Total Quality Management for Software Development

Amal Alhassan
Department of Information Systems,
Faculty of Computing and Information Technology,
Jeddah, Kingdom of Saudi Arabia

Weam Alzahrani
Department of Information Systems,
Faculty of Computing and Information Technology,
Jeddah, Kingdom of Saudi Arabia

Azrilah AbdulAziz
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 successful the total quality management (TQM) implementation. The paper briefly explains the similarities between software development process and product development process. In addition, overview quality measures during the software development life cycle (SDLC). Finally, the paper describes the Deming's quality management method and his fourteen points to implement TQM. The paper discusses how to apply Deming's method in software development process and provides recommendations to ensuring success during TQM implementation.

Keywords

Quality; quality management; software; development process; software development; total quality management;

1. INTRODUCTION

Quality is a continuous improvement process of products and services with a focus on the customer’s satisfaction (Wang, (1995)). The Customer’s satisfaction and quality are interlinked, and these create value for the customer and help him/her to make a decision whether the products or services justify their cost (TQM for IT IS). A commitment from the entire organization is required in ensuring that the products and services have the quality they have been designed for. This approach of quality throughout the entire organization has evolved into what is referred to total quality management (TQM). The Total Quality Management (TQM) is a set of guiding principles that represent the basis of continuously improving the organization. The TQM philosophy can be applied to any development process, be it product development or software development. For software development, the quality is fundamental for both researchers and practitioners. Therefore, the enhancing of software quality is a paramount concern. There are various software development approaches during software development process; these approaches are also referred as “Software Development Process Models,” like Waterfall model. Within a given software development life cycle (SDLC), TQM can be applied to increase measured quality. Deming’s fourteen points of management approach provide guidelines for implementing the TQM concept. This paper sheds light on TQM concepts for software development.

2. LITERATURE REVIEW

The effectiveness of quality management depends on the effectiveness with which performance and results are measured (Kanji, (2002)). According to Bradley, T. J. (1991, June). To achieve the excellence requires the software development community to regularly look for new techniques, and the concept of Total Quality Management (TQM) is fundamental in this effort. Helio Yang, Y. (2001). Pointed out that the functionality, reliability, integrity, maintainability, enhance ability, usability, portability, reusability of the software and the appearance of the user interface could affect software quality. In addition, (Everhart, (1995, June)) state that if TQM in organizations fails, it is because individual organizations treat TQM as a fad, give it lip-service, and rely on slogans to facilitate change rather than sustaining actions. According to Everhart, R., La Salle, A. J., & Khorramshahgol, R. (1995, June)., in software development process, Automated tools (i.e., CASE systems) held great promise for improving the quality of software systems but few have lived up to early expectations. Wherefore, the need for measuring software quality becomes prominent when projects are running over budget and schedule ((Ashrafi, (1998)). (Parzinger, (1998)) ;(Parzinger, (2000)) identified some metrics or measures as critical factors in software quality management

According to, Kan, S. H., Basili, V. R., & Shapiro, L. N. (1994). The quality of product obtains by satisfying customer's needs. To improve software development, there are stages must be followed. Many companies adopted TQM to get customer satisfaction by studying their needs, gathering requirements, and fulfillment. Moreover, Li, E. Y., Chen, H. G., & Cheung, W. (2000). Describe the software quality assurance techniques (SQA) in the development process, is not enough to achieve the quality of software product demanded by the customer. Apply TQM for software development includes entire organization. Software development teams provide an on-the-job training program to workers to gain experience and knowledge. Nevertheless, the communication gap between the developers and the users of software products affect the real implementation of TQM. However, Glowalla, P., & Sunyaev, A. (2015). Referred to the failure of software and information systems programs today’s, is a big concern for researchers. The quality of the product and processes in development is important to a successful software project. TQM provides a causal structure to enhance continuous quality improvement for software development.

3. TOTAL QUALITY MANAGEMENT

3.1 Total Quality Management Definition

TQM is a management approach aimed at satisfying all the customer requirements, needs and expectations using a continuous improvement approach (Wang, (1995)). According to Lee, M. C., & Chang, T. (2005). The word "total" involves everyone and all activities in the company, quality means conformance to meeting customer requirements, and management means quality can and must be managed. It focuses on continuously improvement of ability to deliver high-quality products and services to customers. It suggests that any improvement that is made in the business, be it a better design of a component or a better process of a system, will help to improve the “total quality” of the organization and the quality of the final product (Li, (2000))TQM is the foundation for activities, which include:

- Commitment by senior management and all employees
- Meeting customer requirements
- Reducing development cycle times
- Just in time/demand flow manufacturing
- Improvement teams
- Reducing product and service costs
- Systems to facilitate improvement
- Line management ownership
- Employee involvement and empowerment
- Recognition and celebration
- Challenging quantified goals and benchmarking
- Focus on processes / improvement plans
- Specific incorporation in strategic planning

The TQM principles can be grouped into the following practical and common sense concepts (Wang, (1995)):

1. Customer focus (internal and external customers)
2. Leadership (management role changes to active leadership)
3. Teamwork (multidisciplinary teams, include involvement of customers and suppliers)
4. Continuous improvement process.
5. measurement (the improvement process is based on quantitative and qualitative metrics) and
6. Benchmarking as a driver to improvement in a competitive environment.

3.2 Meaning of Quality

According to Oxford American Dictionary defines quality as "a degree or level of excellence". Therefore, quality is defined and judged by the customers. The "official" definition of quality by the American National Standards Institute (ANSI) and the American Society for Quality Control (ASQC) is "the totality of features and characteristics of a product or service that bears on its ability to satisfy given needs." Obviously quality can be defined in many ways, depending on who is defining it and to what product or service it is related. However, in this paper we attempt to gain a perspective on the dimension of Software quality.

3.3 Software Quality

There are three aspects of software quality: functional quality, structural quality, and process quality (Chappell, (2013)). Functional quality means that the software performs the tasks it is intended to do for its users correctly. Software testing commonly focuses on functional quality.

The second aspect of software quality, structural quality, means the code itself is well structured. Unlike functional quality, structural quality is hard to test for.

The third aspect, process quality, it is the quality of the development process significantly affects the value received by users, development teams, and sponsors, and all three groups have a stake in improving software quality. However, the quality of software is estimated by many of its attributes such as reliability, integrity, maintainability, enhance ability (extensibility), usability, portability, and reusability (Subramanian, (2007). According to Helio Yang, Y. (2001)., the functionality of the software and the appearance of the user interface could also affect software quality. Consequently, these characteristics can affect user satisfaction so it could be used as a measure of software quality.

4. TOTAL QUALITY MANAGEMENT FOR SOFTWARE DEVELOPMENT

The TQM philosophy described above can be applied to any development process, be it product development or software development. Software development is a process in which the developer precisely converts the requirement specifications into software products.

4.1 Product Development Life Cycle (PDLC)

Product development life cycle is a systematic and orderly approach to managing product development activities. It usually follows the problem-solving steps prescribed by Herbert A. Simon: intelligence, design, choice, and review (Joshi).

The development of a new product begins with the stage of requirements analysis through collected the needs of customers, analyzed, and evaluated. Based on the customers' needs and the product specifications, design blueprints of the product are developed during the design stage. According to these blueprints, prototypes of the product are built and tested to evaluate the quality of the prototypes. If a prototype fails the test, the cause of failure is analyzed and identified. In case of failure, the project of developing this new product might have to be canceled (Joshi).

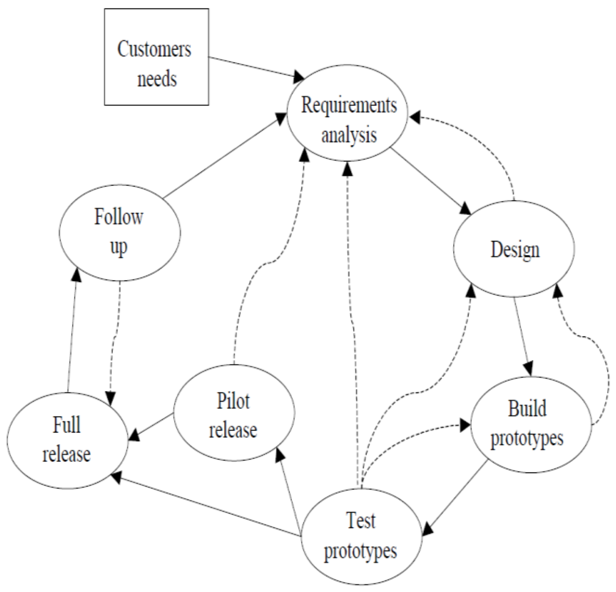

Figure 1 Product development life cycle (Li, (2000))

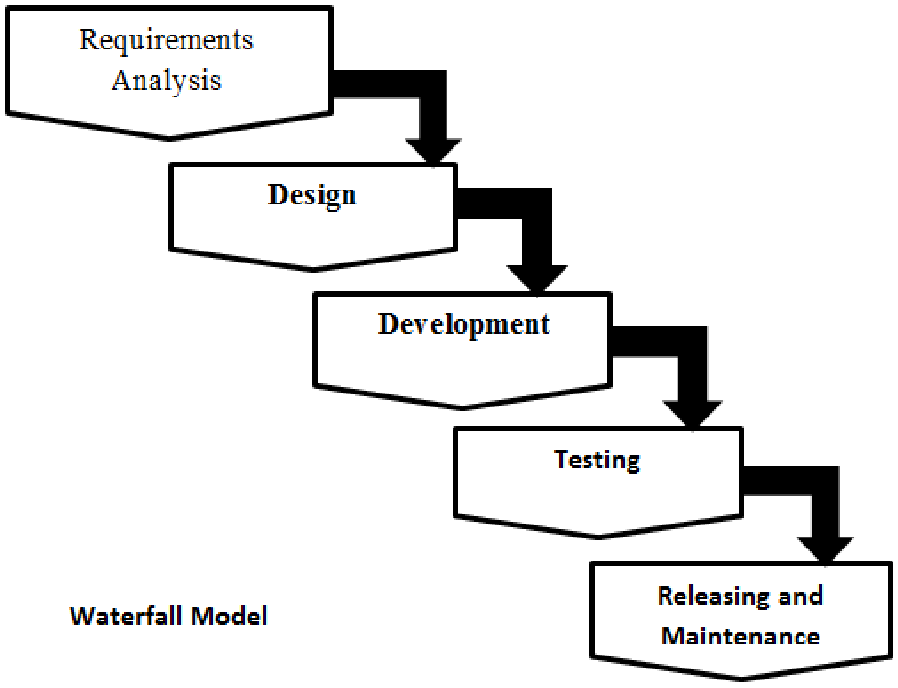

Figure 2 Waterfall model approach (Joshi)

The dashed lines in Figure 1 illustrate the cause of failure is sequentially fed back to the stage where the faulty process be. Once the prototypes passed all the tests, the best one is selected for either a pilot release (a limited scale release to testing) or a full release. If the follow up report indicates that product is successful, gives signal to send back to the full release stage for continuing the production. In the otherwise, the requirements analysis process is triggered once again and the entire product development life cycle is repeated (Li, (2000)).

4.2 Software Development Life Cycle (SDLC)

A software development life cycle resembles the product development life cycle. It usually incorporates the steps of planning, analysis, design, implementation, and support. Each step of software development is divided into separate process phases as Shown Figure 2 "The Waterfall" approach This approach discouraged iterations between phases in the process (Joshi) (Li, (2000)).

A software development life cycle may follow a structured development methodology (SDM), a rapid prototyping methodology (RPM) or a spiral development and enhancement method (SDEM). The SDM typically is applied to a system with clear requirements definitions, well-structured processing and reporting, and a long and stable life expectancy (Li, (2000)).

Table 1 shows details of the SDM process. Under this methodology, iterations between phases in the process are strongly discouraged. It is therefore called a "waterfall" approach. On the contrary, the RPM process allows and encourages such iterations. The SDEM combines the RPM process with the SDM process to shorten the development time required by a project adopting the SDM process.

Table 1 Detailed Phases of Structured Development Methodology (SDM) (Li, (2000))

PDLC Phases	SDM Phases	Phase Objectives
Requirements Analysis	Service Request/Project Viability Assessment	To initiate a project and conduct cost/benefit analysis as well as feasibility study.
	System Requirements Definition	To define project scope, analyze the existing system, and define information requirements, data attributes, and system objectives.
Design	System Design Alternatives	To identify and evaluate alternate system designs and prepare initial project schedules.
	System External Specifications	To specify data flow, user/system interface, system controls, and manual supporting procedures.
	System Internal Specifications	To specify processing logic, file structure, module interfaces, and system architecture.
Build Prototypes	Program Development	To transform programs' internal specifications into program code using a computer language.

Test Prototypes	Testing	To verify and validate the system being developed throughout the system development life cycle.
Pilot/Full Release	Conversion	To convert the data formats and procedures for the new system.
	Installation	To install the hardware and Software for the new system, and cutover the system into production.
Follow Up	Post Implementation Review/Maintenance	To monitor and maintain the quality and performance of the new system.

4.3 Quality Measures during the Software Development Life Cycle (SDLC)

To Produce a quality software product; it is important to take appropriate quality measures during the software development life cycle. A software development life cycle is made up of five stages, as discussed here:

i. Requirements analysis. Past experiences indicate that about 60%-80% of system-development-related failures are due to poor understanding of user requirements. In this regard, during the software development process, major software vendors normally use quality function development (QFD). Software quality function deployment (SQFD) is considered a very useful method to focus on improving the quality of the software development process by implementing appropriate quality improvement approaches to the SDLC requirements solicitation phase. In other words, SQFD is a front-end requirements collection method that quantifiably solicits and defines the customer's critical requirements (Dhillon, (2013)).

ii. Systems Design this is the most critical stage of quality software development because a defect in design is hundreds of times more costly to rectify than a defect during the production stage. More specifically, it means that every dollar spent to increase design quality has at least a hundred-fold payoff during the implementation and operation stages. Concurrent engineering is a widely used method to change systems design and also it is a useful method of implementing total quality management (Dhillon, (2007)) (Dhillon, (2013)).

iii. Systems development. Software Total quality management (TOM) requires the proper integration of quality into the total software development process. After the establishment of an effective quality process into the first stage and second stage of SDLC, the task of coding becomes simple and straightforward. However, for document inspections, the design and code inspections approach can be used. Furthermore, control charts can be utilized to track the metrics of the effectiveness of code inspections (Dhillon, (2007)) (Dhillon, (2013)).

iv. Testing activities must be planned and managed properly right from the start of software development. In addition to designing testing activities with care at each stage of the SDLC. Furthermore, a TQM based software development process must have a set of testing objectives. A six-step metric driven approach can fit quite well with such testing objectives are as follows (Dhillon, (2013)):

1. Establish structured test objectives.
2. Select appropriate functional methods to derive test-case suites.
3. Run functional tests and assess the degree of structured coverage achieved
4. Extend the test suites until the desired coverage is achieved.
5. Calculate the test scores.
6. Validate testing by recording errors not found during the testing process

v. Implementation and Maintenance: Most of the software maintenance activities are reactive. More specifically, programmers frequently zero in on the immediate problem, fix it, and wait until the occurrence of the next problem. As statistical process-control (SPC) can be used to monitor the quality of software system maintenance, a TQM-based system must adapt to the SPC process to assure maintenance quality (Dhillon, (2007)).

According to Kaizen, I. M. (1986)., the goal is increasing quality, By instilling TQM's continuous improvement strategy in every aspect of the software development, an organization never settles for the level that it has reached, no matter how good the product is. Many software companies are starting to implement such culture into their organizations and empowering their employees with the ability to help make improvements even at entry level positions.

5. DEMING MANAGEMENT METHOD

5.1 Describes TQM to Software Development Process

One of Deming ideas was the Plan, Do, Check, Action, which is often shortened to PDCA. We can use PDCA cycle as the basic idea, for software processes development to gain software quality [Figure 3].

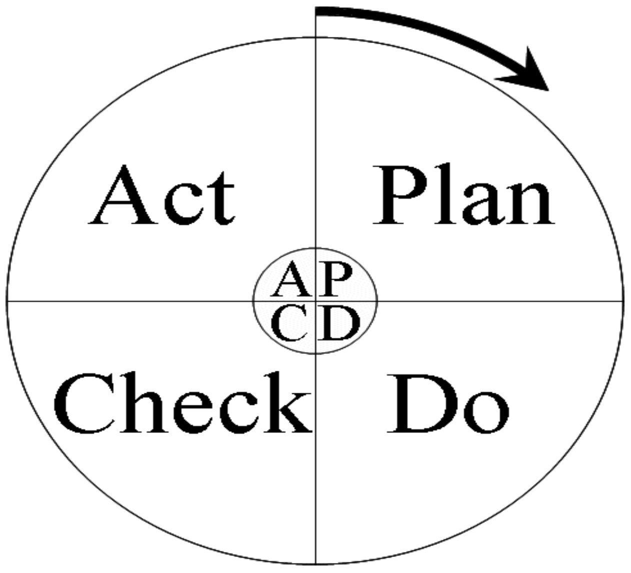

Figure 3 Deming‘s PDCA cycle

Applying TQM to software development, a process can control software quality and productivity, and select a suitable tool that can strengthen the capability of software quality policy, quality awareness, prevention, correction, and feedback. Table 2 describes TQM to software development process (Lee, (2005)).

In Table 2, there are six processes for software development. System planning includes a process of definition, analysis, specification, estimation, and review. The objective of software requirements analysis is the process of discovery and evaluation. Software design process is a process through which requirements are translated into a representation of software. Programming languages and coding translate a detailed design representation of software into a programming language realization. Software testing of design that test systematically uncovers different classes of errors. Software maintenance expands all variable resources maintaining the old system. As shown in Table 2 several TQM development activities involve Business System Planning (BSP), Quality Function Deployment (QFD), and Critical Success Factors (CSF). QFD is cross-functional so that all departments work together to achieve the common goal of satisfying customer demands. CSF is to developing the systems for planning and control. BSC is a method for translating strategy into action and has been successfully implemented in all kinds of companies all around the world. With the integration of TQM activities, the organization develops the software systems can produce a quality assurance plan and successfully carry out all the tasks involved (Lee, (2005)).

Table 2 Describes TQM to software development process (Lee, (2005))

Step	*Item*	*Objective*	*Task*	*Tool*
Plan	Software planning	A process of definition, analysis, specification, estimation and review	• Identify problems • Feasibility study. • Defined software project.	• Brainstorm • CSF • BSP • CSF
Do	Software Requirements analysis	The requirements analysis task is process of discovery and evaluation	• Information flow. • Information structure. • Software requirements specification	• Techniques • CASE tools • Training
	Software design process	Software design is a process through which requirements are translated into a representation of software	• Design process. • Transform analysis. • Transaction analysis. • Data structure design.	
	Programming languages and coding	Translate a detailed Design representation of software into a programming language realization	• Program coding. • Unit testing.	
Check	Software testing	To design tests that systematically uncover different classes of errors	• Valid testing. • Validation testing.	• Static testing. • Dynamic testing.
Action	Software maintenance	It is expanding all variable resources maintaining old system	• Preventive maintenance. • Corrective maintenance. • Adaptive maintenance	• Performance award. • Feedback.

5.2 Deming's Fourteen Points to Software Development

Deming as a guru of TQM adopted fourteen points of management approach that provides guidelines for implementing the TQM concept. These fourteen points can apply to managing software development processes.

1) Create constancy of purpose for improvement of product and service. Software development process traditionally ends when the completed system is handed over to the support group and put into production mode. The development team should be responsible for what they delivered, not the support group. Any quality problem occurs

During the production should be addressed to the development team. Management must (Li, (2000)):

- Establish operational definitions for each step in the software development process.
- Define what is meant by "service to the customer."
- Define standards of development, maintenance, and service for the next years.
- Define the internal and external customer.
- Develop ways to provide better systems and services in less time, using fewer resources.
- Invest tools and techniques to gain quality for software development.

2) Adopt the new philosophy of total quality. The Quality is everyone business. The manager is part of the quality team, not just the worker. In the TQM culture, the quality comes first, and everyone from top to bottom. Must embrace the TQM concept and communicate their support to all members of the software development team (Li, (2000)).

3) Cease dependence on mass inspection to achieve quality. Quality is built in, not added on. It's better to prevent the errors in code or process by experience and knowledge only. Management must install programs to improve software development processes continually. Examples of such programs are job training and job incentive programs (Li, (2000)).

4) End the practice of awarding business based on price tag alone. Many software organizations today are outsourcing their projects to subcontractors. It is important not to award a software contract based on price tag alone. Quality is more important than the difference in costs. Low quality in the long-term will result in a high total cost. It is better to create a long-term relationship with a few loyal and trustworthy suppliers who can produce quality code (Li, (2000)).

5) Improve constantly the systems of production and service. System development processes must continually be improved by introducing new and working methodology, paradigm, standards, practices, techniques, tools, policies, and procedures. All these require the organization to keep tracking the best practice constantly. Each staff member is required to improve oneself by updating or even expanding one's skill set (Li, (2000)).

6) Institute training on the job. In the quality of software, the development team must have appropriate experience and knowledge. The on-the-job training program is an effective means of obtaining such experience and knowledge. In the broadest sense, all staff members must know what their jobs entail and how to do their work. Management must assess the skill level of an employee before he or she is assigned to a software project. Different skill levels can play different roles and assume different responsibilities in a project (Li, (2000)).

7) Institute leadership. Management must lead, not punish. It is manager's job to help staff do a better job and create a better system. Project managers must be trained in basic interpersonal and analytical skills. They must have a solid understanding of statistical process control. They should know that in any software development team whose performance is in statistical control, half of them would always be below average. They should focus on those members whose performance is out of statistical control (Li, (2000)).

8) Drive out fear of job insecurity. Employees must feel secure before they are willing to ask questions, make suggestions, or even expose their weaknesses by asking for help. The policy of long-term employment could easily drive out the fear of job insecurity. Moreover, any staff whose performance is out of statistical control should be offered help in retraining or reassignment. However, if one consistently rejects helps from one's coworkers or supervisors, a layoff may be the last resort (Li, (2000)).

9) Break down barriers between departments. Software development requires a collaborative effort between users and IS staff. For as long as we can remember, communication gap has been the major factor to many implementation failures. Furthermore, today's business system projects would most likely involve different functional areas and require expertise in database processing, client-server computing, and network installation, etc. Therefore, open communication among functional areas and general knowledge across disciplines are necessary for a successful system implementation. This requires appropriate education and training for team members to change their behavior and improve their knowledge (Li, (2000)).

10) Eliminate the slogans, exhortations, or targets for the workforce. Slogans do not build quality systems. MIS management should not ask for an impossible goal or schedule, or unreal level of productivity. They should post their progress to responding to suggestions and in helping the staff improve quality. Encourage the employees, put up their signs or slogans (Li, (2000)).

11) Eliminate the numerical quotas, and work standards. Quotas such as (metrics), goals (schedules), and work standards (unit times) address numbers, are not quality. A software development project that causes haste and non-conformities accomplishes nothing and services no one. Let the project members put up their goals. Managers should help people do a better job by reducing rework, errors, and waste. Everyone must work toward constant improvement, not the achievement of some arbitrary, short-term goals (Li, (2000)).

12) Remove barriers to pride of workmanship. All people are motivated. They would like to make quality products. However, a good workmanship relies on good materials, good tools, good methods, and right timing. Poor materials, broken tools, ineffective methods, or belated schedule are all barriers to pride of workmanship and should be eliminated. Let the software development team put its group identity or team members' names on the software product to take the credit (or the responsibility) to their work (Li, (2000)).

13) Institute a vigorous program of education and retraining for everyone. On-the-job training is effective, but slow, for an employee to acquire a skill set for a particular type of job. The new skill set is needed for the job in a short period. Management must set aside enough budgets to execute a generous education and retraining program for everyone to improve oneself. Under the TQM culture, all employees must know enough statistical method to understand the nature of variation, to manage the special causes of variation. Support for training employees to acquire necessary statistical method should be institutionalized (Li, (2000)).

14) Put everyone on work to accomplish the transformation. The TQM transformation is everyone's job. Everyone has a customer. Ask yourself who is the person receiving your work? All of us must identify our customers to determine precisely what our jobs are. Everyone belongs to a team, to work on the Plan-Do-Check-Act cycle, to address one or more specific issues, to find specific causes detected by statistical signals. Moreover, we must put management to work. Only management can change the culture and environment that dominate any individual's performance. Management must agree on their meaning and on the direction to take. They must acknowledge their mistakes, if any, and have the courage to change. They must explain to a critical mass of people in the organization why change is necessary and that the change will involve everybody. Obviously, people must understand the Fourteen Points to know what to do and how to do it (Li, (2000)).

6. CONCLUSION

Total quality management can be applied to any development process to improvement quality. Once you implemented TQM concept and methods, you are bound to continually improve your system and processes. There are different tools and quality measures to implement TQM improving software development process. Finally, the paper discusses Deming's Fourteen Points to software development and recommends asking yourself constantly, "What and how can I do it better next time?". In addition, take in consideration PDCA cycle (plan-do-check-act) to finish the work of the wheel.

7. REFERENCES

[1] Kaizen, I. M. (1986). The Key to Japan's Competitive Success. MacGraw-Hill, New York.

[2] Li, E. Y., Chen, H. G., & Cheung, W. (2000). Total quality management in software development process. *The Journal of Quality Assurance Institute*, *14*(1), 4-6.

[3] Kan, S. H., Basili, V. R., & Shapiro, L. N. (1994). Software quality: an overview from the perspective of total quality management. IBM Systems Journal, 33(1), 4-19.

[4] Glowalla, P., & Sunyaev, A. (2015). Influential Factors on IS Project Quality: A Total Quality Management Perspective.

[5] Lee, M. C., & Chang, T. (2005). Applying TQM, CMM and ISO 9001 in knowledge management for software development process improvement. *International Journal of Services and Standards*, *2*(1), 101-115.

[6] Joshi, A. C. Software Development Process And The Total Quality Management.

[7] Everhart, R., La Salle, A. J., & Khorramshahgol, R. (1995, June). Applying TQ principles to the requirements phase of system development. In *Engineering Management Conference, 1995. Global Engineering Management: Emerging Trends in the Asia Pacific., Proceedings of 1995 IEEE Annual International*(pp. 223-228). IEEE.

[8] Al-Qahtani, N. D., Alshehri, S. S. A., & Aziz, A. A. The impact of Total Quality Management on organizational performance.

[9] Kanji, G. K., & e Sá, P. M. (2002). Kanji's business scorecard. *Total Quality Management*, *13*(1), 13-27.

[10] Bradley, T. J. (1991, June). 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* (pp. 356-359). IEEE.

[11] Jammal, M., Khoja, S., & Aziz, A. A. (2015). Total Quality Management Revival and Six Sigma. *International Journal of Computer Applications*, *119*(8).

[12] Ashrafi, N. (1998). A decision making framework for software total quality management. *International Journal of Technology Management*, *16*(4-6), 532-543.

[13] Parzinger, M. J., & Nath, R. (2000). A study of the relationships between total quality management implementation factors and software quality. *Total Quality Management*, *11*(3), 353-371.

[14] Powers, J. (1993). TQM in software development organizations. *Quality Progress*, *26*(7), 79-80.

[15] Parzinger, Monica J., and Ravinder Nath. "TQM implementation factors for software development: an empirical study." *Software Quality Journal* 7.3-4 (1998): 239-260.

[16] Subramanian, G. H., Jiang, J. J., & Klein, G. (2007). Software quality and IS project performance improvements from software development process maturity and IS implementation strategies. *Journal of Systems and Software*, *80*(4), 616-627.

[17] Helio Yang, Y. (2001). Software quality management and ISO 9000 implementation. *Industrial Management & Data Systems*, *101*(7), 329-338.

[18] Omachonu, V., Johnson, W. C., & Onyeaso, G. (2008). An empirical test of the drivers of overall customer satisfaction: evidence from multivariate Granger causality. *Journal of Services Marketing*, *22*(6), 434-444.

[19] Chappell, D. (2013). The three aspects of software quality: Functional, structural, and process.

[20] Dhillon, B. S. (2013). *Computer system reliability: Safety and usability*. CRC Press.

[21] Dhillon, B. S. (2007). Software Quality. Applied Reliability and Quality: Fundamentals, Methods and Procedures, 151-164.

[22] Wang, R. Y., Storey, V. C., & Firth, C. P. (1995). A framework for analysis of data quality research. *IEEE transactions on knowledge and data engineering*, *7*(4), 623-640.