal of Services and Standards, pp. 101-115, 2005.
[11] P. &. S. A. Glowalla, "Influential Factors on IS Project Quality: A Total Quality Management Perspective.," 2015.
[12] E. Y. a. C. H.-G. a. C. W. Li, "Total quality management in software development process," The Journal of Quality Assurance Institute, vol. 14, pp. 4--6, 2000.
[13] IEEE, "The use of defect prevention in achieving total quality management in the software life cycle," in Communications, 1991. ICC'91, Conference Record. IEEE International Conference on, 1991, pp. 356--359.
[14] G. H. a. J. J. J. a. K. G. Subramanian, "Software quality and IS project performance improvements from software development process maturity and IS implementation strategies," Journal of Systems and Software, vol. 80, pp. 616--627, 2007.
[15] D. Chappell, The Three Aspects of Software Quality: functional, structural, and process, Technical report, Microsoft Corporation, 2012.
[16] A. J. Ms Namrata Jain, "Software Development Life Cycle: A Detailed Study," International journal of advanced research in computer science, pp. 261-264, 2011.
[17] S. Bhattacharjee, "Software Development Life Cycle," [Online]. Available: http://cab.org.in/Lists/Knowledge%20Bank/Attachments/83/SDLC.pdf.
[18] K. Roebuck, Systems Development Life Cycle (SDLC), Emereo Publishing, 2012.
[19] Y. H. Yang, " Software quality management and ISO 9000 implementation.," Industrial Management & Data Systems, vol. 101, no. 7, pp. 329-338, 2001.
[20] B. S. Dhillon, Computer System Reliability: Safety and Usability, 2013.
[21] B. S. Dhillon, Applied Reliability and Quality, 2007.

[22] ISO 9004-1:1994 - Quality management and quality system elements Part 1: Guidelines, International Organization for Standardization, 1994.
[23] M. Kevitt, "Best Software Test & Quality Assurance Practices in the project Life-cycle," Dublin City, April 2008.
[24] T. J. Bradley, "THE USE OF DEFECT PREVENTION IN ACHIEVING TOTAL QUALITY MANAGEMENT IN THE SOFTWARE LIFE CYCLE," Northem Telecom, Inc., p. 356, 1991.
[25] M. Imai, Kaizen: The Key To Japan's Competitive Success, McGraw-Hill Education, 1986.