MERIT RESEARCH JOURNALS
www.meritresearchjournals.org

Merit Research Journal of Business and Management (ISSN: 2408-7041) Vol. 8(3) pp. 032-039, July, 2020
Available online http://www.meritresearchjournals.org/bm/index.htm

DOI: 10.5281/zenodo.3951027

Review

Total Quality Management in Artificial Intelligence System Development

Khadair K. Hmood

Hashemite University, Zarqa, Jordan

E-mail: hmood@hu.edu.jo

Abstract

The current development and the wide use of Artificial Intelligence (AI) applications and systems have increased the demand for a faster development processes that could compromise the quality assurance of the whole development life-cycle. Quality plays a vital role in software reliability and easier maintenance after sale while the quality requirements are not well-defined in most software production processes due to rapid delivery milestones. This paper discusses the essential and essence of the Total Quality Management (TQM) for AI system development and summarizes the similarities between products and systems development life-cycles. In this paper, we propose the implementation of TQM throughout the different stages in AI system development and AI system training process. Finally, this paper recommends a set of measures to ensure a minimum level of quality assurance before system deployment.

Keywords: Artificial Intelligence System, Development, Process Control, Quality Assurance, Total Quality Management

INTRODUCTION

TQM is a set of principles, best practices, and philosophy first adopted in Japan (Haag et al., 1996). It is an organizational management concept of “the voice of the customer” which promotes “doing the right thing at the right time”. TQM tracks the overall quality measures including quality design and development, quality control and maintenance, quality improvement, and quality assurance within the different teams of the organization.

Software systems have become an essential part of our daily life. A variety of different domains ranging from medical to education to toys are now based on software applications (Benkler, 2019). Billions of dollars are being transferred and used for stock speculation daily based on software applications and real-time data updates around the world (Brzeszczyński and Ibrahim, 2019). However, due to rapid delivery timelines, at times, the quality standards of the software may not be defined. Therefore, ensuring a quality and a reliable system through quality software is a necessity and essential rather than an option to add to a software product (Behutiye et al., 2020). Total Quality Management (TQM) aims at improving the quality assurance process throughout the different stages of software development and ensuring that different software development teams are following the standard quality level of a company (Li et al., 2000).

Computer industry reports illustrate that maintaining a software after-sale has the largest cost portion and hence limits the software development firms from developing new features while putting a lot of efforts in maintaining and fixing current features (Gupta and Sharma, 2015; Khanna et al., 2017; Yau and Collofello, 1980). Several companies have integrated a certain level of quality assurance and quality control into their development process and that reflects on better services and products. However, the rapid growth in artificial intelligent applications development and the high pressure from

competition to deliver an artificial intelligent application faster to the customer compromises the quality and reliability of the applications if not implemented from the very start (Lu et al., 2018). Research suggested implementing quality assurance and control processes at the early stages of software development can highly reduce cost if implemented at late stages of development (Hovorushchenko and Pomorova, 2018). In addition to reducing the overall cost, if good quality management is implemented at the early stages, a long-term existence and better reputation are reflected directly on the software development organizations and ensures a reliable product that gains the customer trust in the organization name.

The TQM is centered on enhancing the customer-centric practices for conveying quality (Khanam et al., 2016). The principles of TQM cover not only the development process but can be stretched to marketing, data storage, finance and R&D within the organization. The overall effect of TQM on software development and computer industry has been investigated widely by researchers (Alhazmi et al., 2017). Software system researchers and practitioners are studying and analyzing this effect and the TQM effect on other field such as manufacturing industry, health, education, and different governments sections and compare how those other organizations have adapted best practices of TQM. As a matter of fact, implementing TQM principles and practices in software industry are more challenging than other manufacturing industries. Software systems are implemented by several teams that are not necessarily within the same geographical location, working on top of each other's implementation and overlaps in development stages. All these factors make it harder, however, such a specific scenario has been handled by source control software that tracks these implementation and code integration overlaps. Yet, there are other issues that could compromise the code or software quality and adapting the TQM practices adds an enormous value to the computer industry.

Quality Management in Software Development

The TQM principles and practices can improve the production development and software quality and not hinder the application delivery times. The customer-centric concept in TQM ensures a customer-motivated economy and better reliability in software development and delivery. The philosophy focuses on a constant enhancement and improvement to achieve higher-value service or product. On the other hand, user-customized software could have extra features that fit a specific user needs that are not required or purchased by other users. In this case, having a standard base software with certain qualities must reflect on the added or higher level features.

The adaptation of TQM principles and practices should begin from the topmost administration level all the way to the last layer in the organization hierarchy. This ensures that the same practices are implemented by the different teams and the quality assurance lies at the same standard level within the same organization. As mentioned earlier in this paper, implementing TQM is software industry is more challenging than other fields due to the software production life-cycle that covers from 7 to a minimum of three different stages; design, development, and testing are the three major stages in the software development life-cycle. Figure 1, shows an agile methodology as one of the most widely used software development life-cycle (Abrahamsson et al., 2017; Kazim, 2017). Carrying the same standard quality level among the different stages adds extra work and harder consideration on the management. Therefore, adapting TQM recommendations and guidelines throughout the different teams within the same department and the different departments in the organization would relieve the management from a lot of pressure and compromises. The development of artificial intelligent applications is not different than the other software applications in term of development life-cycle (Sharma, 2017). It has other challenges on top of the already existing challenges if compared to the service they provide. Artificial intelligent systems have a great similarity in terms of functionalities, but the quality is the determinant of which comes first. Of course, the decisions of what methods or decision-making algorithm to use along with the amount of training of the inference engine have the highest impact on the software popularity, and the leading factors to why the customers are leaning towards a certain product. The long-term existence and trust from customers are guaranteed by other software quality factors.

Literature Review

In the literature, several research articles have discussed the need to integrate the TQM principles into software development life-cycle. Many of these articles studied the application of TQM in specific stage of the software development life-cycle such as Quality Metric Management (QMM) (Osmundson et al., 2003) or proposed application-specific tools such as Total Quality Data Management (TQDM) (Wang, 1998). There exist some international standards for specific development stage i.e. ISO 9001:2015 that fits the software requirement specification and analysis. The ISO 9001: 2015 sets the necessary quality management requirements for the organization to demonstrate their ability to continuously develop products that meets the customer needs and enhances the customer satisfaction by persistent improvements to the software functionalities and customer requirements (Lazarte, 2015). TQM has

Figure 1. Agile Software Development Life-Cycle Methodology (Sami, 2012)

been defined as a management technique for continuously improving the performance at every hierarchal level in administration and in every department to ensure customer satisfaction. TQM integrates fundamental management and technical tools to achieve the total quality by encompassing every part of the organization. It also requires continues planning and training to guarantee all employees at the organization are aligned with the same quality level set by the top management.

The Artificial Intelligent (AI) applications fall under software development field of study. It can be a stand-alone application such as voice recognition systems or it can be part of a larger software system such as lane-departure in new cars (Jackson, 2019). In either case, the standard software development life-cycle is applicable to AI applications and it might also include extra stages such as training of the AI engine for each application. It also has more common updates and features as the training stage evolve and more data are available. Therefore, a good quality management practices and guidelines must be implemented and followed. Software engineering field of research has focused on best practices for the different stages of the software development life-cycle while reusability, portability, security, reliability, and many other are part of the non-functional requirements that must be met as recommended by the software engineering practitioners and researchers (Mall, 2018). On one hand, classifying those as non-functional requirements sends the feeling that those are not part of the “needed” requirement (Eckhardt et al., 2016). On the other hand, even if the user ensures the implementation of those non-functional requirements, it would only ensure that certain qualities are implemented during the implementation stage of development. The other stages of development have to integrate other qualities.

April et. al. (2012) described that the software development processes focus on the problem-solving and pay less attention on the quality control. The authors discussed the software production processes in the computer industry where the continuous software support and improvement has greater impact and cost than the initial development process (April and Abran, 2012). While fixing existing issues after software release is usually a part of the software improvement, it should not be the major part of the after-sale support and maintenance. Improving the software and adding features should be the focus of any organization for continuous success and existence in the competition and the market. TQM provides the framework to standardize and formalize the various steps and processes involved in managing the organization’s quality not only their software product. The TQM recommendation fits the software maintenance best practices theories where the continuous development is the main focus while including finding existing issues, determination of the root cause, resolving those issues, and follow ups on these and other issues.

Software products are categorized as services more than products due to absence of tangible product which makes it harder to standardize the quality of the product; mainly due the subjective nature of measuring service products quality (Song, 2017). However, in the long run and with consistent feedback from customers, a quality level can be determined and set as the minimum desired

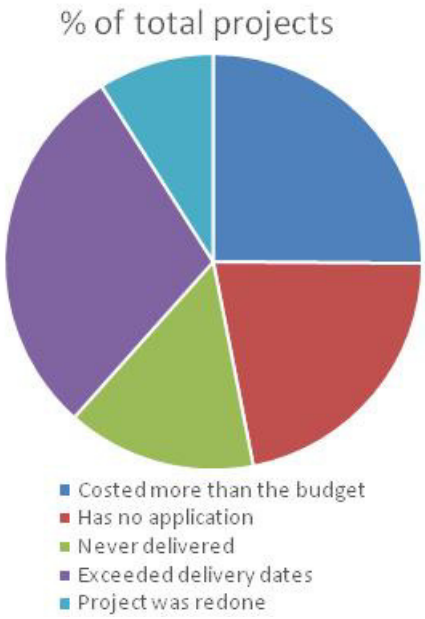

Figure 2. Ratios of software projects with issues (Cohen, 2019)

level but that comes only with good quality management practices and implementation. Several trade-offs between performance and quality control can be raised throughout the development life-cycle and these have to be addressed once and standardized between the different teams and departments in the organization. The opposite can happen if these trade-offs are remained subjective and based on the developer experience and choice. The software quality or performance can vary from one version to another and would highly disappoint the customers by not having a reliable quality and performance (Koczkodaj et al., 2018). Calabrese et. al. (2015) studied the relationship between TQM principles and practices with the overall business performance (Calabrese and Corbò, 2015). The highlighted characteristics of an organization that adapts TQM practices among the different departments are summarized as better teamwork, efficient communication and understanding between the organization and their clients, effective administration, higher employee contributions, continues customer feedbacks, and developed management trust and support. Figure 2

John et al. (2016) studied and analyzed the research field of software quality management and published a survey on the different software quality management techniques. The authors stated that the software quality can be classified into five major categories based on how the literature discusses the software quality in relationship to different quality management concepts and principles. The author further suggests that the different articles in the literature have studied the applicability of TQM, ISO 9000 series of standards, capability maturity model level, defect prediction models, and other quality management practices. The findings in this study are presented in Figure 2. These findings are not surprising to the computer industry and software developers. However, they are worrisome for the long run and existent of any company if no proper quality management techniques are implemented at the different level of the organization (John and Kadadevaramath, 2016).

TQM in Artificial Intelligent Systems

Recent years have seen a rapid growth in AI applications and a rise in competition between large and small software development organizations. These rapid and rising competition can come at the cost of software development quality or the software quality itself. While these AI systems are implemented in critical and sensitive fields such as health or stock markets, the minimum quality assurance level has to be corresponding to the application field. The development of AI systems and the classification engine consists of the two main stages; one is the traditional software development life-cycle, and the other is the training process of the classification engine itself. These two can also be seen as two inter-correlated software that has to be aligned to the same quality standard.

AI is the new era in computer industry and most companies from banks to automobile manufacturer have already adapted or implemented AI systems. Artificial intelligent is a data-driven software solution that relies on

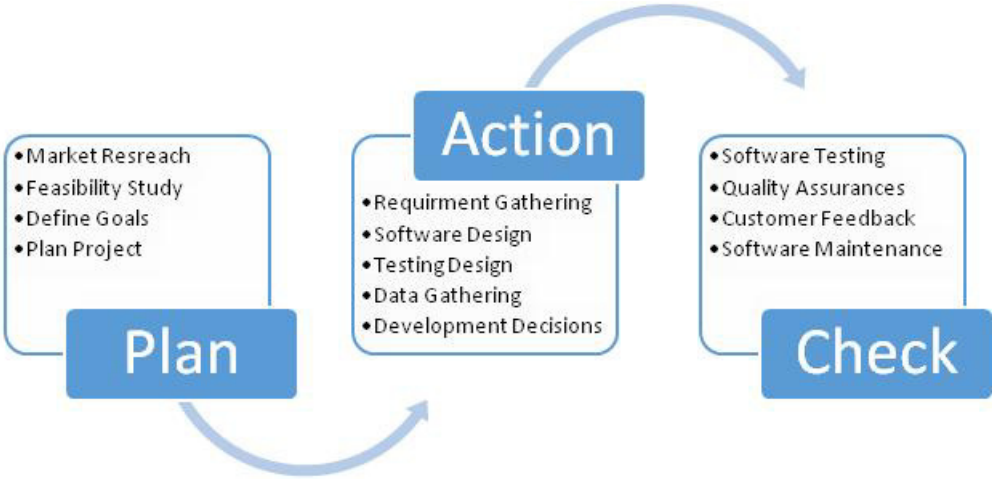

Figure 3. Total Quality Management Tools in an Organization

data only; it uses sets of statistical and probabilistic methods and techniques to determine the outputs (or results) without human-driven assumptions and inputs like the case in other software systems. However, there are sets of one-time settings that need to be decided and picked at the beginning of the application use. These settings have several thresholds such as speed vs. cost, speed vs. accuracy, or accuracy vs. sensitivity. There are other factors during the production involved such as the device size (where the AI application would run) and that is crucial in some industries i.e. automobile manufacturing or smart home devices. These settings and design choices are directly impacting the customers and their satisfaction or acceptance of the AI software solution on one hand, and the cost and challenges of maintaining and improving this software after delivery on the other. Hence, the early implementation of customer-centric quality management arises as a necessity rather than a need.

TQM sets a good example for best quality management practices in several other industries and has been the topic of discussion in software development industry for some years now. Integrating the Quality Seven (Q7) and Management Seven (M7) tools that are commonly used with TQM into the software development life-cycle have been discussed and analyzed in the literature, yet discussing how these concepts and other quality assurance processes are adapted at different administration levels is still immature and there are no standardized guidelines (or very few at best) for integrating best practices of quality management throughout the whole organization in software industry Figure 3.

TQM in the Organization

TQM defines different principles that should be applied to the different organization's processes. Figure 3, represents the different quality management tools, questions they answer, and who is responsible in for each tool. It defines Business System Planning (BSP) as a tool for planning the organization strategy and plans for new product. The BSP sets the guidelines to ensure a standard quality assurance level is implemented in the different geolocation offices and the different department of the organization. It also provides the AI developers with the organization's target quality level and train the different team on adapting to these qualities. Critical Success Factor (CSF), proposed by IBM, suggests developing a system for planning and process control within the organization that can be used for analysis of these critical success factors. On the other hand, Balanced Scorecard (BSC) can be implemented in AI software industry to ensure that the organization's strategy is being followed and applied successfully during the training of the AI system and the development for production. Several examples of the impact of this tool in different industries have shown the importance of integrating BSC method to AI system development and production companies. Finally, Quality Function Deployment (QFD) method guarantees the different departments of the organization are aligned to achieve plan and the goal of the company. If the company is producing a voice recognition system for smart home, the different departments right from marketing, research, management, finance, developers, to human resources have to get involved in supporting this goal.

Applying the TQM principles and techniques into an organization requires certain guidelines to be met for a successful integration of TQM in the AI software development at the different organization levels. There are different guidelines that could be set by an organization. Here we will cover the core seven guidelines.

1. *Understandability.* Quality is the responsibility of every-

one in the organization and not limited to management or developers only. The AI project has to be well-described and understood by the different leaders, managers, analysts, architects, designers, developers, and users to reach a concise, complete, consistent, and well-structured AI system.

2. Flexibility. The AI software design should guarantee a flexible design for several reasons including cost reduction during maintenance and quality software during development. Flexibility in AI systems is a core feature due to the fact stated in section 4.2 that the AI system must adapt the agile development methodology. The different customization for different users and continuous deliveries of improved and new features would require high design flexibility.

3. Efficiency. After an appropriate market research and analyses of competitors, the efficiency of the planned software should be planned. This is mostly covered by the feasibility study of the software project. A well presentation of the quantity and quality of competitor's AI solutions and fair analysis of the competition and survival chances in the market has to be documented. This document is to be presented by management and shared between the different departments and teams to support the organization goal.

4. Software Quality. This is the responsibility of the actual designer and developer of the AI software. Both in training stage and developing stage, the designers have to bear in mind the quality that is required and implement it during designing the project and selection of development tools and resources. Developers will follow the design specifications of the AI software and do the coding at the same quality standard set by the organization. Designers and developers will come into several design decisions to improve the low-level coding quality and this has to follow the overall quality standard of the organization.

5. Quality Assurance. The software testers have to get involved and take responsibility for this check. It confirms that the quality of the AI software meets the quality that is set by the management and aligned with other departments in the organization. Several tests with specific goals should be planned ahead of development to cover the different aspects of the AI software.

6. Long existent. AI software developers and computer industry in general must consider the long usability of the software. The accurate and right amount of support to software after delivery must be planned ahead not only for budgeting purposes but for building a trust in the organization's name.

7. Software Quality requirements. A proper consideration of the non-functional requirements that are all directly related to the software and coding quality itself has to be discussed between the different teams. A brainstorming session has to happen on regular sessions to get the developers feedback and for them to share their experiences. At the end, these developers are the one that are building this AI project and analyze the training data with close look at what quality decisions would best fit the AI solution.

TQM in AI System Life-Cycle

The AI system life-cycle can adapt to most of the agile software development life-cycles. The agile life-cycles guarantees the continuous support, improvement, and deliveries of the software to meet the customer expectations. Like some other software systems, AI systems are customized to fit the user needs in most cases and it also keeps learning to boost the accuracy whenever there is more data. Setting a quality level to meet the different customization of the software between all customers and continuously throughout the different releases of the software should start at the project planning stage. The different quality processes go along the different development life-cycle and certain quality checks have to be carried out on for the different AI software development life-cycle.

1. *Requirement Analysis*. This is the first stage in AI system development life-cycle where the system requirements gathered from market research and customer feedback have to be analyzed and documented. The result of this analysis is the requirement specification document. Several research from the existing literature have identified this as the most crucial step and about 80% of software project failure is rooted back to the requirement specification stage. The system requirement must be well analyzed, presented, and communicated between the different team members. Beside the functional requirements, the quality requirements, or the so the so called non-functional requirements, have to be addressed here in this document along with the systems test and evaluation plan. TQM practices ensure that the quality of the requirement specification document is designed and presented while covering the organization goal.
2. *Design*. In this stage of the AI system development life-cycle, the training and development of the AI system have to be designed. Designing the dataset separation for training or testing and designing the code classes and responsibilities are documented here. The outcome of this stage is the design document and contains different pieces of the project with the goal of each. Issues in design document is hundreds time costlier that issues in the later stages. It serves as the backbone of the software development processes and qualities ensured in the design document is guaranteed in software development stage. Several quality insurances are considered here including flexibility, portability, reusability, and maintainability. Applying TQM principles and tools in the design document helps building successful and usable AI software; the training design is a crucial part of the designing process to ensure the

classification goal, class labels, dataset size, and most importantly the application of this AI system.

3. *Implementation*. This is the real coding and training of the AI system that is based on the requirement specification and system design documents. At this stage the translation of the quality requirements and settings is done and integrated throughout the entire software training and development. There are several other quality decisions that might come up during the implementation or the training process; those decisions have to be presented at the brainstorming sessions to collect different feedbacks about best coding practices. TQM principles are setting good decision-making practices where the customer has always to come first. It recommends the software developer to look at how the customer would like this feature and trade-off decisions have to be taken with what best suits the customers i.e. security is more important than performance (speed) for this particular customer. Always a good balancing of the different quality requirements is totally dependent on the customers and the application.

4. *Software Testing*. Software testing should be planned during the design phase and documented in the software design document (Lewis, 2017). Several tests at different software levels are performed at this stage e.g. unit testing, integration testing, and system testing. Each test is designed with one specific goal and assumptions if there are any. While alternative scenarios are provided in case of unit or system failure e.g. show an adequate message to the user of any unexpected issue or internally handle an error. TQM practices recommend the user acceptance test at this stage before delivery where a sample software can be shown to the customer and have the user use it and provide the organization with a feedback. This feedback has to be studied well and if some customization, fixes, or error have to be considered before delivering the AI software.

5. *Evolution and maintenance*. This phase is crucial with respect to budgeting and building the organization name. A good design and implementation decisions directly affect an easy and feasible maintenance. AI system are evolving by nature and continuous build and support is required, therefore, implementing TQM principles and practices at early stages of the project will reflect positively at the end of the project development. There is no timeline for maintenance where it can be extended to years of support and improvements. Statistical Process Control (SPC) is a good method to track the maintenance and ensure active, reliable, and quality maintenance is practiced by programmers. It requires the software developers to play an active role at this stage to support the system.

CONCLUSION

This paper has discussed the application of total quality management (TQM) principles and best practices in artificial intelligent (AI) software solutions. The TQM adaptation by AI solution provider would reflect in several gains both financially and building trusted name for the organization. This paper analyzes the different TQM tools and practices and provides a strong recommendation for setting and utilizing these tools and guidelines at the different development practices and AI system development life-cycle. The AI system life-cycle has also been discussed and different TQM practices are proposed to best fit each phase within the development life-cycle. This paper analyzes the advantages of applying the TQM principles and tools throughout the different teams within the organization and the different managerial levels have great impact on a successful AI project. It also suggests the implementation of different TQM tools throughout the different organization teams. The customer-centric concept in TQM encourages all levels of employees in the company to work towards one goal which is customer satisfaction which is the ultimate goal for every company.

ACKNOWLEDGMENT

This research has been partially supported by the Hashemite University.

REFERENCES

Abrahamsson PO, Salo J, Ronkainen, J. Warsta (2017). "Agile software development methods: Review and analysis," arXiv preprint arXiv:1709.08439

Alhazmi E, W. Bajunaid, A. Aziz (2017). "Important Success Aspects for Total Quality Management in Software Development," Int. J. Comp. Applications, vol. 157, no. 8, pp. 8-11

April A, Abran A (2012). Software maintenance management: evaluation and continuous improvement, John Wiley & Sons.

Behutiye W, P. Karhapää, L. López, Burgués, Xavier, S. Martínez-Fernández, AM Vollmer, P. Rodríguez, X. Franch, M. Oivo (2020). "Management of quality requirements in agile and rapid software development: A systematic mapping study," Information and Software Technology, vol. 123, pp. 106225 - 106248

Benkler Y (2019). "Don't let industry write the rules for AI," Nature , vol. 569, no. 7754, pp. 161-162

Brzeszczyński J, BM. Ibrahim (2019). "A stock market trading system based on foreign and domestic information," Expert Systems with Applications, vol. 118, pp. 381-399

Calabrese A, M. Corbò (2015). "Design and blueprinting for total quality management implementation in service organizations," Total Quality Management & Business Excellence, vol. 26, no. 7, pp. 719-732.

Cohen H (2019). "Project Management Statistics: 45 Stats You Can't Ignore," 07 02 2019. [Online]. Available: https://www.workamajig.com/blog/project-management-statistics. [Accessed 20 01 2020].

Eckhardt J, A. Vogelsang, DM Fernández (2016). "Are" non-functional" requirements really non-functional? an investigation of non-functional requirements in practice," in the 38th International Conference on Software Engineering

Gupta A, S. Sharma (2015). "Software Maintenance: Challenges and Issues," Int. J. Comp. Sci. Eng. (IJCSE), pp. 23-25

Haag S, M. Raja, LL Schkade (1996). "Quality function deployment usage in software development," Communications of the ACM, vol. 39, no. 1, pp. 41-49

Hovorushchenko T, O. Pomorova (2018). "Methodology of evaluating the sufficiency of information on quality in the software requirements specifications," in IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)

Jackson PC (2019). Introduction to artificial intelligence, Courier Dover Publications

John B, RS Kadadevaramath, EA (2016). Immanuel, "Recent Advances in Software Quality Management: A Review," Merit Research Journal of Business and Management , vol. 4, no. 3, pp. 018-026

Kazim A (2017). "A Study of Software Development Life Cycle Process Models.," International Journal of Advanced Research in Computer Science, vol. 8, no. 1, pp. 15-23

Khanam S, J. Siddiqui, F. Talib (2016). "Role of information technology in total quality management: a literature review," Int. J. Adv. Res. Comp. Eng. Technol. vol. 2, no. 8, pp. 2433-2445

Khanna S, AJ Shah, L. Ramanathan (2017). "Software Maintenance: Challenges and Issues and Models for Reducing the Maintenance Cost," Int. J. Adv. Res. Comp. Sci. vol. 8, no. 3,

Koczkodaj WW, P. Dymora, M. Mazurek and D. Strzałka (2018). "Consistency-driven pairwise comparisons approach to software product management and quality measurement," in International Conference on Dependability and Complex Systems, Springer, Cham.

Lazarte M (2015). "ISO 9001:2015 - JUST PUBLISHED!," 23 September 2015. [Online]. Available: https://www.iso.org/news/2015/09/Ref2002.html. [Accessed 10 August 2020].

Lewis WE (2017). Software testing and continuous quality improvement, ORC press.

Li EY, HG Chen, W. Cheung (2000). "Total quality management in software development process," J. Quality Assurance Institute, vol. 14, no. 1, pp. 4-6

Lu H, Y. Li, M. Chen, H. Kim, S. Serikawa (2018). "Brain intelligence: go beyond artificial intelligence," Mobile Networks and Applications, vol. 23, no. 2, pp. 368-37

Mall R (2018). Fundamentals of software engineering, PHI Learning Pvt. Ltd.

Osmundson JS, JB Michael, MJ Machniak, MA Grossman (2003). "Quality management metrics for software development," Information & Management, vol. 40, pp. 799-812

Sami M (2012). "Mohamed Sami, Software Engineering & Architecture Practices," 15 03 [Online]. Available: https://melsatar.blog/2012/03/15/software-development-life-cycle-models-and-methodologies/. [Accessed 20 01 2020].

Sharma MK (2017). "A study of SDLC to develop well engineered software," International Journal of Advanced Research in Computer Science, vol. 8, no. 3, pp. 520-523

Song W (2017). "Requirement management for product-service systems: Status review and future trends," Computers in Industry, vol. 85, pp. 11-22.

Wang RY (1998). "A Product Perspective on Total Data Quality Management," in Communications of the, 1998.

Yamada S, Y. Tamura (2016). OSS Reliability Measurement and Assessment, Switzerland: Springer International Publishing.

Yau SS, JS Collofello (1980). "Some stability measures for software maintenance," IEEE Transactions on Software Engineering , vol. 6, pp. 545-552